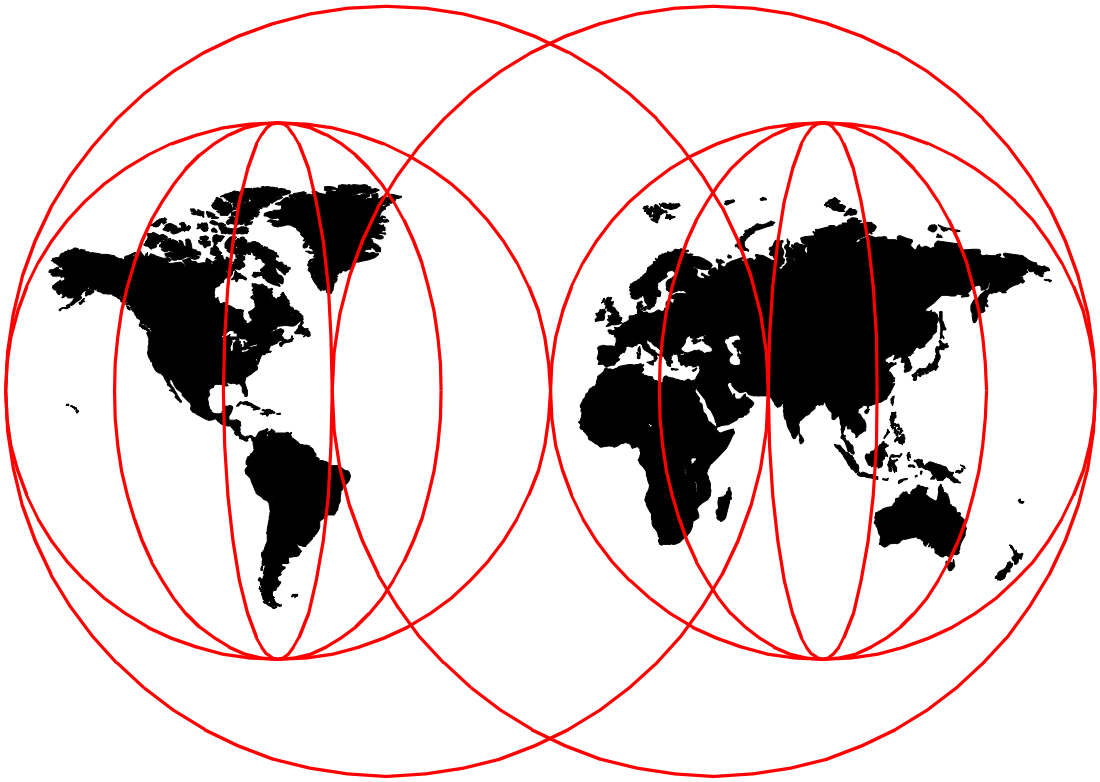
Converting from Oracle AIX to DB2 for OS/390

Paolo Bruni, Debra Eaton, Gregery Green, Luca Montini



International Technical Support Organization

www.redbooks.ibm.com

SG24-5478-00

International Technical Support Organization

SG24-5478-00

Converting from Oracle AIX to DB2 for OS/390

December 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix E, “Special notices” on page 265.

First Edition (December 1999)

This edition applies to Version 6 of IBM DATABASE 2 Universal Database Server for OS/390, Program Number 5645-DB2.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This redbook describes how to define, implement, and document a migration project from Oracle on AIX to DB2 for OS/390. A sample scenario based on a subset of a real application developed in C Language has been used throughout this project. This allowed us to gain first-hand experience and gave us a starting point to evaluate alternatives and to extrapolate to more general considerations applicable to a variety of applications.

This redbook will help you plan for and conduct a conversion of your application from an Oracle AIX environment to a DB2 for OS/390 environment. It particularly applies to Version 6 of IBM DATABASE 2 Universal Database Server for OS/390, Program Number 5645-DB2.

It is primarily intended for database administrators and system designers with Oracle AIX and/or DB2 for OS/390 background: several considerations are applicable to the porting of general data and programs from the AIX platform to the S/390 platform.

We start with an introduction to the project and a summary of our conclusions. This will be especially useful for people responsible for deciding and defining the scope of similar projects.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization San Jose Center.

Paolo Bruni is a Senior Data Management Specialist for DB2 for OS/390 at the International Technical Support Organization, San Jose Center, where he conducts projects on all areas of DB2 for OS/390.

Debra Eaton is a Field Technology Sales Specialist at the IBM Software Migration Project Office in Chicago, IL. She has been with IBM for 15 years, and her areas of expertise on migrations include Oracle and DB2. Debra has written extensively and has presented to several conferences on migration topics.

Gregery Green is a Staff Application Programmer in Mechanicsburg, PA. He has 15 years of experience in application development and maintenance. He has worked at IBM for 3 years. His areas of expertise include most mainframe platform languages and subsystems.

Luca Montini is an Information Technology Specialist working for IBM Italy. He has 3 years of experience in system and database management, and he has been supporting customers in the process and downstream industry areas. His areas of expertise include Oracle RDBMS and application programming. Luca holds a degree in Mathematics from La Sapienza University of Rome.

Thanks to the following people for their invaluable contributions to this project:

Rich Conway
Bob Haimowitz
Vasilis Karras
Frank Kyne

IBM International Technical Support Organization, Poughkeepsie Center

Emma Jacobs
Yvonne Lyon
Elsa Martinez
Joerg Reinschmidt

IBM International Technical Support Organization, San Jose Center

Roger Miller
Rosie Rushidally
Hugh Smith

IBM Santa Teresa Laboratory

Bart Steegmans

IBM Belgium

Chris Tett

Mantech Systems Solution Corporation

Corrado Venturini

DB2 Independent Consultant in Italy

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO redbook evaluation” on page 281 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Introduction

The objective of this redbook is to help you plan for and conduct a conversion of your application from an Oracle AIX environment to a DB2 for OS/390 environment. To do so we have chosen a sample scenario based on a subset of a real application and we have used it throughout the project, documenting the migration step by step. The selection of a specific application and the actual implementation of the migration have allowed us to gain first-hand experience and to test realistic alternatives. On the other hand, the characteristics of this specific application cannot be valid for all possible permutations of application variables. We have also tried, whenever possible, to go beyond the sample scenario and to extrapolate to more general considerations applicable to a larger variety of environments.

This chapter contains an introduction to the project and a summary of our conclusions, and it can be useful to people responsible for decision-making and defining the scope of similar projects.

1.1 Decision to convert

Converting an application across different platforms and different databases is certainly not a trivial task. The decision to convert is generally made at high level and when there is full justification in terms of costs and expected returns on investment. The major issues that bring up the need to convert, and are the main components for building a business case, are related to:

- Performance

  Aspects of performance include scalability, availability, data movement, response time, and the ability to support multiple query workloads.

- Configuration costs

  Realistic cost assessment is based on overall development, maintenance, and tuning cost and the full exploitation of current investments, both in terms of skill sets and reduced licence costs.

- Data integration

  Market trends are highlighting the importance of enterprise servers and the need to avoid data structure fragmentation in order to increase the value of business-critical systems; fragmentation may cause cross-functional currency and consistency issues and hamper innovation.

- Data infrastructure

  Data is no longer an application-specific resource, but an enterprise-wide tool to provide critical business information for a competitive advantage; often it is not enough to navigate through the data, but it is necessary to invest in their infrastructure in order to more easily integrate enterprise views of data.

  The deployment of enterprise resource planning, customer relationship management, and business intelligence systems may raise the need to reconsider the current choice of environment for important applications and appreciate the strength and value of DB2 and S/390. OS/390 with DB2 and Sysplex configurations offer scalability, high capacity, and high availability.

Other reasons for choosing DB2 and S/390 are mainly the ability to leverage existing skills and investments, reduced licence costs, performance guarantees, and synergies from data originating predominantly from other S/390 systems.

Several documents can help in making this decision. Please refer to *Selecting a Server - The Value of S/390*, SG24-4812-01, and its related bibliography for more information.

1.2 Project overview

The project consisted of taking an existing application package based on Oracle and UNIX and converting a meaningful subset of it to use DB2 on OS/390.

The chosen application is based on the IBM Computer Integrated Process and Refinery Operations System (CIPROS) solution package for oil industries, program number 5799-A28. Please refer to Chapter 2, “Project scenario” on page 5, for more details on CIPROS. The client/server architecture of CIPROS solution provides a good case study for a conversion project because:

1. It is a real application now running in several refineries all over the world, so it does not constitute just a “case study”, but it represents instead a “project study”.
2. It contains most of the needed functions to test conversion methodology and technical paths.
3. The objective of using CIPROS on DB2 for OS/390 is a real customer requirement.

Since not all the CIPROS features could be covered during the limited time available for this project, we have isolated a meaningful subset of the application package for our conversion case study.

The team achieved a working application, starting with an empty RS/6000 machine, with only three people working full time for one month. The project elapsed time was two months, but about half of the time was spent documenting all findings and writing the book. Naturally we encountered several problems, but this achievement proves the viability of such conversion.

Chapter 2, “Project scenario” on page 5 contains more information on the application package chosen, and Chapter 3, “Conversion process” on page 13 describes the methodology used. The text of the book follows the general methodology as applied to our particular pilot case. The following chapters describe the conversion by topic: Chapter 4, “System environment” on page 45, Chapter 5, “Database conversion” on page 61, Chapter 6, “Data conversion” on page 109, Chapter 7, “Application conversion” on page 141, and Chapter 8, “Testing, change control, and tuning” on page 183.

The general methodology involves a three stage process. The process in this project ends after stage one, the definition of the strategy, and stage two, the proof of concept — that is, after we have selected part of the total application, set the strategy, and converted programs, database, and data, providing a tested proof of conversion. It does not include stage three, the full implementation and cutover.

The database design, the data, and the batch applications were successfully converted, though unfortunately, in the short time available, we were unable to convert the on-line part of the application. Nevertheless, we feel that the project reached its objectives and demonstrated the viability of the conversion.

Summary of considerations:
The team spent approximately half of the time satisfying the need to document the findings in a meaningful way.

Implementing a database design on DB2 and OS/390 similar to the one on Oracle and AIX and converting the application to run on OS/390 proved to be a challenging but successful task. At the end of the time scheduled for the project we were able to complete it and implement the batch C language jobs. The programs executed successfully and were able to perform embedded and dynamic SQL against the converted data in DB2. The interactive portion of the pilot application was not tested and was excluded from the project because the requested third-party software was not received in time to complete the task within the confines of the residency. We do not think that it would have added much more to our learning process, but it could have helped in assessing alternatives for the middleware architecture.

Of the two main goals of the project — converting the data and replicating the database design in DB2 V6 for OS/390, and converting a specific subset of the C language application to run on OS/390 — the database conversion was certainly the easier task. The common relational logical model and the similarities between the two databases in terms of functions were the contributing factors. It is our opinion that the functions across the two databases are mapping reasonably well today and will map even better in the future.

Moving the actual raw data from AIX to OS/390 took a bit longer than expected because we needed some work-around for populating the tables with the DB2 LOAD utility. The LOAD utility is certainly faster than any program using the normal INSERT SQL command, but has specific rules on the way that input data must be presented. One rule is about variable length record, so that the rows, formatted and padded with blanks at the maximum length when unloading from Oracle, had to be converted to eliminate blanks at the end of the variable length fields, and allow binary length fields to be inserted in the load source files.

Another factor which contributed to complexity was the intricate relationships among the tables based on foreign keys defined in the sample application. The referential integrity considerations required a good analysis of the logical data design. Attention had to be paid to the order of loading and the correctness of the data definition to ensure that no recovery was needed during loading. When pressed with time and not totally familiar with the source data design, several errors may happen.

We experimented with an alternative solution using DataJoiner. Data transfer using Datajoiner was much easier and required no data conversion or JCL; variable length field and referential integrity are maintained with normal DB2 rules enforcement during SQL insert. The drawbacks may be the planning for the product itself, the need for some system programming time when establishing communication, and, most of all, its suitability for moving large amounts of data. Coexistence requirements during the phased implementation of a conversion project, though, can make the investment worthwhile on its own right.

In our scenario the amounts of data were small. If large amounts of data must be moved, using the DB2 LOAD utility would certainly be the more efficient approach, providing better performance and allowing no logging, worth the relative investment in data pre-conditioning. The database physical design must be implemented on the OS/390 server for either approach to work.

Converting the C language code to enable it to use DB2 on OS/390 was the more extensive undertaking. The DCLGEN feature of DB2 provided savings in producing usable C language structures for the tables. However, the design of the application on AIX extensively utilized pointers to host variables. This function is not supported in C by DB2 V6 for OS/390. Host variables had to be specifically defined. Other differences are related to the usage of environment variables and the getenv and __getenv functions of C. Finally some initial learning was needed for the application converters in order to configure and use the precompile, compile, prelink, and link available on OS/390.

The feasibility of the conversion was proved; it can certainly be done given circumstances similar to the pilot effort and assuming there are sufficiently experienced and skilled people available to do the work. We stress that highly skilled personnel are needed on both sides of the conversion if the objectives must be reached in a short time. Also, communication with the site systems personnel needs to be clear, effective, and flow freely, since we found that system support was crucial to our project success.

Chapter 2. Project scenario

This chapter provides a description of the project scenario, constituted by the conversion of the *process* and *laboratory* segments of the IBM Computer Integrated Process and Refinery Operations System (CIPROS) solution for oil industries, program number 5799-A28.

2.1 What is CIPROS?

IBM experience with several refineries and process plant information systems has shown the existence of common situations:

- Different problems of production management and control are approached in a partial way by different plant departments. The integration between the different solutions is usually very poor.
- To reduce investment costs, companies use specialized packages to approach and solve specific problems. The packages are chosen as the best in the market in each specific area. This enables the customers to have a good tool for each specific problem, but without any form of integration.

To obviate this lack of integration, IBM proposes a new approach to Plant Production Information System (PPIS), based on a three-layered architecture:

1. Management Information System (MIS)
2. Process Data Management (PDM)
3. Data Acquisition (DA)

The three-layered architecture needs an integration component to allow coherent connections among all packages and/or applications of the Integrated Plant Information System (IPIS). CIPROS facilitates and helps in managing the logical interconnection of the applications that build an IPIS on the three architectural levels.

2.1.1 CIPROS architecture

CIPROS has been designed in a client/server architecture. All the CIPROS client workstations, the CIPROS servers, and all the other application servers inside IPIS are to be connected through a plant-wide network.

The integration function between all the components of IPIS is the most important role of CIPROS. The user is enabled to access data belonging to different application subsystems by controlling and maintaining all the information inside the relational database using a standardized and user-friendly interface. Menus and specific CIPROS functions, which do not require deep knowledge of the data model structure, guide the user to navigate among the data.

Workstations are the work tools for each plant user. The client/server architecture chosen to develop CIPROS, together with common tools available in the workstation environment, gives additional value to the solution by enabling personalized analysis on data. Figure 1 shows the architecture and the platforms involved in the CIPROS solution.

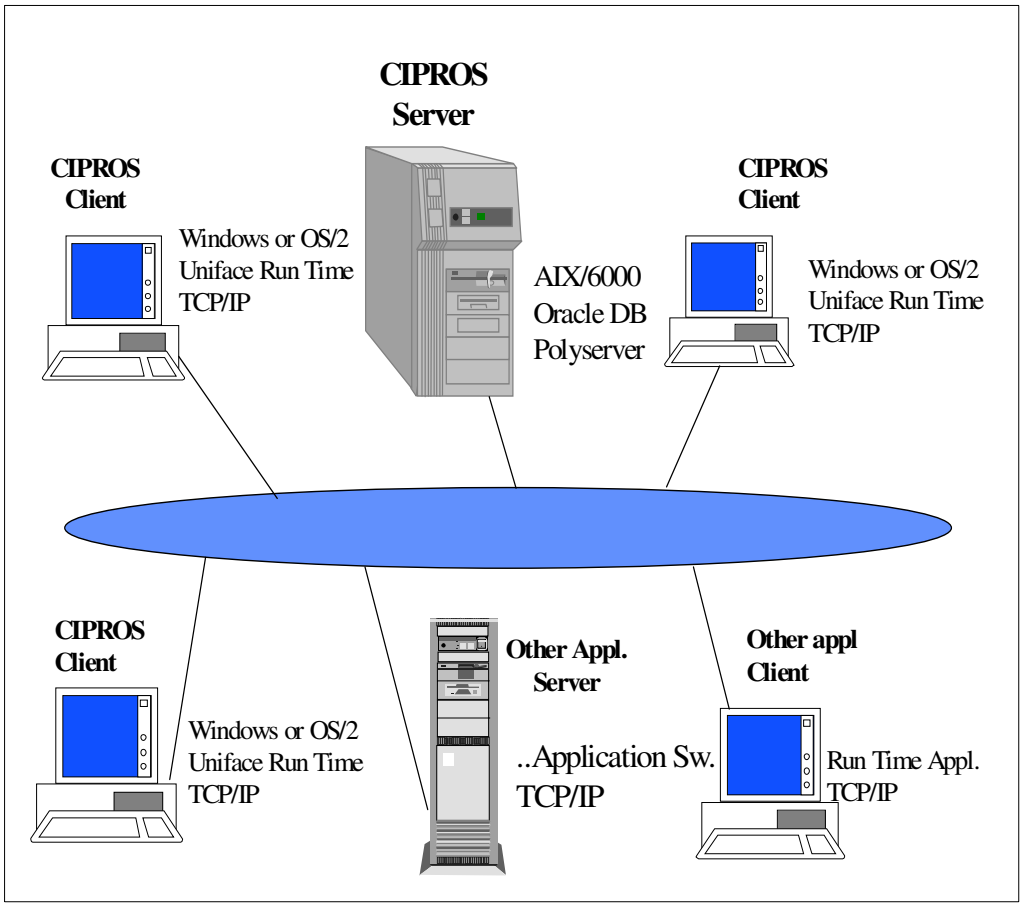


Figure 1. CIPROS client/server architecture

The actual environment used during our project is composed of the following elements:

- AIX 4.2.1 CIPROS server
- Oracle DB 7.3 (on server)
- Uniface (Compuware) Polyserver 7.2.03 (on server)
- Clients OS/2, Windows 3.11, Windows 95, Windows NT 4.0 (all with Uniface Runtime)
- TCP/IP communication protocol.
- MQSeries 5 (both on client and server)

2.1.2 CIPROS components

As shown in 2.1, “What is CIPROS?” on page 5, CIPROS is a complex environment that integrates data originated from several other applications, enabling them to exchange information in a standardized and controlled way.

Relational data model

The core component of CIPROS is a relational data model, which maps the “production scheme” of the plant.

This means that it contains a “definition” of all the facilities of the plant, in a format that is supposed to be useful to all the departments of the plant itself, for a general view of the production process and all the information related to it (measures, movements of materials, and so on).

Tables

To better understand what can be stored inside CIPROS tables, we have divided this information into the following segments:

- Reference Tables — contain definition of plant facilities, materials etc.
- Process and Laboratory Data — contains variables that come from Real Time Database packages and from Laboratory systems.
- Movement Data — contains information related to material movement between facilities of the plant and to external movements (shipments and receipts).
- Databook — contains reference data of the plant related to the ways of running of each unit in terms of yields, qualities, consumptions, capacities, and so on.
- Planning and Scheduling — contains information related to planned material productions and operations for a specific period (generally a month) and day-by-day scheduled operations for the modeled plant part.

Bridges

CIPROS provides communication services to guarantee data exchange between the different subsystems that make up the IPIS at the different architectural levels. The programs that perform all this work are called *Bridges*, so a bridge is an application that automatically transfers data to and from the CIPROS relational database. When a bridge exists on the CIPROS side, an equivalent program must exist on the side of the application that is exchanging data with CIPROS, either to extract data that has to be provided to the bridge on the CIPROS side to be inserted into the database, or to load into the application the data that has been extracted from CIPROS. A bridge is made of two parts:

- Poller: a program that extracts data from CIPROS or from an application and prepares it in a standard format
- Loader: a program that can load data into CIPROS or into an application.

The communication between the Poller and the Loader happens through exchange of files.

A third component of the bridge architecture is the Dispatcher.

The Dispatcher is a standard CIPROS tool that allows communication between the client and the server parts of a CIPROS application, like Poller and Loader of a bridge. Its main purpose is to guarantee the correct transmission of data files whatever the conditions of the communication are at the moment when the request is issued. Since the Dispatcher handles communications between two machines, it has also been enabled to send on-line requests or remote execution commands.

In the case of data file transfer, the Dispatcher has to deal with functions like:

- Communication synchronization
- Automatic restart and reconnect after a failure

- Retransmission of corrupted information
- Acknowledgment of successful transmission
- Security in data transmission
- Independence from communication protocol, hardware and software platforms.

The product chosen to build the Dispatcher logic is MQSeries. This is part of the IBM Open Blueprint and meets all the previous requirements.

MQSeries must be present as a Server on each machine where a Loader runs, and as a Client on each machine where a Poller runs; server functionality contains also a Client part.

Batch programs
CIPROS provides also several batch programs that perform calculation and reporting activities, such as:

- Tank Content/Composition application
- Daily/Monthly average and cumulated values application
- Summary Operation Data feature
- Standard Reports

Graphical User Interface
The management and query operations of the CIPROS database can be performed using the CIPROS Graphical User Interface (GUI). The CIPROS GUI is a Uniface application, based on Compuware *Uniface/Polyserver* architecture, which allows multi-platform and multi-DBMS interface of application development.

CIPROS provides a set of standard tools to allow the user to see, manage, and insert or modify data in the relational database, in a simple and guided way. These consist of user interfaces, basic database access routines and calculation functions, and some production tools as:

- Report Viewer: a tool that enables the user to connect to the relational database, in a manual or automatic way, and create in it reports produced inside or outside CIPROS.
- Log Event facility: a user interface that allows the user to see the status of all the applications in CIPROS, to correct the errors and to start a reload procedure.
- Process and Laboratory Data Analysis feature: an application that enables the user to search, retrieve, report, trend and export outside CIPROS environment, process and laboratory data related to one or more specific facilities, in a simple and driven way.
- DataBook feature: an application that manages all the data related to the runs of the plant units via user driven dialogs. It is based on a set of relational database tables that model all the parameters needed to identify the standard modes of operation of a plant unit and also to simulate (with a Simulation Interpolation function) the run of a unit if some parameters of the unit are changed.

An example of the CIPROS GUI desktop is provided in Figure 2.

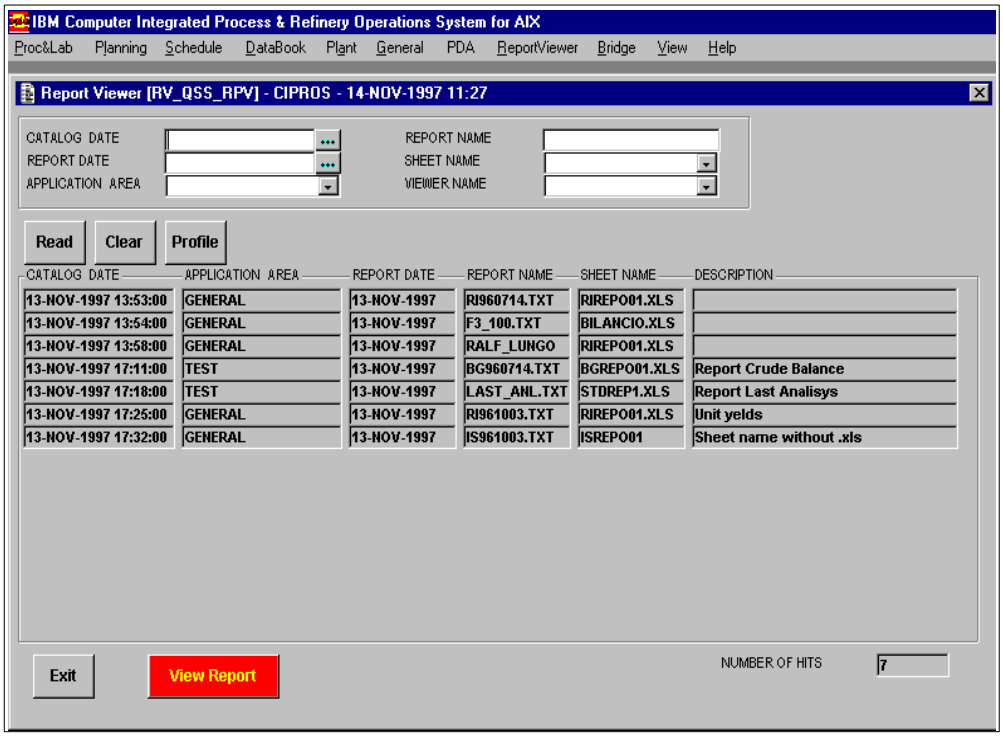


Figure 2. An example of a CIPROS GUI dialog

2.2 Scope of work for the project

As mentioned in 1.2, “Project overview” on page 2, not all the CIPROS features could be covered during the project; the chosen subjects of the conversion consist of:

- The *reference, process* and *laboratory* area of the CIPROS Database (see Figure 3) in terms of physical and logical design
- The *reference, process* and *laboratory* data
- The Real Time Database to CIPROS C-language bridge
- The Laboratory to CIPROS C-language bridge

Moreover, whenever the other related data model areas and applications include significative items, we also try to provide specific examples and migration paths for these items.

The client/server CIPROS GUI application (based on Uniface/Polyserver architecture) is not included in our sample scenario, even though it is part of the conversion project. The Compuware Uniface/Polyserver solution is designed in such a way that it should provide cross-DBMS compatibility and allow for easier porting at a minor cost, but we have been unable to confirm this.

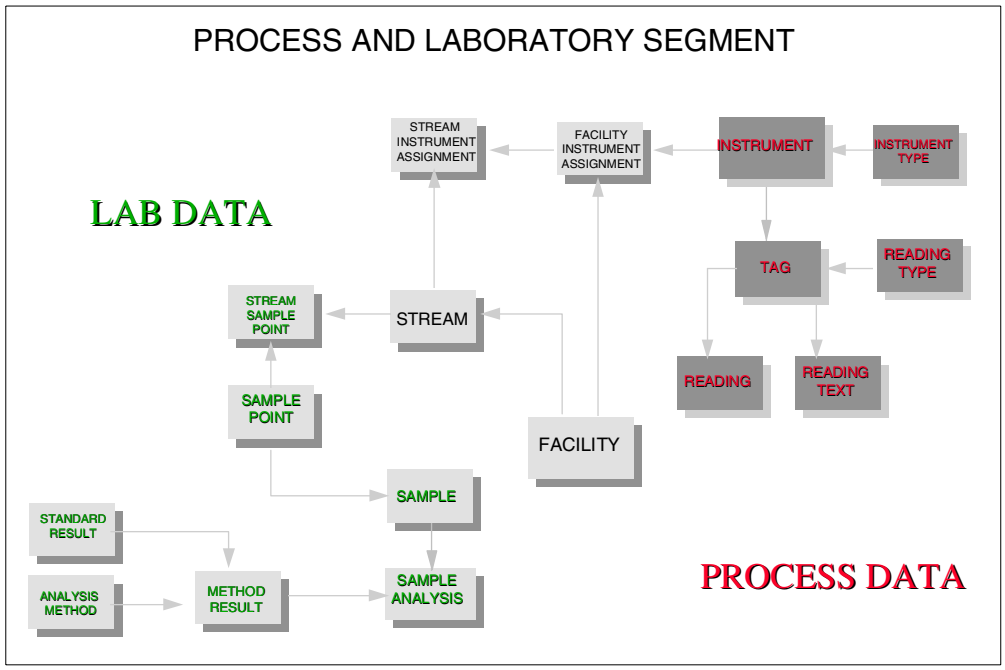


Figure 3. Process and Lab data segments of CIPROS data model

Architectural considerations:

In a real project such as ours, choosing the easiest, fastest, and least expensive paths for the implementation of the project is probably one the most important goals of the project itself.

The architectural choices (in terms of platforms, products, resources) are conditioned by the time available and economic factors.

Our architectural choices have been based on similar assumptions, besides pure technical considerations.

For these reasons we decided to use:

- C-language for OS/390 with embedded SQL, together with DB2 C precompiler for the *bridge* programs
- Uniface/Polyserver architecture (even if we did not have the possibility to include the conversion of the client/server graphical user interface in the project)

The language used and the architectural communication design are the same as the source application.

Other technical solutions were feasible and reasonable for the target environment:

- Since on the OS/390 platform skills on COBOL (or PLI) language are more consolidated in most companies, when comparing to C language, a possible alternative could have been the decision to convert the C programs into COBOL (or PLI) programs with embedded SQL.
- Moreover, the DB2 family provides the DRDA architecture for distributed environments. Using DB2 Connect on the AIX machine and accessing DB2 for OS/390 through DRDA could have been another feasible solution.

Chapter 3. Conversion process

This chapter discusses the conversion process used for the CIPROS database and application conversion from Oracle7 on AIX to DB2 UDB Server for OS/390 Version 6.

3.1 Overview

In defining the conversion process we have used the three-stage conversion methodology as defined in *Planning for Conversion to the DB2 Family: Methodology and Practice,* GG24-4445.

Figure 4 represents the three stages for the conversion:

- Stage one — defining the strategy
- Stage two — proof of concept
- Stage three — implementation and cutover

Stage one (defining the strategy) and stage two (proof of concept) are the focus of this project and redbook. Stage three (implementation and cutover), would be a future project following the completion of this redbook.

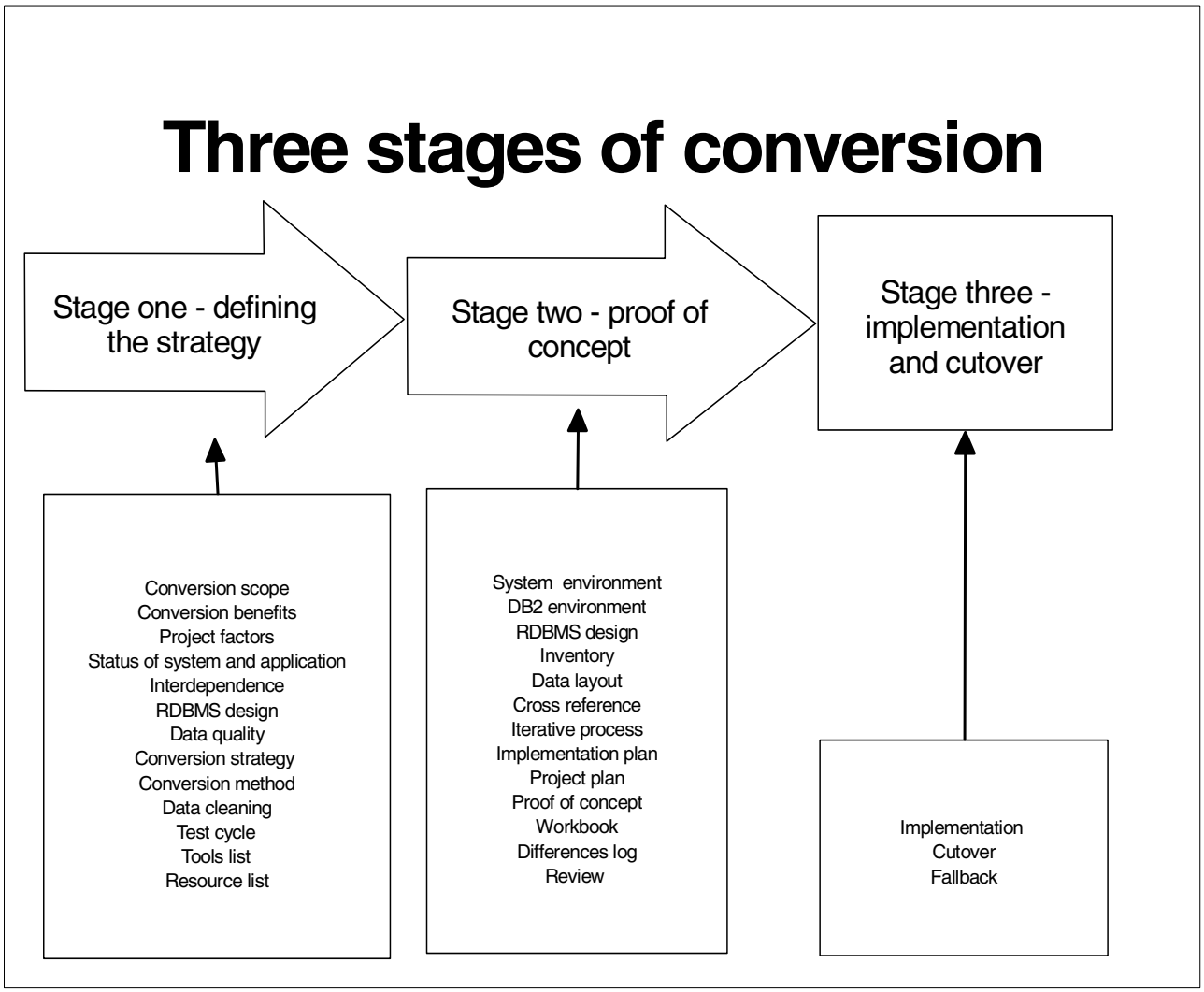


Figure 4. Three stages of conversion

3.2 Stage one — defining the strategy

Stage one (defining the strategy) asks a lot of questions and sorts opinions about the conversion. The results of these are recorded and used to set out a conversion strategy. Stage one includes the following steps:

- Survey
- Business reasons and requirements
- Portfolio analysis
- Strategy definition
- Conversion methods
- Deliverables

Figure 5 displays the inputs, outputs, and tasks for each step in stage one.

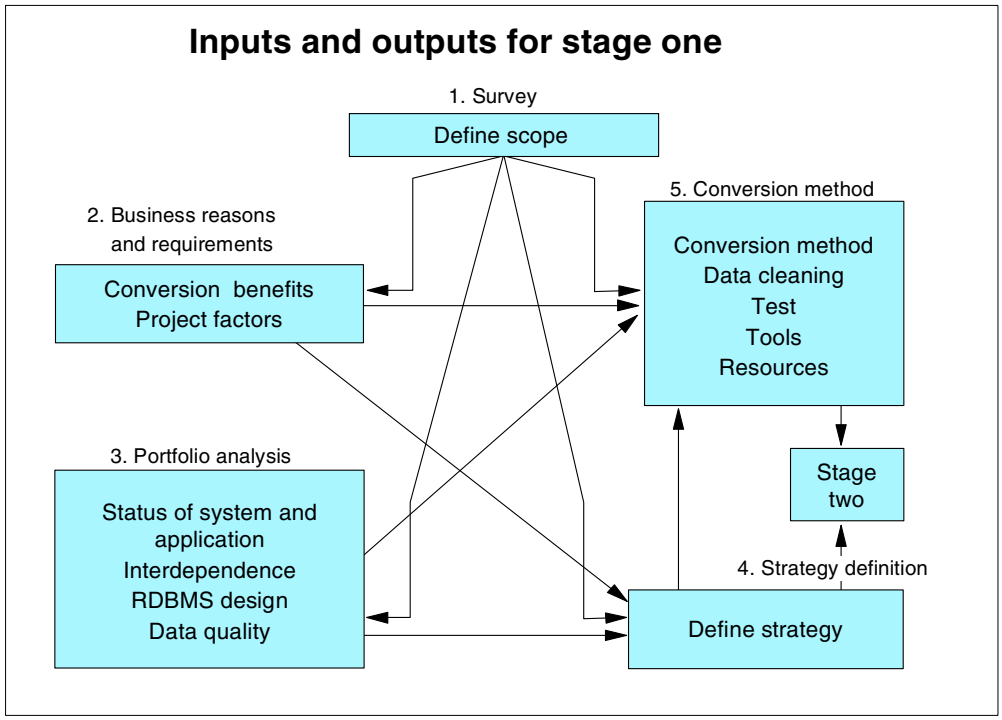


Figure 5. Inputs and outputs for stage one

3.2.1 Survey

This section discusses the objective, inputs, tasks, deliverables, and resources for the survey step of stage one (defining the strategy).

3.2.1.1 Objective

The objective of this step is to set the scope of the conversion. It defines which system or systems are candidates, the data sources, the database, and the programming languages. The survey results are used in 3.2.2, “Business reasons and requirements” on page 16, 3.2.3, “Portfolio analysis” on page 18 and 3.2.5, “Conversion methods” on page 26.

3.2.1.2 Inputs

The inputs are the application and environment areas under consideration.

3.2.1.3 Tasks

The following survey sets the boundary for the conversion proposal. The form shown in Table 1 represents a portion of our survey. Other source and target applications were considered for the conversion. The form captures the main points of the survey.

Table 1. Scope of conversion

Survey Question	Source application/area	Target application/area
Candidate	Process and Laboratory Data	Process and Laboratory Data
Data source	Real Time Database packages, Laboratory Data System and Oracle DB7.3 on AIX	Real Time Database packages, Laboratory Data System and DB2 UDB for OS/390 V6
Database	Oracle DB 7.3 on AIX	DB2 UDB for OS/390 V6
Batch/on-line	Batch via libraries of C functions, message catalog, configuration file and main programs	Batch via libraries of C functions, message catalog, configuration file, and main programs
Programming Language	ProC for AIX, Embedded Static SQL and Dynamic SQL	C for OS/390 V1.8, Embedded Static SQL and Dynamic SQL
Transaction processing	Bridges, Uniface and Polyserver 7.2.0.3 by Compuware	Bridges, Uniface and Polyserver by Compuware
Operating System	AIX 4.2.1, Windows NT 4.0	OS/390 V2.7, Windows NT 4.0

3.2.1.4 Deliverable

The deliverable is a completed statement of scope of conversion, listing the system candidate, the data sources, the database, and the programming languages which are to be considered.

The scope of our conversion is the Process and Laboratory data application segments of the CIPROS system. The data sources are the process Real Time Database package, the Laboratory Data System, and the database. The source database Oracle DB 7.3 on AIX to the target database DB2 UDB for OS/390 V6. The source ProC for AIX, embedded static SQL, and dynamic SQL languages will be converted to the C for OS/390 V1.7, embedded static SQL, and dynamic SQL.

3.2.1.5 Resources

The personnel involved with this step are:

- Project sponsor
- IT architect

3.2.2 Business reasons and requirements

This section discusses the objectives, inputs, tasks, deliverables and resources for the business reasons and requirements step of stage one (defining the strategy).

3.2.2.1 Objective

The objective of this step is to set out business reasons and requirements for the conversion. The business reasons are benefits expected from the move and the grade of importance of each. The business requirements access project factors that relate to the conversion itself. These requirements are not a reason to convert to DB2, but are project factors that represent the business requirements for the conversion. The reasons and requirements recorded now are used in 3.2.5, “Conversion methods” on page 26 to decide what type of conversion is best for a particular installation’s real needs.

3.2.2.2 Inputs

The inputs are the results of 3.2.1, “Survey” on page 15, understanding of problems, requirements, and effects of relational factors.

3.2.2.3 Tasks

Table 2 is completed to represent the business reasons why the conversion is needed. Table 3 is completed to record the business requirements of the conversion. The column headings for the tables are:

Important This is classified as one of the main reasons for the conversion. Items here must be delivered by the project to make the project a success.

Nice to have These are items that are valued and may be used later, but are not important at the moment. The project should try to incorporate these if possible.

Not important These are items that are not necessary. They do not need to be incorporated into the project.

Table 2. Business reasons for conversion

Business reason	Important	Nice to have	Not important
Improve performance		X	
Enable client/server	X		
Enable distributed data	X		
Enable new application areas	X		
Enable development on personal computers			X
Meet government regulations			X
Provide alternative solution to current database vendor	X		
Exploit existing skill base	X		
Exploit latest technology	X		

Table 3. Business requirements for conversion

Business requirements	Important	Nice to have	Not important
Budget	X		
Time	X		
Skills and resources	X		
System availability			X
Better understanding of data		X	

3.2.2.4 Deliverables

The deliverables are a list of completed business reasons for conversion and a list of completed business requirements for conversion.

The business reasons for our conversion were to provide an alternative solution to the current database vendor, exploit the existing skill base, and take advantage of latest technology. In addition, we had to maintain the client /server architecture, distributed data environment and ability to easily develop new applications. Our business requirements were to assess the project from time, skills and resources areas.

3.2.2.5 Resources

The personnel involved with the task are:

- Project sponsor
- Management
- Application and data owners
- Data consultant

3.2.3 Portfolio analysis

This section discusses the objectives, inputs, tasks, deliverables and resources for the portfolio analysis step of stage one (defining the strategy).

3.2.3.1 Objectives

The objective of this step is to understand the current state of the applications and systems, the interdependence, the DBMS design and the data quality. The information obtained in the portfolio analysis is used to make decisions in 3.2.4, “Strategy definition” on page 23 and 3.2.5, “Conversion methods” on page 26.

3.2.3.2 Inputs

The inputs are the results of 3.2.1, “Survey” on page 15, understanding of present applications, systems, data and database in Chapter 2, “Project scenario” on page 5.

3.2.3.3 Tasks

A summary of the portfolio analysis step tasks are:

1. Describe the system and applications
2. Assess interdependence
3. Assess the application
4. Assess the database
5. Assess the quality of the data
6. Portfolio analysis summary

A subset of our portfolio analysis work proceeds as follows, based on the portfolio analysis task list:

1. Describe the system and applications.

   This task describes the system, the application, and their data sources. The data sources include applications and database.

The structure of the CIPROS system is made up of a core function and a set of featured applications. The base component of CIPROS is the relational data model, valid for Oracle or DB2. The data stored inside CIPROS tables are divided into segments grouped by the different applications.The architecture of the CIPROS system allows the split of the system into functional application segments. The application of the CIPROS system we are evaluating is the Process and Laboratory Data application. The database tables for this application contain data that come from Real Time Database application packages and from Laboratory Data System. See Figure 6.

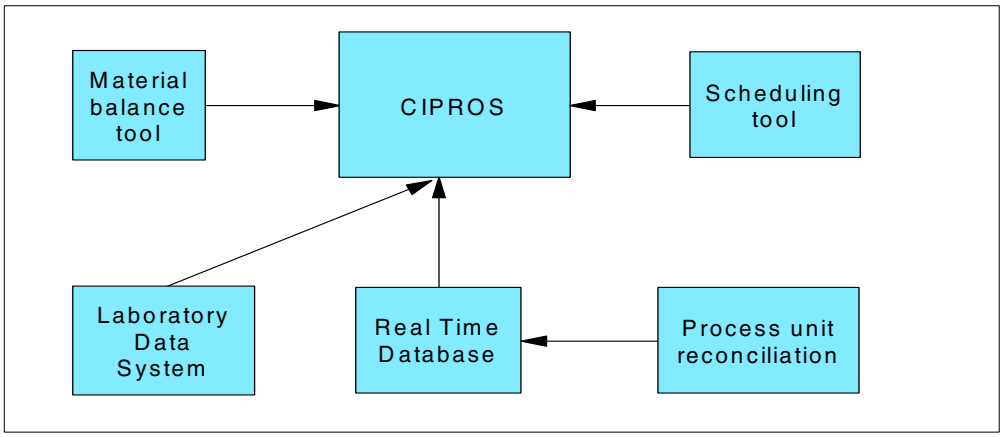


Figure 6. CIPROS architecture

2. Assess interdependence.

   This task identifies application components and the degree of system and application interdependence based on relationships between inputs, outputs and use of data. The level of interdependence will affect the choice and method for conversion because the effect on the related systems must be considered.

   The essential questions that this piece of work will answer are:
   Can the application and the data be moved on a piece-by-piece basis, and if so, what are the pieces?

   The answers are important in three areas:

   1. Pilot — *If we test using a pilot, what piece should we use?*
   2. Testing — *This piece can be tested independently.*
   3. Implementation — *If we move a piece to DB2, it works by itself, and the rest of the application still functions successfully.*

   The database Process tables contain information related to instrumentation measurements with the related timestamp. The process data is received from the Real Time Database system. An example of some of these tables are INSTRUMENT, TAG, and READING. See Figure 3 on page 10.

   The database Laboratory tables contain information related to the results of the analysis performed on a sample that is taken in the plant from a facility. The data is received from the Laboratory Data System. An example of some of these tables are SAMPLE, STANDARD RESULT and ANALYSIS METHOD. See Figure 3 on page 10.

There are several database tables that are common to all CIPROS applications. The Process and Laboratory Data application uses these common tables. An example of some of these tables are STREAM and FACILITY. See Figure 3 on page 10.

The Process and Laboratory Data application is made up of several components. See Figure 7. The main program component for the application uses various libraries based on function. The SQL library component contains .pc functions with SQL calls to the database used by both the Process main program and the Laboratory main program. The Utility library component contains .c functions that are common to all the CIPROS application main programs. Other main programs have other SQL libraries that call the database.

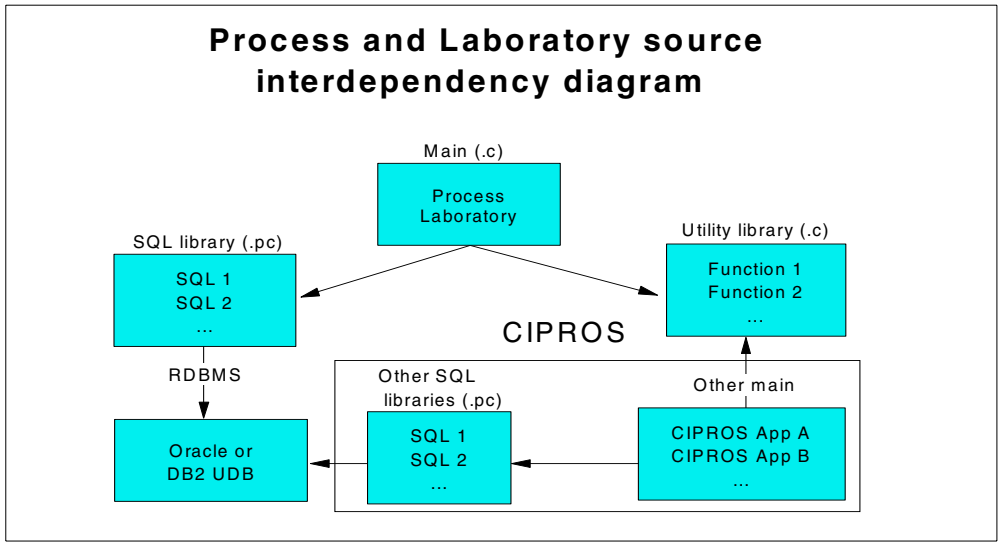


Figure 7. Process and Laboratory source interdependency diagram

3. Assess the application.

   This task uses the Application Property List to describe the properties of conversion candidate applications to assess the benefits of possible changes. The result of this assessment will be input when deciding what strategy to use. Table 4 is a subset of our Application Property List assessment. Other source applications were considered for the conversion

Table 4. Application property list

Property	Application/area
Application name	Process and Laboratory data
Source	IBM software package
Source language	ProC for AIX, embedded static SQL and dynamic SQL
Data groups used	Process, laboratory, utilities
Test data	Good and comprehensive of all functionality of conversion
Test scripts	Good and comprehensive use of application
Test system	Good

Property	Application/area
Number of batch programs for conversion	2
% batch programs used for conversion	100%
Estimate % not needed for conversion	0%
Number of on-line client connections for conversion	3
Number of programs with database calls for conversion	2
Average number of database calls per program for conversion	Process 5 and Laboratory 10
I/O modules for conversion	30%
On-line module usage for conversion	1 module used on average for a transaction
Conversational for conversion	0% of transactions are conversational
Suitability for conversion tool	Database 70%, data 60%, .pc code 60%, .c code 10%
Code consistency	Efficient, no bugs, 4GL, tool
Current data source	Oracle 7.3, Real Time Database and Laboratory Data System
Security	Database and operating system
Associated packages	Unicode (requires changes to the Uniface table names) and Polyserver

4. Assess the database.

   The database design assessment assesses the data. It shows how easy it is to map existing structures into another database system. The result of this assessment will be input when deciding what strategy to use. Table 5 is a subset of our database assessment.

Table 5. Database design assessment

Property	Source application/area
Data group name	instance=CIPROS,tablespace=CPRS_BASE, CPRS_LAB,CPRS_LOGE,CPRS_READING, CPRS_READING_IND,CPRS_PRDSCN
Location	Oracle database, instance=CIPROS
Owning user for application	CIPROS
Other users sharing applications	CPRS_VIEW
Data model	Second normal form, some third normal form

Property	Source application/area
Field definitions	CIPROS documentation provides data dictionary and field and table definitions
Data dictionary	CIPROS documentation automatically generated by CASE tool
Data dictionary data validity	The data is up-to-date
Data quality	There are no errors
Data corruption	0%
Data duplication	Some duplication, most second normal form, some third normal form
Number of entities	66 (64 tables and 2 view)
Number of fields	Min field/table = 2, max field/table = 31, average field/table =4
Number of bytes	Average daily amount of data = 10MB
Number exits and type	No

5. Assess the quality of data.

   This step assesses the accuracy of the actual data for quality. The result of this corruption assessment will be used in the strategy definition step.

   A data corruption assessment was completed prior to this project with no errors in the data. Database errors, data corruption caused by programming, and data errors in the actual data were checked.

6. Portfolio analysis summary.

   This task summarizes the findings of the Portfolio analysis step. Table 6 states the current state of the applications and systems, the interdependence, the database design, and the data quality.

Table 6. Portfolio analysis summary

Characteristic	Application/area
Application name	Process and Laboratory data
Business value	Enhanced value to the business can be provided by offering alternative vendor solutions if this application uses data that is in a DB2 database
Interdependence	The data and program components for this application are very independent, except for the use of the common Utility library
Technology level — programs	The programs are efficient, no bugs, and use a 4GL tool
Technology level — data	The data dictionary can be used to discern the meaning of the tables and columns
Data quality	Clean
Database	Oracle 7.3 for AIX
Size	The proposed area of conversion is small; less than 100 tables, less than 100 programs
Status	The application is stable and provides enhancements

3.2.3.4 Deliverables

The deliverables for the Portfolio analysis are the following items:

- Single sheet representation of each application
- System and application interdependency analysis
- Application characteristics assessment
- Application property list assessment
- Database design assessment
- Data quality assessment
- Portfolio analysis summary

3.2.3.5 Resources

The personnel for the Portfolio analysis are the following:

- Data manager and senior database administrators
- Applications development manager and senior professionals
- Operations manager and senior professionals
- User manager and senior users
- Security personnel

3.2.4 Strategy definition

This section discusses the objectives, inputs, tasks, deliverables, and resources for the strategy definition step of stage one (defining the strategy).

3.2.4.1 Objective

The objective of this step is to define the strategy for the proposed conversion based on the knowledge of the source applications and system. First, the overall strategy is defined, then the conversion starting point is stated. The result of the strategy definition will be used in 3.2.5, “Conversion methods” on page 26.

3.2.4.2 Inputs

The inputs for this step are the deliverables from the 3.2.1, “Survey” on page 15, 3.2.2, “Business reasons and requirements” on page 16 and 3.2.3, “Portfolio analysis” on page 18.

3.2.4.3 Coexistence

Where a piece of the application is to be moved separately, the question of how the two parts of the total application will function must be answered.

The alternatives available are:

- Single phase — No coexistence is needed. Everything can and will be moved in a single phase.
- Complete separation — Parts are really separated and can be moved individually.
- Batch connection only — Batch programs can be changed to read one part and write to a file for use by another part.
- Duplicate data — Some data is duplicated in both old and new DBMS, it is updated by periodic batch or by continuous propagation. This is suitable for non-volatile data in read-only for one application piece.

- Calls to two DBMSs — Some applications might retain a few calls to the old DBMS. This might depend on the support for DRDA, or other techniques, and certainly has performance implications. It also means that some programs might need to be changed twice.

In our case we assessed the application and decided that coexistence was not necessary for the selected subset as we could move in a single phase.

3.2.4.4 Tasks

A summary of the strategy definition step tasks are:

1. Decide on the overall strategy.
2. Choose a starting point.

A subset of our strategy definition step process proceeds as follows:

1. Decide on the overall strategy.

   This task determines the strategy for the way forward in a conversion. The result of this step will be a completed strategy statement used for the basis of the conversion. Table 7 is a subset of our assessment.

Table 7. Conversion strategy statement

Decision Area	Comments
General strategy	Piece by piece
Process	A definable application in the system and its data will be converted from the source system and application to the target source system and application. No enhancements are made to the application during the conversion, and future enhancements are made in DB2.
Source system	Oracle 7.3 and CIPROS
Target system	DB2 UDB for OS/390 V6
Attitude to outside assistance and tools	Use outside assistance and tools where needed
Confirm priority order	1. Quality of result 2. Speed 3. Least risk 4. Least cost

2. Choose a starting point.

   This task determines which application in the system will be used for the conversion pilot. The result of this step is a completed starting point statement.

   Table 8 is a subset of our assessment. This assessment compared applications in the system by weighted factors. The application with the highest score indicates which application to convert. The application that received the highest total score (95) in our assessment was the Process and Laboratory Data application.

Table 8. Application comparison by weighted factor

Factor for piece by piece strategy	Weight	Process and Laboratory Application/Area (1-low through 5-high)
Enhanced business value	X1.5	5
Interdependence	X1.5	5
Technology level	X1	4
Size	X1	5
Total Score	**X5**	**95**

Once the application in the system that will be used in the conversion is selected, a starting point for the conversion can be defined. Table 9 is a summary of our starting point for the conversion. This table records the starting point for the application and the conversion pilot.

Table 9. Choice of conversion starting point application and pilot

Question	Choice
Which system will be converted?	CIPROS and Oracle 7.3 for AIX
Which application will be the conversion starting point?	Process and Laboratory data
What conversion strategy will be used?	Piece by piece
What about reporting and reporting programs?	Not applicable
Is this a piece of a large application?	No
How will coexistence be handled?	Not applicable
What will be used as a pilot?	Process and Laboratory data application in the CIPROS and Oracle 7.3 AIX system will be converted to the Process and Laboratory Data application in the CIPROS and DB2 UDB for OS/390 V6 system.
Any special factor for security?	No

3.2.4.5 Deliverables

The deliverables for the strategy definition are the following items:

- Overall conversion strategy statement
- Choice of starting point application and pilot

3.2.4.6 Resources

The personnel for the strategy definition are the following:

- Data manager and senior database administrators
- Application development manager and senior professionals
- Operations manager and senior professionals
- User manager and senior users
- Security administrator

3.2.5 Conversion methods

This section discusses the objectives, inputs, tasks, deliverables and resources for the conversion methods step of stage one (defining the strategy).

3.2.5.1 Objective

The objective of this step is to decide which conversion method to use for the pilot. Based on the selection of the conversion method, data cleaning, testing cycle, tools and resources will be determined. The result of the conversion method step will be used in section 3.3, “Stage two — proof of concept” on page 32.

3.2.5.2 Inputs

The inputs for this step are the results for the survey, 3.2.1, “Survey” on page 15, 3.2.2, “Business reasons and requirements” on page 16, 3.2.3, “Portfolio analysis” on page 18 and 3.2.4, “Strategy definition” on page 23.

3.2.5.3 Tasks

A summary of the conversion methods step tasks are:

1. Evaluate different conversion methods
2. Evaluate source and target database features
3. Select a conversion method
4. Select a conversion end point
5. Decide about data cleaning
6. Define a test cycle
7. Select tools
8. List resources
9. Conversion summary

A subset of our process for the Conversion method step proceeds as follows:

1. Evaluate different conversion methods.

   This task evaluates the different conversion methods and their advantages and disadvantages. The result of this step will be an understanding of all the conversion methods. See Table 10 for the results of our evaluation.

Table 10. Conversion method overview

Conversion Method	Description	Advantages	Disadvantages
Translation	*Data layout 'as is' to DB2 *Program calls changed 1:1	*Easiest method *Easy for tools	*Few advantages of DB2 *Design inflexible for future
Transparency	Module written to - Intercept calls to old database - Translate to SQL - Route to new Database	*Data may be restructured *Tools can be used for data *Applications can be rewritten later * Low risk	*Performance a problem * Module difficult to write *Application maintenance increased *Conversion takes much longer
Re-engineering	Make changes to take advantage of new database - Time and data - Numbers - Sequence	*Obtain advantages of DB2 *Allows limited redesign	*Longer time *Tools need some intervention
Reverse engineering	*Capture old design into tools *Reverse into -Data model -Process model *Generate new DB2 design *Generate new application	*Design may be optimized for performance *Later enhancements easier *Possible to remove redundant code	*Tools do not handle all situations *Longer time
Redevelop	*Check requirements with users *Add enhancements *Start from scratch	*Tools available *Known process *Proper documentation easy *Future maintenance easy	*Much longer development time *Less efficient code from tools * Existing investment lost
Corporate model	*Model business at high level *Model part of business *Get new application working *Gradually add other parts	*Business modeled as whole *Less redundant code *Case tools available	*Significant time before benefits *Large up front investment *Existing investment lost *Need to manage changes

2. Evaluate source and target database features.

   This task evaluates the incompatibilities between the source database features and the target database features. The result of this step will be a list of items in the source database solution that must be redesigned for the use with the target database solution. Table 11 is a subset of the incompatibilities between Oracle 7.3 for AIX and DB2 forOS/390 V6 features for our project.

Table 11. Oracle and DB2 feature incompatibilities

Feature	Oracle 7.3 for AIX	DB2 UDB for OS/390 V6
Data types		
	DATE	DATE or TIMESTAMP
	NUMBER	NUMBER, INTEGER or FLOAT
Programming	Use of C pointers for host variables	No use of C pointers for host variables
Error handling	Use of C system calls to UNIX operating system for message catalog	C system calls cannot use similar calls to MVS system for message catalog
Database	Table, foreign key and index names are longer than 18 characters	Table, foreign key and index names are less than or equal to 18 characters

3. Select a conversion method.

   This task selects the conversion method for the pilot. The information gathered in Table 10 on page 27 and Table 11 on page 28 are used as input to this task.

   We decided to use the translation conversion method for compatible features of Oracle 7.3 for AIX and DB2 UDB for OS/390 V6 and the re-engineering conversion method for incompatible features of Oracle 7.3 for AIX and DB2 UDB for OS/390 V6.

4. Select a conversion end point.

   This task defines the end point of the pilot conversion project. The result of this task will be a list check points and end points for the project. Figure 8 is a summary of the check points and end point for our project

5. Decide about data cleaning.

   This task defines how the data will be cleaned for the conversion. The data for our project had been cleaned prior to the start of our project, so it was not necessary for us to clean the data.

   If data cleaning is necessary for a project, three areas must be considered:

Validation Checking and testing are done for incorrect or corrupt data.

Corrections Once a corruption has been identified, it must be corrected.

Audit-trail Any changes are recorded on a searchable audit trail.

.

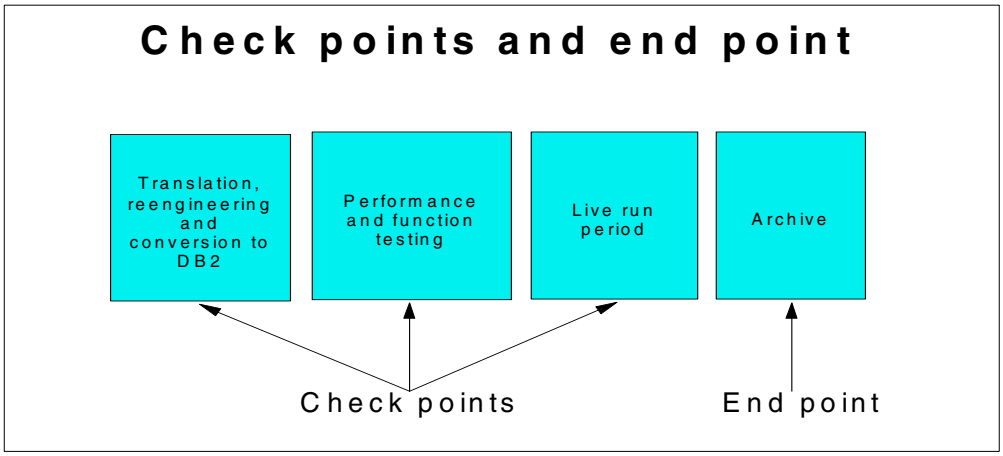


Figure 8. Project check points and end point

6. Define a testing cycle.

   This task defines the testing cycle for the conversion project. Figure 9 is a diagram of our testing cycle.

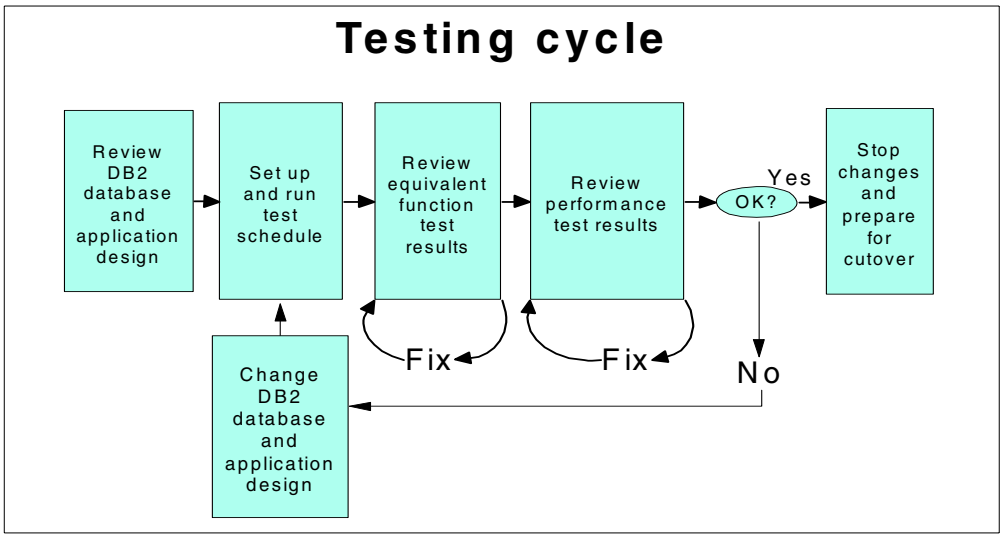


Figure 9. Testing cycle

7. Select tools.

   This task evaluates the use of tools during the project. No tool can provide 100% conversion without manual intervention but they do reduce the amount of that intervention. Table 12 is a summary of our very reductive analysis and even smaller adoption of tools for this project.

Table 12. Tools requirements and investigation list

Tool area	Description	Requirement	Candidate
Data analysis	Pictorially represent data entities and relationships	Not applicable	Bachman, Cheyenne, ADW
DBMS design and conversion	Read old DBMS, generate entity model, create new model for new DBMS	Automate DDL conversion	Implemented through scripts
Source program conversion	Convert source code from code using old DBMS to code using SQL for DB2	Not applicable	Mantech
Propagation	Propagate asynchronous and synchronous updates from old DBMS to DB2	Not applicable	DPROP
Unload and reload	Move data from source database to target database	Load into DB2	DB2 LOAD utility, Data Joiner
On-line identical function testing	Test that new code still provides same function as old code	Compare results of old and new application	Executed through sample transactions and queries
Full testing analysis	Make sure all parts of the program are tested	Not applicable	Not applicable
Batch output comparisons	Compare batch outputs run with old and new code	Not applicable	Not applicable
Stress testing of system	Run test transactions through new system at rate equal to or greater than the production service	Not applicable	Not applicable
Real-time performance monitoring	On-line performance monitor	Verify that performance are within the expected ranges	DB2 PM

8. List resources.

   This task sets up a list of resources that will be needed for the project. Table 13 is a summary of the resources for this project.

Table 13. Resource list

Function	Part time	Full time	Description	Person
Sponsor	X		Committed executive sponsor who wants the project to succeed	
Project manager		X	Person to plan, track progress, resources and interfaces between groups	
DB2 database administrator	X		Person to resolve DB2 issues including performance tuning	
Oracle database administrator	X		Person to resolve Oracle issues	
Business analyst	X		Assesses the impact of the conversion on the business	

Function	Part time	Full time	Description	Person
Conversion consultant	X		Understands source and target environment, ensures right questions are asked	
Source application programmers	X		Understands the old application, language and SQL	
Target application programmers		X	Understands the target language and SQL	
System programmer	X		Installs DB2, sets up OS/390 and system standards	
Test system specialist	X		Set up and run testing system	
Operations analyst	X		Convert utilities, back up system	
Change control specialist	X		Controls design changes	
Performance specialist	X		Understands performance tools	

9. Conversion summary.

This task summarizes the conversion method for the project. Table 14 is a summary of our conversion decisions.

Table 14. Project conversion summary

Item	Comment
System	CIPROS
Application	Process and Laboratory data
Source database	Oracle 7.3 for AIX
Target database	DB2 UDB for OS/390 V6
Source operating system	AIX 4.2.1, Windows NT 4.0
Target operating system	OS/390 V1.7, Windows NT 4.0
Conversion method	* Translation for compatible features of Oracle and DB2 * Re-engineering for incompatible features of Oracle and DB2
Conversion check points	Translation, re-engineering, conversion to DB2, performance, function testing, live run period
Conversion end point	Archive
Data cleansing requirements	Validation, corrections and audit trail completed prior to this project
Test cycle	The test cycle will review the DB2 database and application design, set up and run the test schedule, review equivalent function test results, review performance test results, change DB2 database and application design where necessary, stop changes and prepare for cutover
Tools	DB2 Load utility
Resources	Sponsor, project manager, DB administrators, business analyst, conversion consultant, programmers, test specialist, operations analyst, change control specialist, performance specialist

3.2.5.4 Deliverables

The deliverables for the conversion method are the following items:

- Selection of conversion method
- Completed data cleaning
- Completed test cycle
- Completed tools list
- Completed resource list
- Conversion summary

3.2.5.5 Resources

The personnel for the strategy definition are the following:

- Data manager and senior database administrators
- Application development manger and senior professionals
- Operations manager and senior professionals
- User manager and senior users

3.2.6 Defining the strategy deliverables

The deliverables of stage one (defining the strategy) are:

- Conversion scope
- Conversion benefits
- Project factors
- Status of the system and application
- Interdependence
- Database design
- Data quality
- Conversion strategy
- Conversion method
- Data cleaning
- Test cycle
- Tools list
- Resource list

3.3 Stage two — proof of concept

Stage two (proof of concept) validates the opinions recorded in stage one (defining the strategy) and thinks through the details of how the conversion project will actually function. The results of this stage are the conversion of the first application as a pilot and a tested proof of concept. This tested proof of concept is used as input for stage three (implementation and cutover).

The activities in stage two are covered in detail in the following sections. Input from stage one is used in each activity. The main activities during stage two are shown in Figure 10. Each discrete activity needs to take into account other activities. The main activities are:

- Once-only tasks
- Iterative tasks
- Implementation plans
- Proof of concept

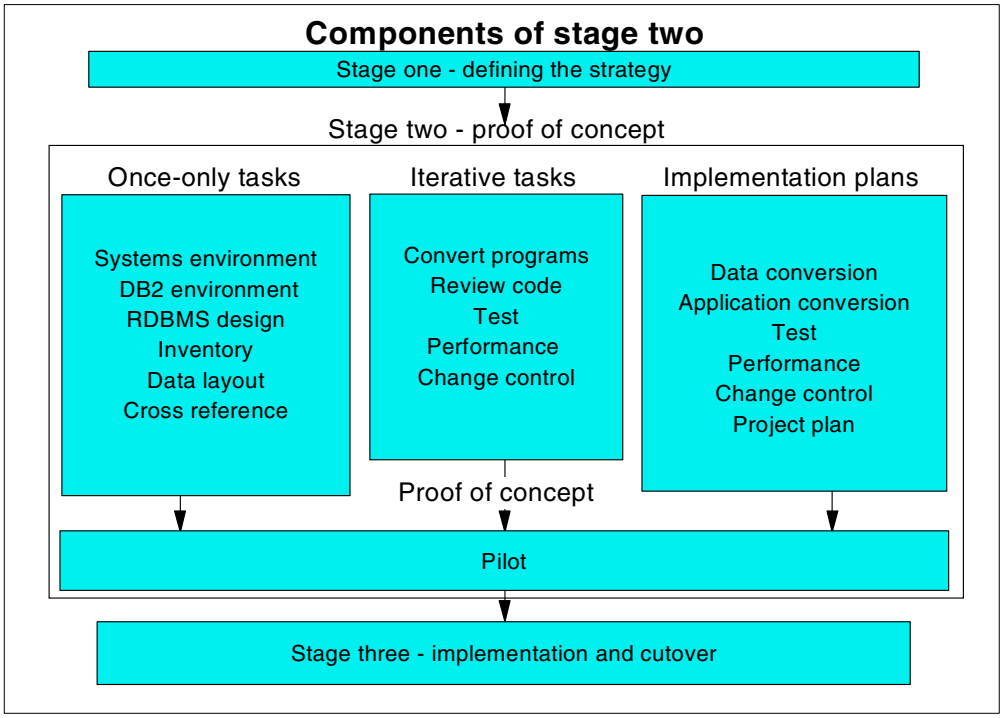


Figure 10. Components of stage two

3.3.1 Once-only tasks

There are a number of tasks that have to be done only once during stage two in order for the conversion to function. The task activities are based on input from section 3.2, “Stage one — defining the strategy” on page 14. The once-only tasks for a conversion include:

- Systems environment
- DB2 environment
- Database design
- Inventory
- Data layout
- Cross reference

3.3.1.1 Systems environment

Both the source and target system environments will need to be configured and set up for the conversion. Chapter 4, “System environment” on page 45 discusses the following system environment tasks, tools and resources for our project:

- Configuration
- Security
- Backup and recovery
- Storage management
- Communication
- Compiler
- AIX to OS/390 terminology mapping
- OS/390 tools and tips

The required tools for the system environment are:

- AIX C, awk, sed
- FTP for AIX
- FTP for OS/390
- DB2 UDB for OS/390 V6 utility programs
- JCL

The resources for the system environment are:

- UNIX systems
- UNIX programmers
- Oracle DBA
- OS/390 systems
- S/390 storage administrator
- S/390 C Language programmers

3.3.1.2 DB2 environment

A separate DB2 environment is best for a conversion and leads to the least conflicts with other work. If DB2 is new to the installation, then the whole range of systems, including systems programming, program test, system test, and production will need to be set up. Section 5.1, “DB2 environment” on page 61 discusses the following DB2 environment tasks, tools and resources for our project.

- Installation
- Configuration
- Security
- Backup and recovery
- Storage management
- Communications

The required tools for the DB2 environment are:

- DB2 UDB for OS/390 V6 utility programs
- DB2 interactive functions
- DB2 sample programs

The resources for the DB2 environment are:

- DB2 system programmers
- DB2 application programmers
- DB2 DBA

3.3.1.3 Database design

Good DB2 database design is important for assuring identical function, good performance, future maintenance and good operations. Much of what goes on here is standard DB2 design. Section 5.2, “Database design” on page 64 discusses in more detail the following database design tasks, tools and resources for our project.

- Logical design
- Physical design

The required tools for system environment are:

- DB2 UDB for OS/390 V6 utility programs
- JCL

The resources for system environment are:

- Oracle DBA
- DB2 DBA
- OS/390 systems and support
- DB2 programmers

3.3.1.4 Inventory

The inventory is an extremely important component of the conversion. Without a comprehensive inventory of both the programs and the data sources, cross-related to the new system, it is impossible either to perform a reliable conversion or to make detailed plans and estimates for the rest of the project. The inventory is related to:

- Source programs
- Source data source
- Target programs
- Target data source

The required tools for inventory are:

- DB2 UDB for OS/390 V6 utilities

The resources for inventory are:

- Oracle DBA
- DB2 DBA

- Oracle programmers
- DB2 programmers

3.3.1.5 Data layout

The source data source must be analyzed. Every component must be placed in the inventory with a cross reference to the corresponding position in the new database. In 5.3, “Data layout” on page 105 we discuss the following data layout tasks, tools and resources for our project:

- Source tables
- Source columns
- Target tables
- Target columns

The required tools for data layout are:

- DB2 UDB for OS/390 V6 utility programs

The resources for data layout are:

- Oracle DBA
- DB2 DBA
- Oracle programmers
- DB2 programmers

3.3.1.6 Cross reference

A cross reference of fields to source tables and columns to target tables and columns needs to be built so that any field reference can be changed to the equivalent in DB2. Input for the cross reference is from the inventory task 3.3.1.4, “Inventory” on page 35 and data layout tasks 3.3.1.5, “Data layout” on page 36. Section 5.4, “Cross reference” on page 107 discusses the cross reference we used for our project.

3.3.2 Iterative tasks

When the once-only tasks are complete, the iterative tasks can be begin. The iterative tasks take the application programs through all the steps needed until they are ready for cutover. Section 7.1, “Proof of concept iterative process” on page 141 demonstrates the use of this process during our project. The iterative tasks, tools and resources for our project are.

- Convert application programs
- Review program code
- Run tests
- Performance tuning
- Change control

The required tools for the iterative tasks are:

- DB2 UDB for OS/390 V6 utility programs

The resources for data layout are:

- Oracle DBA
- DB2 DBA
- Oracle programmers
- DB2 programmers

3.3.3 Implementation plans

The implementation plans are used as input to an overall project plan. The plans are based on input from section 3.2, “Stage one — defining the strategy” on page 14. The implementation plans for a conversion include:

- Data conversion plan
- Application conversion plan
- Test plan
- Performance plan
- Change control plan

A project plan and review plan for the proof of concept is developed based on the individual implementation plans.

3.3.3.1 Data conversion plan

The data will need to be unloaded from the source database and reloaded into the target database. The data conversion plan will detail how this will take place, with what tools and resources. See Chapter 6, “Data conversion” on page 109 for details.

The tasks for our data conversion plan are:

- Clean data
- Unload data from Oracle
- Create programs to prepare data for file transfer and DB2 format
- Reformat data for DB2
- Create partitioned data sets on OS/390 for the data
- Transfer data from the AIX system to the OS/390 system
- Test data for correct format
- Load data into DB2

The required tools for data conversion are:

- AIX C, awk, sed
- FTP for AIX
- FTP for OS/390
- DB2 UDB for OS/390 V6 utility programs
- JCL

The resources for data conversion are:

- UNIX Programmers

- UNIX systems
- Oracle DBA
- DB2 DBA
- OS/390 systems
- DB2 programmers
- Storage administrators

3.3.3.2 Application conversion plan

The application conversion plan will say how the programs are to be converted. The application conversion plan will detail how this will take place, with what tools and resources. See Chapter 7, “Application conversion” on page 141 for details.

The tasks for our application conversion plan are:

- Proof of concept iterative process
- Programs for pilot
- Program redesign
- Program preparation
- Program conversion
- Program testing cycle
- Change control
- Program status

The required tools for application conversion are:

- AIX C
- FTP for AIX
- FTP for OS/390

The resources for application conversion are:

- UNIX Programmers
- UNIX systems
- Oracle DBA
- DB2 DBA
- OS/390 systems
- DB2 programmers

3.3.3.3 Test plan

Testing is a very important area because over half of all conversion activity, and perhaps as much as 80%, will be testing, changing, and then retesting. The test plan will detail how this will take place, with what tools and resources. See Chapter 8, “Testing, change control, and tuning” on page 183 for details.

The tasks for our test plan are:

- Function
- Unit

- System
- User acceptance

The tools we used for testing are:

- AIX C, awk, sed
- FTP for AIX
- FTP for OS/390
- DB2 UDB for OS/390 V6 utility programs
- JCL

The resources for testing are:

- UNIX Programmers
- UNIX systems
- Oracle DBA
- DB2 DBA
- OS/390 systems
- DB2 programmers

3.3.3.4 Performance plan

The application and system must be performance tuned during the conversion. The performance plan will detail how this will take place, with what tools and resources. See Chapter 9, “Performance tuning” on page 203 for more details.

The tasks for our performance plan are:

- Database configuration
- System configuration

The required tools for performance tuning are:

- Explain
- DB2 PM

The resources for performance tuning are:

- UNIX systems
- UNIX programmers
- Oracle DBA
- OS/390 systems
- S/390 storage administrator
- DB2 DBA

3.3.3.5 Change control plan

The way that changes are made is crucial. For example, if the DBMS design is changed over the conversion, how does this affect data and programs? The change control plan is a process which makes sure that changes to solve one problem do not cause problems with other areas. The change control plan will

detail how this will take place, with what tools and resources. See Chapter 10, “Change control” on page 205 for more detail.

The tasks for our change control plan are:

- Document change request
- Review change
- Review effect of change
- Approval of change

The required tools for change control are:

- Current tools or procedures adopted in your enterprise (like SMP/E for OS/390 systems, SCLM for applications)

The resources for change control are:

- Team leader
- Project leader

3.3.4 Proof of concept

The pilot for proof of concept is performed after the project plan is complete. A project review is performed at the end of the pilot.

3.3.4.1 Project plan

A project plan is developed for the proof of concept pilot based on the individual implementation plans, once-only tasks, and iterative tasks. The whole project needs to be controlled by a project manager. This person will keep track of all the activities needed to ensure the success of the project.

Two items worthy of special mention are the workbook and the problem and differences log kept during the project.

Workbook	A manual that states SQL standards, conversion processes, programming standards, testing, and tools
Problem and differences log	A log of all the problems and their resolutions

The project plan includes tasks, resources and estimate. Table 15 is a subset of our project plan. It represents the general tasks our project.

Table 15. Proof of concept conversion plan

Task	Resource	Estimate
Once-only tasks		
Systems environment		X
DB2 environment		X
Database design		X
Inventory		X
Data layout		X
Cross reference		X
Iterative tasks		
Convert programs		X
Review code		X
Test		X
Performance		X
Change control		X
General		
Data conversion		X
Application conversion		X

3.3.4.2 Review

When the pilot programs are capable of running well with DB2, the equivalent tests have been passed, and the performance is acceptable, then there should be a thorough review of the project to date and the plans for the overall conversion.

The reviewers should be people who have skills in all the main areas including project management, the data source, the source system, the source application, and especially DB2. The objectives of the review are:

- That the project is on course and the issues well understood
- That the project will deliver the required system
- That the final system is likely to deliver the required performance, integrity and benefits required
- To recommend changes to the plans in the light of the results of the pilot

3.3.4.3 Tested proof of concept

After the successful pilot and review, the tested proof of concept will be a fact. Adjustments to the various conversion plans will be needed for section 3.4, “Stage three — implementation and cutover” on page 43 in the light of the pilot and the review. Figure 11 represents a summary of the steps for the proof of concept. These steps are discussed in stage two (proof of concept).

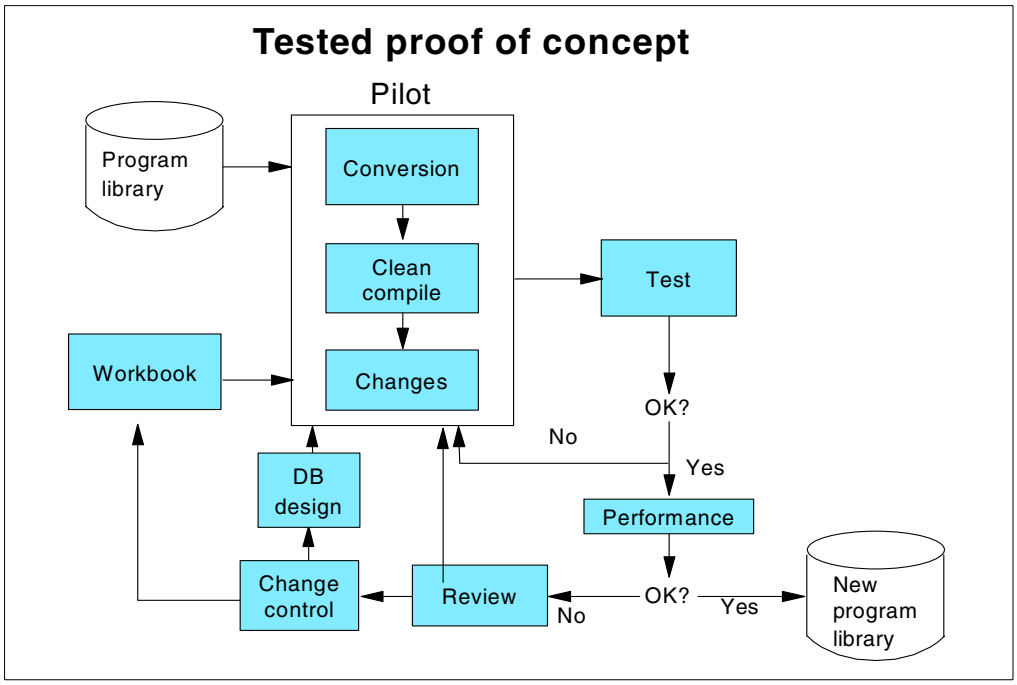


Figure 11. Tested proof of concept

3.3.5 Deliverables

The deliverables for stage two (proof of concept) are the following items:

- Target system environment
- DB2 environment
- Database design
- Inventory
- Data layout
- Cross reference
- Iterative process
- Implementation plan
- Project plan
- Review
- Proof of concept
- Workbook
- Differences log

3.3.6 Personnel

The personnel for stage two (proof of concept) are the following:

- Data manager and senior database administrators
- Application development manager and senior professionals
- Operations manager and senior professionals

3.4 Stage three — implementation and cutover

Stage three (implementation and cutover) involves taking the lessons learned during the pilot, and using them to revise the plans worked out during stage two (proof of concept), then roll the rest of the programs through the conversion and test process. There will be problems along the way, but having built on a sound and practical foundation will ensure that the project meets its objectives and is on time and within budget. The end point for our redbook project is stage two, however. Stage three would be the next project for a real user environment.

Chapter 4. System environment

The setup for the system environment is a once-only task, as discussed in 3.3.1.1, “Systems environment” on page 34. This setup includes the following activities:

- Configuration
- Security
- Backup and recover
- Partitioning of data
- Communication
- Compiler
- AIX to OS/390 terminology mapping

In this chapter we describe the source and the target system environments. While the source system environment is normally dedicated and directly dependent in terms of definition upon the Oracle subsystem and its usage, the reader has to be aware that the DB2 subsystem is only one of the several subsystems that concur in defining the requirements and the sizing of the target S/390 environment. This explains why several database related functions are described in this chapter for the source environment, while for the target system environment, we only provide a very general description of the existing system. The target database environment is described in more detail in Chapter 5, “Database conversion” on page 61.

A very simple description of the OS/390 tools and functions utilized, and some tips for the non-initiated, are reported in Appendix D, “OS/390 TSO tools and tips” on page 257.

4.1 The source system environment

In this section we describe the project environment of the CIPROS application. It contains a description of the physical definitions of the source database machine (Oracle 7.3 for AIX) and its logical configurations (users, groups, communication, logical volumes, and general settings).

4.1.1 System configuration and physical design consideration

The Oracle 7.3 Server has been installed on an RS/6000 machine, 520H Power Station, running the AIX 4.2.1 operating system. Its hostname is BALTIC. We have installed and configured the operating system and the network connection in order to be able to reach the server machine from any of our client Windows NT machines. For the installation and configuration of the AIX operating system, please refer to the *AIX Version 4.2 Installation Guide*, SC23-1924.

You can check the operating system level with the following command, issued by an AIX shell:

```
oslevel
```

The current operating system level is then prompted (4.2.1.0, in our case).

The communication setup is described in 6.3, “File transfer and format programs” on page 116. Refer also to *AIX Version 4 System User’s Guide: Communications and Networks*, SC23-2545.

The machine we used for our project has six 400 MB SCSI internal disks, all belonging to the *rootvg* volume group. Each physical partition (PP) of rootvg volume group is 4 MB.

You can check the configured disks and the volume group they belong to with the following command:

```
lscfg | grep hdisk; lspv
```

Figure 12 shows the results:

```
  + hdisk0            00-07-00-0,0      400 MB SCSI Disk Drive
  + hdisk1            00-07-00-1,0      400 MB SCSI Disk Drive
  + hdisk2            00-07-00-2,0      400 MB SCSI Disk Drive
  + hdisk3            00-07-00-3,0      400 MB SCSI Disk Drive
  + hdisk4            00-08-00-0,0      400 MB SCSI Disk Drive
  + hdisk5            00-08-00-1,0      400 MB SCSI Disk Drive
  hdisk0          0000270700106442    rootvg
  hdisk1          0000270700106cd9    rootvg
  hdisk2          00002707001075a7    rootvg
  hdisk3          0000270700107e42    rootvg
  hdisk4          0000270700030fcc    rootvg
  hdisk5          00002707001087c9    rootvg
```

Figure 12. Output of lscfg and lspv commands

AIX JFS Manager is the AIX O.S. facility to manage the AIX file systems, in terms of files and directories.

Oracle database implements database managed (DMS) containers for the table spaces. In Oracle, they are generally called *data files* and can be defined as JFS files. It is possible to assign to the same table space, one or more data files, that is, one or more JFS files.

AIX Logical Volume Manager (LVM) provides the possibility to define logical volumes on physical disk volumes (*raw devices*, in classical database definition).

Raw devices can be used as database containers, too. In the current naming convention, usually the name *data files* is referred both to JFS files and to raw logical volumes.

Figure 13 shows the relationship between a logical volume and its components. In this example, the logical volume is made up of two mirrored logical partitions, that means four physical partitions on two different disks. The logical volume is the table space container for an Oracle table space.

Note that in the case of a table space on a raw device, the logical volume must be created in advance (for the JFS files, the file is automatically created by the RDBMS).

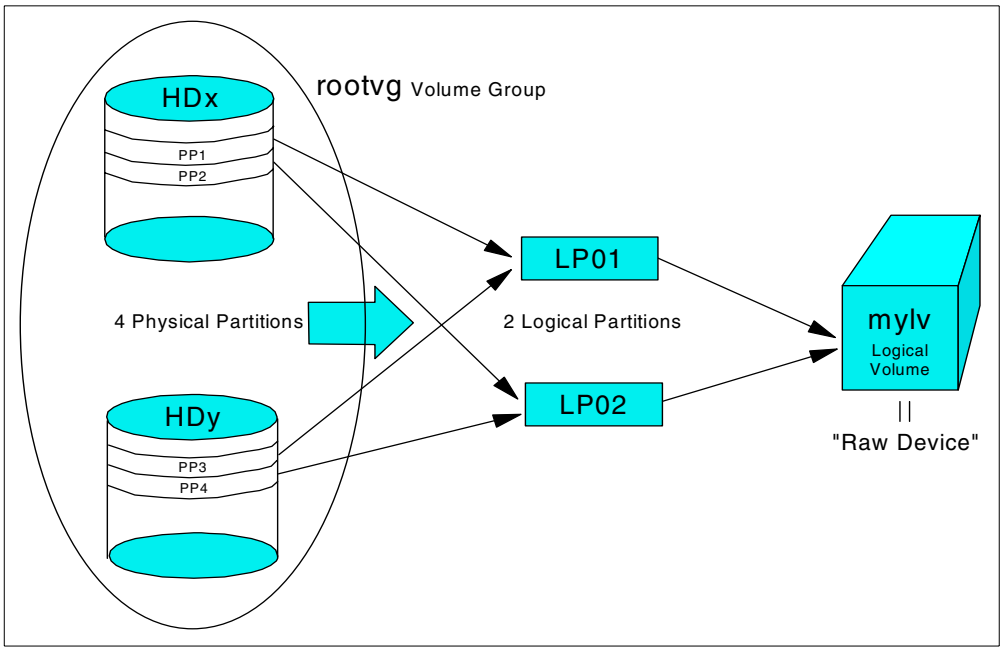


Figure 13. Relationship between a logical volume and its components

The creation of a logical volume is shown in 4.1.2, “Creating table space containers” on page 48. For all the references to the AIX device management, refer also to *AIX Version 4 System Management Guide: Operating System and Devices*, SC23-2525.

The physical design of the database is the first important operation while creating a new database. It is important to evenly spread data and index spaces among the available traditional storage media in order to optimize space availability and usage, avoid I/O contention on disks, reduce the risk of losing data, and optimize database access, according to the application data flow.

For example, if you have a large table, often accessed by user applications, it is convenient to separate data and indexes on different table spaces, in different storage media.

For our project, we have created the CIPROS table spaces containers (data files) on raw devices.

The following system groups have been created, using SMIT or the command:

```
mkgroup -'A' group_name
```

- *cipros*: the CIPROS system group
- *mqm*: the MQSeries owner group
- *dba*: the Oracle DBA group
- *poly*: the Polyserver owner group

The following system users have been created, using SMIT or the command:

```
mkuser pgrp='group_name' user_name
```

- *cipros*: the owner of CIPROS applications, belonging to *cipros* group
- *mqm*: the MQSeries owner, belonging to *mqm* group
- *oracle*: the Oracle DBA user, belonging to *dba* group
- *poly*: the Polyserver owner, belonging to *poly* group

4.1.2 Creating table space containers

For each one of the Oracle table spaces, a set of raw devices has been created, using the AIX SMIT tool or with the command `mklv`, as in the following example:

```
mklv -y'mylv' -t'raw' rootvg 12 hdisk3
```

With this command, the *mylv* device is created on disk device hdisk3, in the rootvg volume group, with 48 MB of allocated space distributed across 12 physical partitions (PP) of 4 MB (no similarity to the DB2 partitions in OS/390) with *raw* as logical volume type. This is just a description, but it can be useful to differentiate the table space containers across all the system logical volumes.

In this example no mirroring has been used for the logical volume. This means that there is a one-to-one correspondence between the logical and the physical partitions, that i, each physical partition has just one copy in the logical volume.

If we want to create a mirrored logical volume, as shown in Figure 13 on page 47, the following command must be issued:

```
mklv -y'mylv' -t'raw' -c'2' rootvg 2 hdiskx hdisky
```

This command can be issued for all the data containers of the database.

For the CIPROS table spaces, the following raw devices have been created, as listed in Table 16.

Table 16. List of Oracle CIPROS containers

Logical volume name	Volume group	# of PP	Description
baselv	rootvg	12	CPRS_BASE table space
readdtlv	rootvg	1	CPRS_READING table space
readixlv	rootvg	1	CPRS_READING_IND table space
lablv	rootvg	1	CPRS_LAB table space
logelv	rootvg	1	CPRS_LOGE table space
prdscnlv	rootvg	1	CPRS_PRDSCN table space

Note: The sizes of the data table spaces of our project database are very small. In an operating environment, usually the READING table spaces (both data and index) should be dimensioned according to the daily data flow coming from the Real Time database and according to the timeframe the customer wants to keep on-line in the database. In real-life production environments, the typical size for the READING table is 20 to 30 GB.

In addition to the user data table spaces, an Oracle database also requires a set of default table spaces and system objects:

- Control files
- System catalog table space
- Rollback segments table space
- Temporary table space
- USERS default table space
- TOOLS default table space

For these objects, a set of raw devices has been created as listed in Table 17:

Table 17. List of Oracle system containers

Logical volume name	Volume group	# of PP	Description
systemlv	rootvg	16	CIPROS database catalog
rbs01lv	rootvg	8	Rollback segments table space
temp01lv	rootvg	4	Temporary table space
toolslv	rootvg	1	TOOLS default Oracle table space
userslv	rootvg	1	USERS default Oracle table space
crtl01lv	rootvg	1	Control file, copy #1
crtl02lv	rootvg	1	Control file, copy #2
crtl03lv	rootvg	1	Control file, copy #3

In our project we did not specify the disk each logical volume has been created on. This should be always done in a real operating environment. For instance, the control files should be allocated on different disks.

An Oracle database requires also a specific space for the *redo log* files, which record all the changes made to the database before being written on the dedicated storage media.

In our database, for the Oracle redo log files, a specific JFS 32 MB file system, allocated on */home/oracle/redolog,* has been created.

4.1.3 The CIPROS database

Oracle 7.3 RDBMS has been installed in */home/oracle* directory. Refer to the *Oracle7 Release 7.3 for AIX Installation Guide*, A43771-1 for detailed information on Oracle installation.

The */home/oracle* directory is a 396 MB file system owned by the *oracle* system user, belonging to the *dba* system group. Then, a *cipros* database has been created (*cipros* instance, that is ORACLE_SID=*cipros*), using the system containers indicated in Table 17 on page 49 and the Oracle default scriptcrdb*cipros.*sql by executing crdb2*cipros*.sql for the creation of the database (modified for our environment) and of all the system objects (the table spaces *system, temp, rbs, tools*, *users*, the system catalog, the default administration users *internal, sys* and *system*). Please refer to *ORACLE 7, The Complete Reference,* ISBN 0-07-882285-8, and to the ORACLE 7 standard documentation for further details on Oracle installation and configuration.

The content of the two files is listed in Figure 14 and Figure 15.

```
spool /home/oracle/app/oracle/admin/cipros/create/crdbcipros.lst

connect internal

startup nomount
pfile=/home/oracle/app/oracle/admin/cipros/pfile/initcipros_0.ora

create database "cipros"
    maxinstances 8
    maxlogfiles  32
    maxdatafiles 60
    character set "US7ASCII"
    datafile
'/dev/rsystemlv'size   65000K
    logfile
'/home/oracle/redolog/redocipros01.log'size 4M,
'/home/oracle/redolog/redocipros02.log'size 4M,
'/home/oracle/redolog/redocipros03.log'size 4M;

disconnect
spool off
```

Figure 14. crdbcipros.sql file

```
spool crdb2cipros.lst
connect internal
@/home/oracle/app/oracle/product/7.3.3/rdbms/admin/catalog.sql
connect internal

create rollback segment r0 tablespace system
storage (
initial 16k
next 16k minextents
2 maxextents 20);

alter rollback segment r0 online;

create tablespace rbs datafile
'/dev/rrbs01lv'size   32000K
default storage (
initial1M
next        1M
pctincrease   0
minextents   2);

create tablespace temp datafile
 '/dev/rtemp01lv'size   16000K
default storage (
initial      256k
next         256k
        pctincrease  0);

create tablespace tools datafile
'/dev/rtoolslv'size  4000K;

create tablespace users datafile
'/dev/ruserslv'size   4000K;

create rollback segment r01 tablespace rbs;
create rollback segment r02 tablespace rbs;
create rollback segment r03 tablespace rbs;
create rollback segment r04 tablespace rbs;

alter rollback segment r01 online;
alter rollback segment r0 offline;
drop rollback segment r0;

alter user sys temporary tablespace temp;
alter user system default tablespace tools temporary tablespace temp;

connect system/manager
@/home/oracle/app/oracle/product/7.3.3/rdbms/admin/catdbsyn.sql

spool off
```

Figure 15. crdb2cipros.sql file

The configuration of the Oracle database is written into two Oracle specific files:

- init*cipros*.ora
- config*cipros*.ora

as listed in Figure 16 and Figure 17.

```
#
# $Header: initx.orc 1.1 95/02/27 12:14:56 wyim Osd<unix> $ Copyr (c) 1992
Oracle
#

ifile = /home/oracle/app/oracle/admin/cipros/pfile/configcipros.ora

rollback_segments = (r01,r02,r03,r04)

# tuning parameters

db_files = 60

db_file_multiblock_read_count = 8                                    # SMALL
db_block_buffers = 200                                               # SMALL
shared_pool_size = 3500000                                             # SMALL
log_checkpoint_interval = 10000
processes = 50                                                       # SMALL
dml_locks = 100                                                      # SMALL
log_buffer = 8192                                                      # SMALL
sequence_cache_entries = 10                                          # SMALL
sequence_cache_hash_buckets = 10                                     # SMALL
max_dump_file_size = 10240      # limit trace file size to 5 Meg each

open_cursors = 200
compatible = 7.3.0.0.0
global_names = TRUE
```

Figure 16. initcipros.ora file

```
*#
# $Header: cnfg.orc 1.1 95/02/27 12:14:25 wyim Osd<unix> $ Copyr (c) 1992
Oracle
#
# cnfg.ora - instance configuration parameters

control_files           = (/dev/rctrl01lv,
                           /dev/rctrl02lv,
                           /dev/rctrl03lv)
background_dump_dest    = /home/oracle/app/oracle/admin/cipros/bdump
core_dump_dest = /home/oracle/app/oracle/admin/cipros/cdump
user_dump_dest = /home/oracle/app/oracle/admin/cipros/udump
#log_archive_dest = /home/oracle/app/oracle/admin/cipros/arch/arch.log
#db_block_size <blocksize>

db_name                 = cipros
```

Figure 17. configcipros.ora file

After the base installation, we have also issued the *pupbld.sql* script (for the creation of SQLPlus product and user profiles) and *catproc.sql* (for the creation of system tables, views, procedures and packages).

For a detailed description of the Oracle database objects, please refer to *ORACLE 7, The Complete Reference, ISBN 0-07-882285-8,* and the ORACLE 7 standard documentation.

The Process and Laboratory segments of the CIPROS database are made up of six table spaces:

- CPRS_BASE: base reference tables (used by all database areas, including Process and Laboratory areas)
- CPRS_LAB: Laboratory specific tables
- CPRS_READING: READING table data (Process area)
- CPRS_READING_IND: READING table index (Process area)
- CPRS_LOGE: Log Event feature tables (bridges)
- CPRS_PRDSCN: Process and Laboratory Data Analysis tool specific tables (both Process and Laboratory areas)

Each one of these table spaces have been created on the related logical volume, as listed in Table 16 on page 48, with an Oracle DDL command as following:

```
CREATE TABLESPACECPRS_BASE
  DATAFILE'/dev/rbaselv'
  SIZE 48000K
  DEFAULT STORAGE
   (MAXEXTENTS UNLIMITED
  PCTINCREASE 0);
```

Note: The creation of the all the CIPROS table spaces is performed by the CIPROS installation procedure, after configuring a specific definition file. In our scope of work it is interesting to describe at least one example of table space creation.

To run the execution of the statements, we have used SQLPlus, the standard Oracle command line, that can be issued by a *cipros* shell with the command:

```
sqlplus
```

In this case, we can also create a file (for instance, *crtbbase.sql*) containing the CREATE TABLESPACE statement listed above, and then run the command:

```
sqlplus system/manager @crtbbase.sql
```

Note: The table space must be created on the *character device* corresponding to the *baselv* logical volume (the *raw device*) and not on the */dev/baselv* device (block device).

Note: The device file */dev/rbaselv* must be owned by the Oracle user. Since, at the first creation, the device belongs to root/system, before creating the table space we had to change the ownership of the file with the command:

```
chown oracle:dba /dev/rbaselv
```

4.1.4 Process and Laboratory bridges

An important segment of the CIPROS solution is constituted by the *bridges.*

As we described in 2.1.2, “CIPROS components” on page 6, bridges allow the CIPROS database to exchange data with all the other refinery areas, providing, with their mapping facility, standard and unique information to the end-user applications.

An important part of CIPROS communication segment is also the *Dispatcher*, an MQSeries-based module which allows communication synchronization between source (poller) and target (loader) machines.

We do not analyze the dispatcher modules in our conversion study, concentrating our analysis on the ***Process loader*** and the ***Laboratory loader***, that are the two C-language programs that interface directly with the CIPROS database, by loading the related data.

The structure of the C programs is shown in Figure 18:

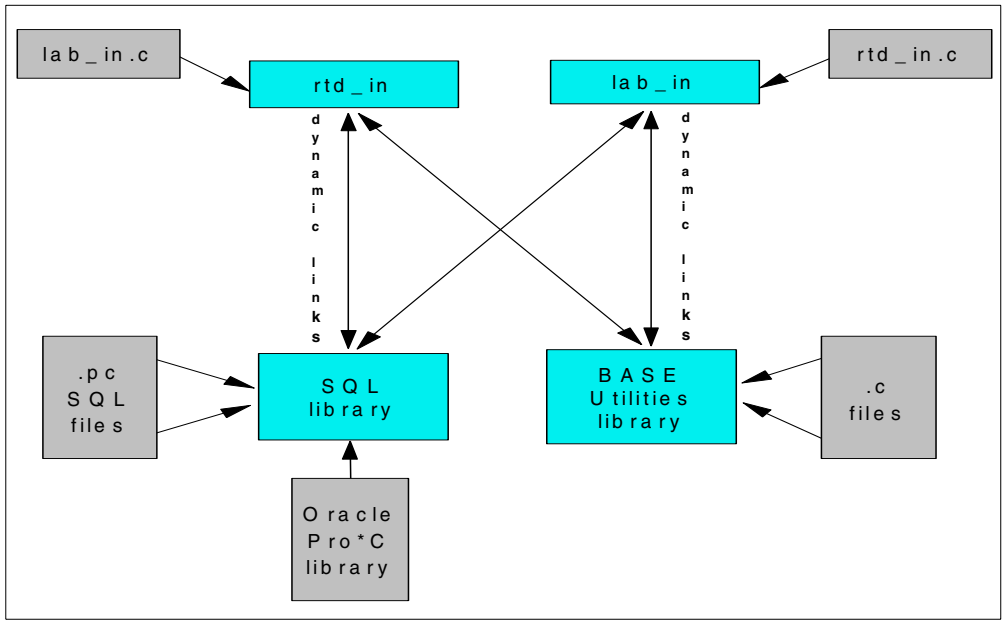


Figure 18. Relationship diagram of CIPROS Process and Laboratory bridges

The main executable programs are *rtd_in* and *lab_in*, whose sources are constituted respectively by the *rtd_in.c* and *lab_in.c* source files.

The two programs dynamically link to the following two Dynamic Linked Libraries (DLLs):

- The SQL functions library consists of several *.pc* source files, linked in a unique object file, and then compiled as a dynamic library with the *ar* AIX C compiler command; the SQL library is also precompiled with the Oracle Pro*C precompiler files. This library contains the SQL functions for querying, inserting and updating CIPROS tables.
- The BASE utilities library consists of several *.c* source files, linked in a unique object file and then compiled as a dynamic library as the SQL library. This library contains general-purpose functions, such as opening, reading, writing, and closing a file, accessing the message catalog, and so on.

4.1.4.1 Process loader

The Process loader bridge is responsible to load into CIPROS, in a controlled way, data related to Process values.

The CIPROS Process loader performs two main functions:

- Define/update INSTRUMENT related tables occurrences (INSTRUMENT, TAG)
- Insert/update READING occurrences

The CIPROS loader bridge is supposed to be the authoritative source for definition of occurrences inside the relational database tables.

This means that all the information that allows the program to define/update tables on the relational database must be present in the transfer file used by the bridge. This enables CIPROS, in cooperation with the poller part of the bridge that runs on the Real Time side, to automatically define into the CIPROS database a new TAG, added in the REAL TIME environment, and from this moment on, to start storing the values related to it inside CIPROS database.

The same procedure is also followed for the update function: Changes on the Real Time side are automatically reflected into the relational database.

4.1.4.2 Laboratory loader

The Laboratory loader bridge is responsible to load into CIPROS, in a controlled way, data related to Laboratory values.

Starting from the SAMPLE_POINT, occurrences of all entities can be automatically defined inside CIPROS Laboratory segment of the CIPROS Data Model, starting from the information contained in the transfer file records coming from the Laboratory system during the bridge process.

ANALYSIS_METHOD and STANDARD_RESULT, together with SAMPLE_POINT definition, make the 'static' part of the lab tables. The loader can automatically create new occurrences into these tables, as well as new occurrences into the SAMPLE and SAMPLE_ANALISYS (the tables that contain the information on sample and values of analysis performed on it).

The loader is also responsible for mapping operations between the two environments using mapping tables.

4.1.5 The data

The Process and Laboratory data segment of the CIPROS Data Model, as described in 2.2, “Scope of work for the project” on page 9, contains reference and operating tables for the refinery data coming from Real Time (process) and Laboratory databases.

4.1.5.1 The tables

We divide the related tables into four groups:

- ***General tables***: These tables contain general data used by all the applications (such as the ENGINEERING_UNIT tables) and the "domain" tables provided for decoding some general codes used by all the applications (such as D_IN_OR_OUT, containing the code used for INput or OUTput streams).
- ***Reference tables***: These tables define the facilities of the plant, the materials, the parameters related to the units of operation and so on. Inside them you can find the definitions of the UNITS, the TANKS, the PIPELINES, the LOADING RACKS. Also stored here are the STREAMS, tables that contain the definitions of input and output points of the materials to and from the UNITS.
- ***Process tables***: These tables contain information related to instrument measurements with the related timestamp. The INSTRUMENT may represent a real instrument in the plant field or not. Different TAGS, that represent different types of information, can be related to an INSTRUMENT. Values are stored in a table called READING with their own timestamp and percentage of reliability.
- ***Laboratory tables***: These tables contain information related to the results of the analyses performed on a SAMPLE that is taken in the plant from a facility. Once a SAMPLE_POINT in the plant is defined, all the results of the analyses performed on the SAMPLE taken from that SAMPLE_POINT can be stored in the table called SAMPLE_ANALYSIS. In the table structure there are also the two tables ANALYSIS_METHOD and STANDARD_RESULT which, together, uniquely identify the "name" of Laboratory analysis performed on a sample.

In addition to these types of tables, we also have some tables containing specific information related to the CIPROS GUI desktop (such as the HELP_TABLE table, containing the on-line help of the CIPROS GUI).

4.1.5.2 Oracle data types used in CIPROS data

In general, CIPROS table definitions use several Oracle data types. Let us consider some examples.

In Figure 19 you see an example of a General table, containing just VARCHAR2 fields, and the output of a SELECT statement on the table.

```
SQL> desc compos_type;
 Name                           Null?    Type
 ------------------------------ -------- ----
 COMPOS_TYPE                    NOT NULL VARCHAR2(10)
 DESCRIPTION                             VARCHAR2(40)

SQL> select * from compos_type;

COMPOS_TYP DESCRIPTION
---------- ----------------------------------------
VOLUME     Volume
MASS       Mass
```

Figure 19. Example of a General table

In Figure 20 you see an example of a Reference table, containing also NUMBER and NUMBER(n) fields, with the output of a SELECT statement on the table.

```
SQL> desc loading_rack
 Name                           Null?    Type
 ------------------------------ -------- ----
 LOAD_RACK_NAME                 NOT NULL VARCHAR2(20)
 LOAD_RACK_TYPE                          VARCHAR2(10)
 LOAD_PT_NUM                             NUMBER(3)
 FEED_LINE_DIAM                          NUMBER
 ENG_UNT_FLDIAM                          VARCHAR2(10)
 DESCRIPTION                             VARCHAR2(40)
 PRINT_FNAME                             VARCHAR2(8)

SQL> select * from loading_rack;

LOAD_RACK_NAME       LOAD_RACK_ LOAD_PT_NUM FEED_LINE_DIAM ENG_UNT_FL
-------------------- ---------- ----------- -------------- ----------
DESCRIPTION                                PRINT_FN
---------------------------------------- --------
LR1                  BARGE                1             32 ft
Loading rack 1 - Barge loader            LR1

LR2                  TRUCK                2             62 ft
Loading rack 2 - Truck loader            LR2

LR3                  RAIL                 3             23 m
Loading rack 3 - Rail loader             LR3

LR4                  SHIP                 4            213 ft
Loading rack 4 - Ship loader             LR4
```

Figure 20. Example of a Reference table

In Figure 21 you see an example of a Laboratory table, containing DATE and CHAR fields.

```
SQL> desc sample_analysis
 Name                           Null?    Type
 ------------------------------ -------- ----
 SAMPLE_PT_NAME                 NOT NULL VARCHAR2(20)
 TIMESTAMP                      NOT NULL DATE
 ANLY_METH_NAME                 NOT NULL VARCHAR2(20)
 STD_RESLT_NAME                 NOT NULL VARCHAR2(20)
 ANLY_VALUE                              VARCHAR2(20)
 FLAG_RANGE                              CHAR(1)
 FLAG_CONF                               CHAR(1)
 FLAG_RECONF                             CHAR(1)

SQL> select * from sample_analysis where STD_RESLT_NAME='D15';

SAMPLE_PT_NAME       TIMESTAMP ANLY_METH_NAME       STD_RESLT_NAME
-------------------- --------- -------------------- --------------------
ANLY_VALUE           F F F
-------------------- - - -
T_0001_SP            03-JUN-99 ASTM D-86            D15
0.801                Y Y Y

T_0002_SP            03-JUN-99 ASTM D-86            D15
0.928                Y Y Y

T_0003_SP            03-JUN-99 ASTM D-86            D15
0.827                Y Y Y

T_0004_SP            03-JUN-99 ASTM D-86            D15
0.492                Y Y Y
```

Figure 21. Example of a Laboratory table

In Figure 22 you see an example of a Process table.

```
SQL> desc reading_text;
 Name                           Null?    Type
 ------------------------------ -------- ----
 TAG_NAME                       NOT NULL VARCHAR2(20)
 TIMESTAMP                      NOT NULL DATE
 VALUE                                   VARCHAR2(1024)

SQL> select * from reading_text where TAG_NAME='CDU_STATUS.SNP';

TAG_NAME             TIMESTAMP
-------------------- ---------
VALUE
--------------------------------------------------------------------------------
------
CDU_STATUS.SNP       03-JUN-99
Disconnected
```

Figure 22. Example of a Process table

4.2 The target system environment

In this section we describe the DB2 for OS/390 environment and its configuration.

4.2.1 Configuration

The mainframe system is a sysplex. Our TSO, known to us as SC63TS, is running as a logical partition (LPAR) on 9672-R76 hardware with OS/390 Version 2 Release 7 (OS/390 for the purposes of this book). The relational database management system is DB2 UDB for OS/390 Version 6.

OS/390 is packaged together with all products that are considered part of the system environment. TCP/IP, for example is included in the packaging and does not need to be added to the environment. All various software parts of the system are at the same level of the operating system.

A list of the base features of OS/390 V2 R7 can be found in *OS/390 V2R7.0 Planning for Installation,* GC28-1726-06.

The software subsystems, such as DB2, are separate products.

4.2.2 Security

Data stored in DB2 can be accessed by using the DB2 access functions or by directly accessing the data objects at the physical data set level. In both cases the appropriate level of access control must be in place. The security implementation at the target system is based on standard RACF and DB2 functions which map access levels between DB2 objects and authorization identifiers (IDs). RACF makes sure that only the DB2 subsystem can access its data and, within DB2, RACF and DB2 security functions provide the IDs with a set of granular privileges related to DB2 objects such as databases, tables, and rows, and the type of actions that can performed on them.

The logon process is described in Appendix D, “OS/390 TSO tools and tips” on page 257.

4.2.3 Backup

The backup and recovery strategy for the target system is based on standard DB2 utilities (Image COPY, RECOVER, REBUILD, and so on). These are covered in detail in the *DB2 UDB Server for OS/390 V6 Administration Guide,* SC26-9003, and the D*B2 UDB Server for OS/390 V6 Utility Guide and Reference,* SC26-9015.

We have made backup copies of the database table spaces with the normal DB2 image copy utility. This is run at regular intervals during the data migration process. If recovery is needed, it is achieved in one of two ways:

- If the data has not changed from the initial load, the table (or tables) may be easily dropped, recreated, and reloaded. This may be attractive early on, due to the referential integrity issue.
- Later on, once the data has been loaded and possibly updated, the needed recovery strategy based on cyclic execution of COPY, REORG, secure archiving of logs, and tested recovery procedures must be in place.

You will find sample JCL for image copies and other utilities in Appendix B, “Sample DB2 for OS/390 jobs” on page 215.

4.2.4 Partitioned data sets

In a native OS/390 environment a file hierarchy is maintained through the use of up to three levels of qualification for their names. The three levels of qualification are defined with the names *project*, *group* and *type*.

Our project has been named CIPROS, and this name was chosen as the first level qualification of the PDS. Since most of the code and changes are related to the C language, the second level of qualification has been designated as "C". The third level relates to the various kinds of files that are needed, such as source, include, JCL, DCLGEN, DLL, DBRM, load, and so on.

PDS can be used to maintain a library of members with this approach. Inside each PDS is a set of members whose attributes are the same. For example, all of the C source code is maintained in the CIPROS.C.SOURCE PDS, each with its own member. The fully qualified name would be CIPROS.C.SOURCE(BRIDGE).

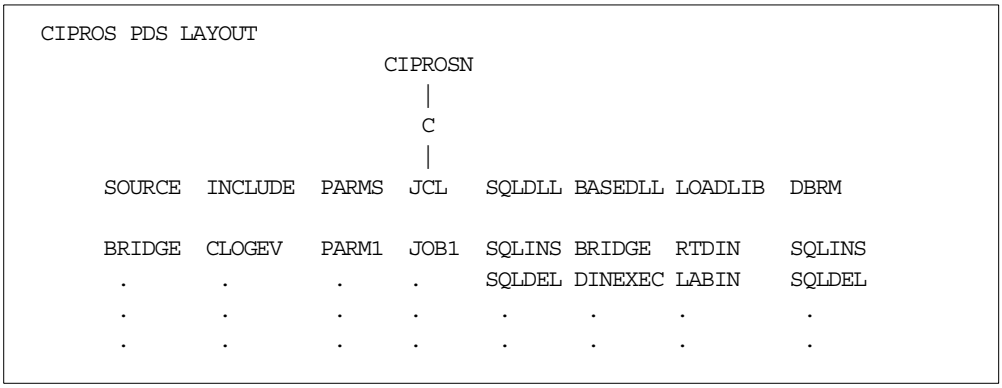


Figure 23. A typical PDS hierarchy.

4.2.5 Communications

See 6.5, “Transferring data from AIX to OS/390” on page 118 for examples of the functions utilized during the project.

4.2.6 Compilers

The C compiler used was the OS/390 R2 V7 C compiler. An example of the compile JCL used is reported in Appendix B, “Sample DB2 for OS/390 jobs” on page 215. The JCL procedures used may be found in your installation system procedure library. Often this library is named "SYS1.PROCLIB".

Chapter 5. Database conversion

In this chapter we discuss the database conversion tasks, which include:

- DB2 environment
- Database design
- Inventory
- Data layout
- Cross reference
- Oracle and DB2 feature incompatibilities

5.1 DB2 environment

The setup for the DB2 environment specific for the conversion is a once-only task discussed in 3.3.1.2, “DB2 environment” on page 34, which includes the following activities:

- Installation and configuration
- Security (including userid)
- Backup and recovery
- System work space
- Communications
- Sample DB2 database and code

The project assumption is that the DB2 subsystem is already established and in use at the target site. We only highlight the changes necessary to accommodate the new converted application either in an existing DB2 subsystem, or in a new dedicated one.

5.1.1 Installation and configuration

The installation and configuration of a DB2 subsystem is achieved by filling in the various parameters presented by the several panels that constitute the installation DSNTINST CLIST and then executing several jobs that are part of the output of the CLIST execution.

Normally, the addition of a new DB2 application in an environment where DB2 subsystems are already in production requires a minimal update of the installation parameters. The decision of defining a new DB2 subsystem depends on the size and the service level requirements of the application being migrated. For more information on the various aspects of the installation, refer to *DB2 UDB for OS/390 Version 6 Installation Guide*, GC26-9008.

5.1.2 Security

Security in OS/390 and DB2 is usually achieved by a combination of usage of Resource Access Control Facility (RACF) and the DB2 provided security functions. RACF, or equivalent product, is the controller of accesses to the system and its data at logon time. RACF is designed to work with the various access points to an OS/390 system, such as TSO.

Users are part of a RACF group, and the group has more or less access to menu options and data. If a group or user has access to DB2, DB2 may still control access to various functions and tables by use of GRANTS of authority to perform those functions or access the tables. For instance, SELECT authority may be granted to PUBLIC for a particular table. Under PUBLIC, anyone with access to the DB2 subsystem could view the contents of the table. However, UPDATE, INSERT, or DELETE authority may be granted only to specific users, or not at all.

The two-layer combination of access control software and DB2 internal security facilities has been the most commonly used one. However, some shops are moving toward placing all security responsibilities in the hands of a systems security group. These people usually do not know DB2, nor do they need to. Access control can be achieved though software such as RACF alone. DB2 V5 and OS/390 Release 4 have introduced the possibility of delegating to RACF the DB2 security definitions utilizing the RACF/DB2 External Security Module. See *Ready for e-business: OS/390 Security Server Enhancements*, SG24-5158 for a description of this function and related bibliography.

Security for our project, beyond the login security to OS/390 TSO/ISPF, is achieved through the use of GRANT and REVOKE DB2 commands.

Backup and recovery is achieved through the normal DB2 facilities provided. These subjects are covered in detail in the *DB2 UDB for OS/390 V6 Administration Guide,* SC26-9003, and the *DB2 UDB for OS/390 V6 Utility Guide and Reference*, SC26-9015. Examples of the backup and recovery jobs used during the project can be found in B.11, “JCL for REORG, RUNSTATS and COPY of CIPROS table spaces” on page 223, B.12, “JCL for RECOVER of a CIPROS table space” on page 226, and B.13, “JCL for rebuilding a CIPROS index” on page 227.

5.1.3 Sample DB2 database and code

DB2 provides a full set of samples. The sample libraries also contain jobs to establish facilities you may need — for example, User Defined Functions or UDF. The samples cover code for various languages, the usage of the various utilities, setting up, maintaining and accessing sample databases. Sample jobs provide examples of JCL for the use of LOAD, REORG, COPY, and other utilities. The sample databases themselves demonstrate the use of various DB2 features, such as partitioned and segmented table spaces. These libraries reside in data sets defined according to your site standards. They might be found under names such as ’DB2V610.NEW.SDSNSAMP’ which is the default name of the version of DB2V610.SDSNSAMP customized by the DSNTINST installation procedure.

For our project, we executed job DSNTEJ1. Refer to Appendix B, “Sample DB2 for OS/390 jobs” on page 215 for a listing of this job. DSNTEJ1 establishes a sample database and executes various utilities for it. This was done to establish a DB2 environment in which to test the C language precompile, compile, prelink, link, bind and run in preparation for using similar jobs to execute the projects converted C language code.

Job DNSTEJ2D, also found in Appendix B, “Sample DB2 for OS/390 jobs” on page 215, produces the executable C language version of the sample phone application and runs it. The sample produces several updates to the sample tables and writes a report.

Both of these jobs ran with minor modifications to the data set names and DB2 system parameters. The sample installed high level qualifier, DB2V610X, needed to be set, and our instance of DB2 known as DB2X. See Figure 24.

```
//SYSTSIN   DD  *
 DSN SYSTEM(DB2X)
 BIND PLAN(DSN8BD61) MEMBER(DSN8BD3) ACT(REP) ISOLATION(CS)
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61)  -
      LIB('DB2V610X.RUNLIB.LOAD')
 RUN  PROGRAM(DSN8BD3) PLAN(DSN8BD61) -
      LIB('DB2V610X.RUNLIB.LOAD')
 END
```

Figure 24. Sample JCL changes example

User defined functions

Job DSNTEJ2U creates a number of DB2 user defined functions, UDF. See B.21, “DSNTEJ2U - DB2 sample JCL to create user defined functions” on page 240. The UDFs provide special formatting and handling for data. In particular, we needed the ALTDATE and ALTTIME UDFs.

To use UDFs, you must ensure that some prerequisites are in place. First, the Workload Management (WLM) application environment must be in place: UDFs are handled like stored procedures with WLM. A good source of information for setting up the environment for stored procedures is the redbook *WOW! DRDA Supports TCP/IP: DB2 Server for OS/390 and DB2 Universal Database*, SG242212.

WLM has requirements for Resource Measurement Facility (RMF) and, since we are operating in a client/server environment, the RACF Remote Sharing Facility must be active for Distributed Relational Database Architecture (DRDA) use. The functionalities of these required products are described in detail in the *OS/390 V2R6.0 RMF User's Guide,* SC26-9015; *OS/390 V2R7.0 MVS Workload Management Services,* GC28-1773-06; *OS/390 V2R7.0 MVS Planning: Workload Management,* GC28-1761-08; and *DB2 UDB for OS/390 V6 Administration Guide*, SC26-9003.

In creating the DB2 UDFs and compiling the supporting C language programs, we noticed that the C programs needed to be modified to be fetchable and to compile properly. The C programs associated with the UDFs are DSN8DUAD, DSN8DUAT, DSN8DUCD, DSN8DUCT, DSN8DUCY and DSN8DUTI. They may be found in a PDS with a name similar to DSN610.SDSNSAMP. To have them compile correctly in the job, you have to add the statement found in Figure 25 to the code. This line should be added as the first line of code in the C program. You need to change the program name to match the program you are inserting the code into. The program name in the example is DSN8DUAD.

```
#pragma linkage(DSN8DUAD,fetchable)
```

Figure 25. Pragma needed in UDFs

Since you might be using DRDA, and you will need it if you consider the DataJoiner option discussed, we recommend that the DB2 system parameter for extended security be set to YES. It can be set in the DSNTIPR panel of the installation procedure or in the system installation job DSNTIJUZ in the macro DSN6SYP, as shown in Figure 35 on page 83. This setting allows more informative error messages to be returned with detailed reason codes when security problems occur. This helps the DRDA clients in problem determination without involving the host operator. See the *DB2 UDB for OS/390 V6 Installation Guide,* GC26-9008 for a discussion of this and other parameters.

```
DSN6SYSP    AUDITST=NO,                                          X00410086
              BACKODUR=5,                                          X00410087
              CONDBAT=64,                                          X00410088
              CTHREAD=70,                                          X00410089
              DBPROTCL=DRDA,                                       X00410090
              DLDFREQ=5,                                           X00410091
              DSSTIME=5,                                           X00410092
              EXTRAREQ=100,                                        X00410093
              EXTRASRV=100,                                        X00410094
              IDBACK=20,                                           X00410095
              IDFORE=40,                                           X00410096
              IDXBPOOL=BP0,                                        X00410097
              LBACKOUT=AUTO,                                       X00410098
              LOBVALA=2048,                                        X00410099
              LOBVALS=2048,                                        X00410100
              LOGAPSTG=0,                                          X00410101
              LOGLOAD=50000,                                       X00410102
              MAXDBAT=64,                                          X00410103
              MON=NO,                                              X00410104
              MONSIZE=8192,                                        X00410105
              PCLOSEN=5,                                           X00410106
              PCLOSET=10,                                          X00410107
              RLF=NO,                                              X00410108
              RLFTBL=01,                                           X00410109
              RLFERR=NOLIMIT,                                      X00410110
              RLFAUTH=SYSIBM,                                      X00410111
              ROUTCDE=(1),                                         X00410112
              EXTSEC=YES,                                          X00410113
              SMFACCT=(1),                                         X00410114
              SMFSTAT=YES,                                         X00410115
              STATIME=30,                                          X00410116
              STORMXAB=0,                                          X00410117
              STORPROC=DB2ZSPAS,                                   X00410118
              STORTIME=180,                                        X00410119
              TBSBPOOL=BP0,                                        X00410120
              TRACSTR=NO,                                          X00410121
              TRACTBL=16,                                          X00410122
              URCHKTH=0,                                           X00410123
              WLMENV=                                               00410124
```

Figure 26. Example of the DSNSYSP macro

5.2 Database design

The database design is a once-only task, discussed in 3.3.1.3, “Database design” on page 35, which includes the following tasks:

- Physical design
- Logical design

5.2.1 Physical design

5.2.1.1 Approach to data definition and SQL

The Data Definition Language (DDL) and Data Manipulation Language (DML) used to establish the DB2 database on OS/390 was produced by scripts written and run on the AIX system.

To achieve a set of tables and data that matched the Oracle/AIX design as closely as possible, the information contained in the Oracle catalogue was used to model the code. The scripts produced the DDL, which was then modified by hand where needed, though little in the way of manual intervention was required, except to use the files of SQL to build jobs to run on OS/390. The approach was deliberate and step-wise. The DDL for the table spaces was written manually on OS/390. The DDL for the tables, indexes, views, primary, and foreign keys was almost entirely produced by scripts on AIX. To keep the complexity of the scripts under control, each of the components of the database definition was done separately. This is why, for example, in the CREATE TABLE SQL, you will see no parameters for designation of primary or foreign keys. These were done later with ALTER TABLE statements.

5.2.1.2 The database creation

The database creation is achieved either of two ways. The SQL needed to establish the database can be executed in interactive DB2 (DB2I option under TSO), or through a sample program interface in an OS/390 batch job stream.

We chose to use the batch approach. The JCL can be found in Appendix B, “Sample DB2 for OS/390 jobs” on page 215. The jobs are named beginning with CIPROS. The database creation job is CIPROSDB, for example. The job CIPROSDB was used to create the storage group, database, table spaces and tables.

Our first step was to establish a storage group. The storage group establishes where the database reside on the server by designating the volume or volumes the database will use. Notes and details of the syntax of this and other SQL statements discussed here can be found in *DB2 UDB for OS/390 V6 SQL Reference*, SC26-9014.

Since our set of tables has a relatively small amount of data, the storage group to hold our database is only on one volume. The statement for storage group creation is:

```
CREATE STOGROUP CIPROS01
    VOLUMES ("SBOX10")
    VCAT  DB2V610X;
```

Our volume is named SBOX10, and the catalog for the database is the one established for DB2 at DB2 install time. We had the luxury of being the only users of this DB2 subsystem.

The database is created with the following statement:

```
CREATE DATABASE CIPROS
   STOGROUP CIPROS01
   BUFFERPOOL BP0
   CCSID EBCDIC;
```

In our case, the statement specifies buffer pool as BP0, which is also the default in our system. Buffer pools are used by DB2 as temporary storage needed during I/O activity. For example, if a program selects a row of data from a table, the page containing that row is placed in a buffer in the buffer pool in memory. Buffers increase performance by lowering the amount of data movement needed. For our small application, BP0 was used. In real production you may want to point your data to a buffer pool different from BP0 and even set up a different default at system level. You need to talk to your DBA to define number and sizes of the buffer pools involved. Information on buffer pools and their management can be found in *DB2 UDB for OS/390 V6 Administration Guide,* SC26-9003.

The CCSID parameter sets the encoding scheme for the data. Since the converted system runs on a standard OS/390 system, we explicitly set the encoding to EBCDIC.

5.2.1.3 Table space definitions

Table space is the DB2 basic unit of data storage. A table space can hold one or more tables. The table space may be simple, segmented, or partitioned. Both simple and segmented table spaces may hold one or many tables. The difference is in how the data is held and managed. Normally, you use the segmented table spaces in both cases, trying to group tables that are relatively small and have similar recovery requirements, or assigning the whole segmented table space to larger tables. If the table is really large, you might consider the partitioned table space, which holds pages only of a single table, but spreads them over multiple partitions based on index ranges. This can help in providing better performance and easier operations. Each has advantages and disadvantages, depending on your data. Information and comparisons are reported in *DB2 UDB for OS/390 V6 Administration Guide,* SC26-9003. We chose the segmented table space approach mainly because we have several relatively small tables in the application.

The DDL for table space creation is shown in Figure 27.

```
CREATE TABLESPACE CPRSBASE
    IN CIPROS
    USING STOGROUP CIPROS01
              PRIQTY 150000
              SECQTY 20
              ERASE NO
    LOCKSIZE PAGE LOCKMAX SYSTEM
    BUFFERPOOL BP0
    CLOSE NO
    SEGSIZE 16
    CCSID EBCDIC;
```

Figure 27. Example of a table space definition

Note: Since DB2 accepts 8 characters table space names as a maximum, all the table spaces have been renamed into 8 character names. Also, no "_" symbols (underscore) are allowed.

Specifying the SEGSIZE parameter in the CREATE TABLESPACE statement implies the creation of a segmented table space.

CPRSBASE is the name of our largest table space holding most of the tables. PRIQTY is the primary space allocation. SECQTY is the secondary allocation amount that is used by DB2 when more space is needed, in this example it is a small value because the data is very stable. The amounts are in KB. ERASE NO indicates that the data is not erased from the disk after a delete operation. The data is only logically deleted and is not accessible by DB2. If YES is used the data is reformatted. LOCKSIZE PAGE LOCKMAX SYSTEM is the default for the granularity of the locks while the application is accessing the data. Row level locking is available, but since our conversion is for a batch subsystem it is not needed and can cause only unnecessary overhead. CLOSE NO indicates that the data set may not be closed when the table is not in use. SEGSIZE tells DB2 the table space is segmented and the integer following it states the number of pages in each segment.

See Table 18 for a list the five CIPROS segmented table spaces defined in DB2 and the number of tables each one contains for a total of 65 tables.

Table 18. DB2 CIPROS table spaces

Table space name	Type	Number of tables
CPRSBASE	Segmented	53
CPRSLAB	Segmented	2
CPRSLOGE	Segmented	3
CPRSREAD	Segmented	2
CPRSPRDS	Segmented	5
Total number of tables		65

5.2.1.4 Users, roles, and groups

Oracle has an internal security management to grant access to the database.

WIth the CREATE USER and CREATE ROLE statements, the Oracle DBA can create, inside Oracle itself, authorized users and groups of users (roles, in Oracle terminology). Users can also be "identified externally" in order to use the operating system security management for the authentication towards the Oracle database.

We can use the CIPROS application to provide an example of Oracle management of users.

The CIPROS application has two standard default users (*cipros* and *cprs_view*) and two standard default roles (*role_cipros* and *role_view*).

The *cipros* and its role *role_cipros* have all the authorizations on CIPROS tables. In particular, *cipros* user is the creator and the owner of CIPROS objects.

The *cprs_view* and its role *role_view* have only SELECT grants on the CIPROS tables.

An example of the SQL statement needed to create the *cipros* user (to be issued by *system* Oracle user from a *SQLplus* command line) is:

```
create user cipros
        identified by ciprospwd
        default tablespace CPRS_BASE
        temporary tablespace TEMP
        quota unlimited on CPRS_BASE
        quota unlimited on CPRS_READING
        quota unlimited on CPRS_READING_IND
        quota unlimited on CPRS_MVMNT
        quota unlimited on CPRS_LOGE
        quota unlimited on CPRS_LAB
        quota unlimited on CPRS_REPV
        quota 5M on USERS
        quota 5M on TEMP;
```

We have then created a role called *role_cipros* with the following statement:

```
create role role_cipros;
```

After that, we have granted access and authorizations to the role *role_cipros:*

```
grant
        ALTER ANY CLUSTER,
        ALTER ANY INDEX,
        ALTER ANY PROCEDURE,
        ...
        ...
        ...
        SELECT ANY SEQUENCE,
        SELECT ANY TABLE,
        UPDATE ANY TABLE
to
        role_cipros;
```

For a complete list of Oracle authorizations and grants, refer to *ORACLE 7, The Complete Reference,* ISBN 0-07-882285-8.

Finally we have granted the role to the user *cipros* with the following Oracle SQL statement:

```
grant role_cipros to cipros;
```

In this way, the *cipros* user inherits all the authorizations defined for the role he is granted.

Security in OS/390 and DB2 is usually achieved by a combination of usage of RACF and of the DB2 provided security functions, refer to 5.1.2, “Security” on page 61.

DB2 provides GRANT and REVOKE Data Control Language statements of SQL to grant and revoke object access or system authorities. Refer also to 5.2.1.9, “Authorizations” on page 85.

In our project, we have decided not to migrate the Oracle original users and roles into RACF/TSO users and groups. We have used the user PAOLOR3 with system administrator level of authority SYSADM to create our database objects.

You should consult your security department before migrating users and groups definitions in order to comply with your security standards.

5.2.1.5 Data types comparison

in this section we briefly describe the data types managed by Oracle and DB2 and point out some differences.

Oracle data types

Table 19 shows the list of basic Oracle data types and their description, refer also to *ORACLE 7, The Complete Reference,* ISBN 0-07-882285-8 for more information.

Table 19. Oracle primary data types

Oracle data type	Description
NUMBER	Numeric field (maximum value: $1*10^{125}$, maximum precision: 38 digits). If NUMBER(n) is specified, the total length of the number can be *n* as maximum (*n*<=126). If NUMBER(d,p) is specified, the number can have *d* total digits with a precision of *p* digits (digits after decimal point)
CHAR(n)	Fixed-length character data, *n* chars long, up to 255 bytes
VARCHAR2(n)	Variable length character string having maximum length n bytes. The maximum length is 2000
RAW(n)	Binary data, *n* bytes long, with *n* up to 255 bytes.
LONG	Character data of variable length up to 2 GB
LONG RAW	Binary data of variable length up to 2GB
DATE	Valid date range from January 1, 4712 BC to December 31, 4712 AD.
ROWID	Hexadecimal string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudo-column.

In Table 20 a list of further Oracle data types is shown. Most of these data types are accepted by the CREATE TABLE statement for compatibility with other SQL databases.

Table 20. Oracle additional data types

Oracle data type	Description
MLSLABEL	Data type format of an operating system label (4 bytes representation). This data type is used primarily with Trusted Oracle.
DECIMAL	Same as NUMBER. If no arguments are passed, a default data type NUMBER(38) is assumed
FLOAT	According to the documentation, it should be the same as NUMBER. But if no arguments are passed, a default of 126 digits is assumed. If FLOAT(n) is specified, the FLOAT field, when inserted, is truncated or approximated. FLOAT(d,p) cannot be specified.
INTEGER	Same as NUMBER(38). INTEGER does not accept arguments for the size. The inserted numbers are approximated to the nearest integer value.
LONG VARCHAR	Same as LONG
SMALLINT	Same as INTEGER
VARCHAR(n)	Same as VARCHAR2, but its use is not advisable since this data type may change in future versions of Oracle.

DB2 for OS/390 V6 data types

DB2 for OS/390 V6 provides a large number of data types, in addition to the possibility for users to define their own data types.

Table 21 provides a list of the DB2 for OS/390 V6 data types, refer to *DB2 UDB for OS/390 Version 6 SQL Reference,* SC26-9014 for a complete description of DB2 data types.

Table 21. DB2 for OS/390 V6 data types

DB2 for OS/390 data type	Description
SMALLINT	Small integer with range (-32768 +32767)
INTEGER INT	Large integer with range (-2147483648 +2147483647)
DEC DECIMAL NUMERIC	Packed decimal number. DECIMAL(d,p) where *d* is the total number of digits (max 31), *p* is the precision. Default for *p* is 0, that is DECIMAL(d) means DECIMAL(d,0). Default for *d* is 5, that is DECIMAL means DECIMAL(5,0)
FLOAT(n)	Floating point number. If 1<=*n*<=21, the format is single precision floating point, if 22<=*n*<=53, the format is double precision floating point
REAL	Single precision floating point number. The range is -7.2E+75 to 7.2E+75. In this range, the largest negative value is -5.4E-79, the smallest positive value is 5.4E-79

DB2 for OS/390 data type	Description
DOUBLE DOUBLE PRECISION FLOAT	Double precision floating point number. The range is -7.2E+75 to 7.2E+75. In this range, the largest negative value is -5.4E-79, the smallest positive value is 5.4E-79
CHAR(n) CHARACTER(n)	Fixed length character string, with 1<=n<=255. Default is *n*=1, that is CHAR means CHAR(1)
VARCHAR(n) CHAR VARYING(n) CHARACTER VARYING(n)	Varying length character string. The value of *n* depends on the page size, and cannot be greater than 32714. For CHAR and VARCHAR data types the "FOR *subtype* DATA" clause can be also specified, where *subtype* can be: - SBCS for single bit data - MIXED for mixed data - BIT for BIT data
CLOB(n) CHAR LARGE OBJECT(n) CHARACTER LARGE OBJECT(n)	Character large object string of maximum length *n,* with 1<=*n*<=2147483647. *n K* or *n M* or *n G* can be also specified, where K means 1024 bytes, M means 1048576 bytes, G means 1073741824 bytes
BLOB(n) BINARY LARGE OBJECT(n)	Binary large object string of maximum length *n,* with 1<=*n*<=2147483647. *n K* or *n M* or *n G* can be also specified, where K means 1024 bytes, M means 1048576 bytes, G means 1073741824 bytes
GRAPHIC(n)	Fixed length graphic string, with 1<=*n*<=127
VARGRAPHIC(n)	Varying length graphic string, 1<=*n*<=R/2, where R is the maximum row size minus 2 bytes
DBCLOB(n)	Double bytes character large object string of maximum length *n,* with 1<=*n*<=1073741823. *n K* or *n M* or *n G* can be also specified, where K means 1024 bytes, M means 1048576 bytes, G means 1073741824 bytes
DATE	A three part value for year, month and day
TIME	A three part value for hour, minute and second
TIMESTAMP	A seven part value for year, month, day, hour, minute, second and microsecond
ROWID	A value that uniquely identifies a row in a table

Comparison between Oracle and DB2 data types

Table 22 provides a mapping table for data type comparison and conversion between Oracle and DB2.

Table 22. Comparison between Oracle and DB2 data types

Oracle data type	DB2 for OS/390 V6 data type
NUMBER	FLOAT
NUMBER(n)	FLOAT(n). If *n*<16, SMALLINT should be used. If 16<=*n*<=31, INTEGER should be used.
NUMBER(d,p)	DECIMAL(d,p), when *p*<=31. If *p*>31, FLOAT or DOUBLE can be used
DECIMAL(n)	Same as NUMBER(n)
DECIMAL(d,p)	Same as NUMBER(d,p)
FLOAT	FLOAT
FLOAT(n)	FLOAT(n), when n<53. If n>=53, FLOAT or DOUBLE can be used
SMALLINT	SMALLINT or DOUBLE
INTEGER	INTEGER or DOUBLE
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
VARCHAR2(n)	VARCHAR(n)
LONG VARCHAR	CLOB(2 G)
RAW(n)	CHAR(n) FOR BIT DATA
LONG	CLOB(2 G). A ROWID column is also required for LOBs in DB2. If data stored in the source database is shorter, less space can be reserved for DB2 CLOBs. If the effective length of LONG Oracle data is less then the maximum length of a VARCHAR in your DB2 target table, you can also use VARCHAR
LONG RAW	BLOB(2 G). A ROWID column is also required for LOBs in DB2. If data stored in the source database is shorter, less space can be reserved for DB2 BLOBs. If the effective length of LONG RAW Oracle data is less then the maximum length of a VARCHAR in your DB2 target table, you can also use VARCHAR FOR BIT DATA
DATE	TIMESTAMP (but including microseconds)
ROWID	ROWID GENERATED ALWAYS BY DEFAULT. While in Oracle it is a pseudo data type and ROWID column is created automatically at table creation, in DB2 it is a data type that requires an explicit creation in the CREATE/ALTER table statements
MLSLABEL	Not applicable to DB2 for OS/390

DATE and TIME data types

Oracle has the DATE data type. DB2 has DATE, TIME, and TIMESTAMP. The Oracle DATE is equivalent to the DB2 TIMESTAMP data type, but does not include microseconds. In Oracle there is a default external format, depending on the language conventions used for the instance configuration. To extract data in another format, the TO_CHAR function must be used, as in the following example:

```
select TO_CHAR(sysdate,'YYYY-MM-DD-HH24.MI.SS') from DUAL;
```

DB2 uses the TIMESTAMP data type, which includes up to microseconds, and whose ISO standard external format is YYYY-MM-DD-HH24.MI.SS.mmmmmm. DB2 has also the DATE data type for the dates (ISO standard external format: YYYY-MM-DD) and the TIME data type (ISO standard external format: HH.MM.SS).

The formats for string representation of dates in DB2 is shown in Table 23.

Table 23. DB2 format for string representation of DATE data types

Format name	Abbreviation	Date format
ISO standard	ISO	yyyy-mm-dd
IBM USA standard	USA	mm/dd/yyyy
IBM European standard	EUR	dd.mm.yyyy
Japanese industrial standard - Christian Era	JIS	yyyy-mm-dd
Installation defined	LOCAL	depending on installation

From within your application program you have different options between DB2 and Oracle in dealing with date format and its components.

Table 24 maps the equivalent functions.

Table 24. Comparison between Oracle and DB2 DATE components format

Oracle	DB2 for OS/390 V6
SELECT TO CHAR(DATE_FIELD,'DD')	SELECT DAY(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'DAY')	SELECT DAYNAME(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'D')	SELECT DAYOFWEEK(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'DDD')	SELECT DAYOFYEAR(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'HH24')	SELECT HOUR(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'MI')	SELECT MINUTE(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'MM')	SELECT MONTH(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'MONTH')	SELECT MONTHNAME(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'SS')	SELECT SECOND(DATE_FIELD) FROM TABLE
SELECT TO CHAR(DATE_FIELD,'YYYY')	SELECT YEAR(DATE_FIELD) FROM TABLE

The formats for string representation of times in DB2 are shown in Table 25.

Table 25. DB2 format for string representation of TIME data type

Format name	Abbreviation	Date format
ISO standard	ISO	hh.mm.ss
IBM USA standard	USA	hh:mm AM/PM
IBM European standard	EUR	hh.mm.ss
Japanese industrial standard - Christian Era	JIS	hh:mm:ss
Installation defined	LOCAL	depending on installation

With DB2 for OS/390 V6, several sample user-defined functions are provided to manage date and time formats which are different from the standard ones, as shown in Table 26.

Table 26. Sample user-defined functions provided with DB2 for OS/390 V6

Function name	Description
DSN8.ALTDATE	Returns the current date or a user-specified date in a user-specified format
DSN8.ALTTIME	Returns the current time or a user-specified time in a user-specified format
DSN8.DAYNAME	Returns the name of the day of the week on which a date in ISO format falls
DSN8.MONTHNAME	Returns the name of the month in which a date in ISO format falls

Refer to Appendix C, “Sample data preparation program” on page 253 for the configuration and usage steps of these DB2 sample functions, and see *DB2 UDB for OS/390 Version 6 SQL Reference,* SC26-9014 for a description of these and other DB2 user-defined functions. Refer also to 5.2.1.16, “User-defined functions and stored procedures” on page 103.

Numeric data types

Usually the numeric formats cannot be converted just starting from the data type definition of the table, but also, the content of the numeric field should be taken into consideration. For example, the NUMBER(n) Oracle data type has a natural corresponding value in DB2 in FLOAT(n), but in Oracle NUMBER(n) means a simple INTEGER number.

ROWID data type

In Oracle, the ROWID pseudo column is always created with the CREATE TABLE statement. The name of the pseudo-column is ROWID for all the tables.

DB2 for OS/390 V6 supports the ROWID data type, and it can be defined with a CREATE TABLE or ALTER TABLE statement.

A table cannot have more than one ROWID column. It must be specified with the GENERATED ALWAYS or GENERATED BY DEFAULT options. The BY DEFAULT option can be specified only if a unique, single-column index on the ROWID column is created, otherwise the SQL INSERT statement and the LOAD utility cannot be used to add rows to the table. Unless you are using data

propagation, it is recommended to use the ALWAYS clause. the ROWID cast function is provided to access directly ROWID columns.

Moreover, a ROWID column is mandatory when the table contains LOB columns (BLOB, CLOB, DBCLOB).

The internal representation of a row ID value is transparent to the user. The value is never subject to translation because it is considered to contain BIT data. The length of ROWID column, as described in the LENGTH column of catalog table SYSCOLUMNS, is the internal length, which is 17 bytes. The length as described in the LENGTH2 column of DB2's catalog table SYSIBM.SYSCOLUMNS is the external length, which is 40 bytes.

You can see an example of creation of the ROWID column in Figure 28.

```
CREATE TABLE EMPLOYEE (NAME CHAR(30) NOT NULL,
                       DEPT CHAR(10) NOT NULL,
                       ROWID ROWID GENERATED ALWAYS);

INSERT INTO EMPLOYEE (NAME, DEPT) VALUES ('HUGH FANTOZZI','FINANCIAL');

SELECT NAME, ROWID FROM EMPLOYEE;
```

Figure 28. Example of a table with a ROWID column

Large OBjects

The BLOB, CLOB and DBCLOB definition in DB2 for OS/390 V6 requires the definition of a ROWID column in the table containing the large object column(s).

Once the table is been created, some additional steps are also required to access the table.

Refer to 5.2.1.6, "Table definition conversion" on page 75 for a description of the syntax for defining tables containing large object columns.

5.2.1.6 Table definition conversion

The syntax of the CREATE TABLE statement is quite different between Oracle and DB2 for OS/390 V6. The main difference is that in the Oracle CREATE TABLE statement some clauses can be specified, which can modify some physical table definitions (effective at table space level). In DB2 all the physical definitions are always implemented at table space level.

This fact implies that a full conversion of the table definitions should be divided into two steps:

- Analysis of the physical definition of the table and conversion into corresponding table space definitions in DB2 (refer to 5.2.1.3, "Table space definitions" on page 66)
- Migration of the logical definition of the table into a DB2 CREATE TABLE statement

Managing long names

Another difference to take into consideration is that in DB2 table names cannot be longer than 18 characters, while in Oracle, the maximum is 30 characters.

We can retrieve the Oracle table names longer than 18 bytes from the Oracle catalog with the following statement:

```
select table_name, length(table_name) from cat
where length(table_name)>18;
```

The same limitation to 18 characters is valid for column names. You can retrieve the column names longer than 18 characters with the following statement:

```
select table_name, column_name, length(column_name)
from user_tab_columns where length(column_name)>18;
```

A similar command can be used also for the index names, which have the same limit, as shown also in 5.2.1.7, “Indexes and primary keys conversion” on page 80:

```
select index_name, length(index_name) from user_indexes where
length(index_name)>18;
```

As shown in 5.2.1.3, “Table space definitions” on page 66, also the table space names have to be changed if they do not comply with DB2 limitation. The command to find table space names with the same limit is:

```
select tablespace_name, length(tablespace_name) from dba_tablespaces where
length(tablespace_name)>18;
```

Managing numeric and null fields

In some cases there no direct unique matching between Oracle and DB2 data types. An example is provided by NUMBER(n), that can be converted into SMALLINT or INTEGER or FLOAT, depending upon the value of *n*. Refer to 5.2.1.5, “Data types comparison” on page 69.

Furthermore, since primary key fields cannot be NULL, Oracle database automatically changes to NOT NULL the primary key fields when executing the command ALTER TABLE ADD PRIMARY KEY (if this has not been done in the CREATE TABLE statement). This is not true in DB2, you need to add the NOT NULL clause to the primary key fields in the CREATE TABLE statement.

Tables containing large objects

Even if the CIPROS Process and Laboratory segments do not contain LONG or LONG RAW columns, they need a separate treatment.

As shown in 5.2.1.5, “Data types comparison” on page 69, LONG and LONG RAW Oracle columns can be mapped respectively into CLOB and BLOB DB2 objects.

This is not enough on the DB2 side, since a ROWID column is necessary to correctly create the table with large object fields.

After that, the following steps are also needed before being able to access the table:

- Create a table space to store the LOB data with the CREATE LOB TABLESPACE statement
- Create a table to store the LOB data with the CREATE AUXILIARY TABLE statement
- Create a unique index on the auxiliary table

Note that both the table and the auxiliary table must be created in the same database.

Figure 29 shows an example on how to create a table with large objects. Refer also to *DB2 UDB for OS/390 Version 6 SQL Reference,* SC26-9014.

```
 CREATE TABLE CIPROS.LOBTAB (F1 CHAR(255) FOR BIT DATA,
                             F2 CLOB(2 G),
                             F3_ROWID ROWID GENERATED ALWAYS);

CREATE LOB TABLESPACE LOBTABTS
           IN DSNDB04
           LOG NO;

CREATE AUXILIARY TABLE LOBTABAU
IN DSNDB04.LOBTABTS
STORES CIPROS.LOBTAB
COLUMN F2;

CREATE UNIQUE INDEX X_LOBTAB ON LOBTABAU;
```

Figure 29. Example of a table containing large objects

The DUAL pseudo table

The Oracle DUAL pseudo (or dummy) table is used by applications to extract information or data from the database which are not stored in user tables.

Typical examples are the following SQL statements that you can use to select the current date or display a message:

```
select sysdate from dual;
select 'HALLO' from dual;
```

DB2 has the SYSIBM.SYSDUMMY1 table which can be used in similar situations as follows:

```
SELECT CURRENT DATE FROM SYSIBM.SYSDUMMY1;
SELECT 'HALLO' FROM SYSIBM.SYSDUMMY1;
```

Automatic conversion of table definitions
Paying attention to the exceptions (as for large objects columns), the simplest way to migrate the table definition and creation using an automatic procedure is to start from an Oracle export file and process the CREATE TABLE statements with an AIX shell script, which uses *sed* and *awk* UNIX commands (refer also to *DATABASE 2 for AIX Conversion Guide - Oracle 7.1 to DB2 Version 2,* SG24-2567).

The first step is to use the *export* Oracle utility to extract the objects definition.

First, we have created a flat file containing the list of tables belonging to the part of the source application which is the object of our conversion project. This file is useful also in other steps of the conversion.

Second, a parameter file *exp.prf* for the export utility has been created, as shown in Figure 30, containing the instructions for the *exp* Oracle command and the abbreviated list of all the tables involved in the conversion of our application:

```
userid=cipros/cipros
rows=n
constraints=y
grants=y
indexes=y
file=tables.exp
tables=
ANALYSIS_METHOD
.
.
.
VALUE_TYPE
```

Figure 30. Parameter file exp.prf for the Oracle export utility

Then, the *exp* command has been issued as follows:

```
exp parfile=exp.prf
```

We obtain a *tables.exp* file (as indicated in the *exp.prf* file) containing just the table definitions, including indexes, constraints and grants but not the data (as specified in the *exp.prf* file with the option *rows=N*).

Once the export file has been created, we can start processing the file with the shell script ***ddltabs.sh*** provided in A.1, “ddltabs.sh script” on page 191.

This shell script can be issued with the syntax:

```
ddltabs.sh tables.exp > tables.ddl
```

It performs the following operations:

- Selects just the CREATE TABLE statements
- Deletes the PCTFREE/STORAGE clauses from the CREATE TABLE statements
- Converts the column data types, according to the tables described in 5.2.1.5, “Data types comparison” on page 69
- Deletes the double quote symbols and adds a ";" character at the end of each statement
- Changes the old table space names into new ones, according to a list of local environment variables defined at the beginning of the script
- Changes some data types into new ones, according to a list of local environment variables defined at the beginning of the script
- Creates a temporary file containing the list of the table fields which compose the primary key of each table
- Changes the field definition of the primary key fields to NOT NULL, by using the *pk.awk* and the *sednn.sh* external files (see A.1.1, “sednn.sh script” on page 194 and A.1.2, “pk.awk script” on page 194)
- Changes the table names that have to be changed into the new names, according to a list of local environment variables defined at the beginning of the script
- Formats the statements in a more readable layout by inserting new lines after the field definitions and before the table space definition

The substitution of the table space and table names has to be performed in order to comply with the DB2 constraints on table space definitions.

The last step can be useful to change some specific Oracle definition that can be mapped into a different DB2 data type. In our example, we decided to change all the VARCHAR2(1) and VARCHAR2(3) into CHAR and CHAR(3) instead of VARCHAR(1) and VARCHAR(3) respectively.

Moreover, an environment variable DBN is provided at the beginning of the script for the automatic insertion of the DB2 database name as a table space qualifier.

In Figure 31 we show an example of a CREATE TABLE statement with the old Oracle and the new DB2 syntax.

```
Old Oracle CREATE TABLE statement:
CREATE TABLE "LOADING_RACK" ("LOAD_RACK_NAME" VARCHAR2(20),
"LOAD_RACK_TYPE" VARCHAR2(10), "LOAD_PT_NUM" NUMBER(3, 0), "FEED_LINE_DIAM"
NUMBER, "ENG_UNT_FLDIAM" VARCHAR2(10), "DESCRIPTION" VARCHAR2(40),
"PRINT_FNAME" VARCHAR2(8))  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
STORAGE(INITIAL 20480 NEXT 20480 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1) TABLESPACE "CPRS_BASE"

New DB2 CREATE TABLE statement:
CREATE TABLE LOADING_RACK (LOAD_RACK_NAME VARCHAR(20) NOT NULL,
             LOAD_RACK_TYPE VARCHAR(10),
             LOAD_PT_NUM INTEGER,
             FEED_LINE_DIAM FLOAT,
             ENG_UNT_FLDIAM VARCHAR(10),
             DESCRIPTION VARCHAR(40),
             PRINT_FNAME VARCHAR(8))
             IN CIPROS.CPRSBASE;
```

Figure 31. Example of a CREATE TABLE statement

The *ddltabs.sh* script does not perform any change on field names, since CIPROS tables do not contain column names longer than 18 characters. If your application has a few column names that have to be changed, it is easier to do it manually on the DDL file, otherwise you can modify the script by adding a global substitution in the same way it is done with table names.

Table creation on DB2

See B.5, “JCL for creation of storage group, database, table spaces and tables” on page 218. Job CIPROSDB contains the converted DDL as input to DB2’s sample program DSNTIAD. See Figure 32.

```
//*        STEP  2: CREATE CIPROS TABLES, VIEWS
//STEP002 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
 DSN SYSTEM(DB2X)
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -
      LIB('DB2V610X.RUNLIB.LOAD')
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD  *
CREATE TABLE ANALYSIS_METHOD (ANLY_METH_NAME VARCHAR(20) NOT NULL,
             DESCRIPTION VARCHAR(40))
             IN CIPROS.CPRSBASE;
```

Figure 32. Job step with create table

5.2.1.7 Indexes and primary keys conversion

The same approach used for the table conversion can also be used for the indexes.

Syntax differences in index definitions

In this case there are some additional differences in the definition of primary indexes between Oracle and DB2.

With Oracle a primary key can be defined in either one of the following two ways available for specifying different parameters of the ALTER TABLE command:

- ALTER TABLE table_name ADD CONSTRAINT constr_name PRIMARY KEY
- ALTER TABLE table_name ADD PRIMARY KEY

In the first case, the statement automatically creates a unique index on the table with the same definition as the primary key, and it gives to the unique index the same name of the primary key constraint. In the second case, a unique index is created on the table, but a random unique name is given by Oracle to the index.

In DB2 for OS/390 V6 the automatic creation of a primary index with a default name is performed only by the CREATE TABLE statement (and only if the processor is the owner of the schema), while the ALTER TABLE ADD PRIMARY KEY statement creates only the primary key definition.

This leads to the following considerations:

- If the primary key of the Oracle table has been created using the first syntax, we can create both the index (using the constraint name as the index name) and the primary key
- If the primary key of the Oracle table has been created using the second syntax, we can only create the primary key. The corresponding primary index has to be created according to the definition of the primary key fields, by choosing an appropriate naming convention for the new indexes. An example is reported in Figure 33 on page 83.

In both PRIMARY KEY clause types, the fields composing the primary key are automatically set to NOT NULL by Oracle in the table definition. DB2 for OS/390 V6 requires the explicit NOT NULL clause in the CREATE TABLE statement. See also 5.2.1.6, “Table definition conversion” on page 75.

Moreover, a null string and NULL are handled in different ways by DB2, while Oracle considers a null string just the same as the NULL value (see also 7.3.5, “Pointers” on page 154).

Automatic conversion of index definitions

The script ***ddlind.sh*** (see A.2, “ddlind.sh script” on page 194) creates, for each ALTER TABLE ADD CONSTRAINT PRIMARY KEY Oracle statement, two DB2 statements, one for the unique index and the other one for the primary key, and for each ALTER TABLE ADD PRIMARY KEY statement, a DB2 statement only for the primary key.

Starting from the Oracle export file, it prompts to standard output the DDL for the indexes in DB2 syntax.

The script also looks for the ALTER TABLE ADD CONSTRAINT UNIQUE statements and convert them into CREATE UNIQUE INDEX statements.

Finally, it converts the CREATE (UNIQUE) INDEX statements into DB2 syntax.

As for the tables, also the index names have the limit of 18 characters, so the script allows to automatically change the index names. You can retrieve from the Oracle catalog the list of the indexes having long names with the following statement:

```
select index_name, length(index_name) from user_indexes
where length(index_name)>18;
```

When you use the DB2 implicit VSAM definition for indexes, DB2 creates a name for the index space based on the index name. If the index names are very similar, as it can happen coming from a conversion from longer names, you may have problems with the required uniqueness of names for VSAM allocation of the index space (especially if the indexed tables belong to the same database). You might need to modify again the index names.

In DB2 for OS/390 V6, the index maximum length is 255-n, where n=# of NULL columns composing the index. Oracle allows longer indexes (the maximum length is platform-dependent, in our environment it is 1578 bytes).

The shell script can be issued with the syntax:

```
ddlind.sh tables.exp > indexes.ddl
```

This script performs the following operations:

- Selects just the ALTER TABLE ADD CONSTRAINT PRIMARY KEY/UNIQUE and CREATE INDEX statements
- Deletes the INDEX STORAGE clause
- Creates the CREATE INDEX/ALTER TABLE ADD PRIMARY KEY statements in the DB2 syntax
- Deletes the double quote symbols and adds a ";" character at the end of each statement
- Changes the table and index names that have to be changed with the new names, according to a list of local environment variables defined at the beginning of the script
- Formats the statements in a more readable layout by inserting new lines before the field and storage definitions

Moreover, at the beginning of the script some local environment variables are provided, containing the database name, the STOGROUP, the PRIQTY, the SECQTY and the ERASE clauses that should be used for the creation of the DB2 statements. They can be evaluated for the applicability of default values to be used for all the CREATE INDEX statements in the script. Refer to *DB2 UDB for OS/390 Version 6 SQL Reference,* SC26-9014 for a complete description of CREATE INDEX syntax and options.

Every exception should be processed manually, by changing the default values in the final output file. If cluster indexes or specific values for the FREE_BLOCK section or other specific DB2 parameters such as GBPCACHE, BUFFERPOOL or PIECESIZE are needed, they should be analyzed in detail in order to optimize the utilization of DB2 structures and facilities.

In Figure 33, Figure 34 and Figure 35 we show some examples of Oracle ALTER TABLE and CREATE INDEX statements with the old Oracle and the new DB2 syntax.

```
Old Oracle ADD CONSTRAINT PRIMARY KEY statement:
ALTER TABLE "ANALYSIS_METHOD" ADD  CONSTRAINT "ANALYSIS_METHODP1" PRIMARY
KEY ("ANLY_METH_NAME") USING INDEX STORAGE (INITIAL 20480 NEXT 20480
MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 50 FREELISTS 1 ) TABLESPACE
"CPRS_BASE"

New DB2 CREATE UNIQUE INDEX/ADD PRYMARY KEY statements:
CREATE UNIQUE INDEX  ANALYSIS_METHODP1
            ON  ANALYSIS_METHOD
            (ANLY_METH_NAME)
            USING STOGROUP CIPROS01
                  PRIQTY 4000
                  SECQTY 400
                  ERASE NO;

ALTER TABLE  ANALYSIS_METHOD  ADD PRIMARY KEY
            (ANLY_METH_NAME);
```

Figure 33. Example 1 - ALTER TABLE statement

```
Old Oracle CREATE INDEX statement:
CREATE INDEX "SAMPLEI2" ON "SAMPLE" ("MTRL_NAME" )  PCTFREE 10 INITRANS 2
MAXTRANS 255 STORAGE (INITIAL 20480 NEXT 20480 MINEXTENTS 1 MAXEXTENTS
2147483645 PCTINCREASE 50 FREELISTS 1) TABLESPACE "CPRS_BASE"

New DB2 CREATE INDEX statement:
CREATE INDEX SAMPLEI2
            ON SAMPLE
            (MTRL_NAME )
            USING STOGROUP CIPROS01
                  PRIQTY 4000
                  SECQTY 400
                  ERASE NO;
```

Figure 34. Example 2 - CREATE INDEX statement

```
Old Oracle ALTER TABLE ADD CONSTRAINT UNIQUE statement:
ALTER TABLE "MATERIAL" ADD  CONSTRAINT "MATERIALC2" UNIQUE ("MTRL_NUM")
USING INDEX STORAGE (INITIAL 20480 NEXT 20480 MINEXTENTS 1 MAXEXTENTS
2147483645 PCTINCREASE 50 FREELISTS 1 ) TABLESPACE "CPRS_BASE"

New DB2 CREATE UNIQUE INDEX statement:
CREATE UNIQUE INDEX  MATERIALC2
            ON  MATERIAL
            (MTRL_NUM)
            USING STOGROUP CIPROS01
                  PRIQTY 4000
                  SECQTY 400
                  ERASE NO;
```

Figure 35. Example 3 - ALTER TABLE statement

Index creation on DB2
See B.6, “JCL for creation of indexes for CIPROS tables” on page 220. Job CIPROSIX contains the converted DDL as input to DB2’s sample program DSNTIAD. See Figure 36.

```
//*        STEP  1: CREATE CIPROS INDEXES
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DB2X)
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -
      LIB('DB2V610X.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
CREATE UNIQUE INDEX  ANALYSIS_METHODP1
             ON  ANALYSIS_METHOD
             (ANLY_METH_NAME)
             USING STOGROUP CIPROS01
                   PRIQTY 400
                   SECQTY 40
                   ERASE NO;
```

Figure 36. Index creation

5.2.1.8 Foreign keys conversion
Also the definitions of foreign keys can be extracted by the Oracle export file used for tables and indexes. The syntax is very similar in both databases.

The script ***ddlfk.sh*** (see A.3, “ddlfk.sh script” on page 197) converts each ALTER TABLE ADD CONSTRAINT FOREIGN KEY Oracle statement into DB2 syntax.

The shell script can be issued with the syntax:

```
ddlfk.sh tables.exp > fk.ddl
```

The script performs the following operations:

- Selects just the ALTER TABLE ADD CONSTRAINT FOREIGN KEY statements from the Oracle export file
- Converts the syntax of the ALTER TABLE ADD FOREIGN KEY statement
- Deletes the double quote symbols and adds a ";" character at the end of each statement
- Changes the table names that have to be changed with the new names, according to a list of local environment variables defined at the beginning of the script
- Formats the statements in a more readable layout by inserting a new line before the definitions of the FOREIGN KEY and before the REFERENCES clause

In Figure 37 we show an example of a CREATE TABLE statement with the old Oracle and the new DB2 syntax.

```
Old Oracle ALTER TABLE ADD FOREIGN KEY statement:
ALTER TABLE "STREAM_INSTR_ASSIGNMENT" ADD CONSTRAINT
"STREAM_INSTR_ASSIGNMENTF2" FOREIGN KEY ("STREAM_NAME","FACILITY_NAME")
REFERENCES "STREAM" ("STREAM_NAME","FACILITY_NAME")

New DB2 ALTER TABLE ADD FOREIGN KEY statement:
ALTER TABLE STREAM_INSTR_ASG
            FOREIGN KEY (STREAM_NAME,FACILITY_NAME)
            REFERENCES STREAM (STREAM_NAME,FACILITY_NAME);
```

Figure 37. Example of a FOREIGN KEY definition

The self-referenced foreign keys have to be handled manually, since DB2 for OS/390 V6 requires the explicit ON DELETE CASCADE or ON DELETE NO ACTION clause in the ALTER TABLE ADD FOREIGN KEY statement.

The default DELETE rules are normally different between Oracle and DB2 (NO ACTION in Oracle and RESTRICT in DB2) unless otherwise stated in the CURRENT RULES DB2 special register. If the value of the CURRENT RULES is not set explicitly it contains *DB2* and the default is RESTRICT. If the value of CURRENT RULES is set to *SQL*, the default is NO ACTION. Refer to the DELETE rules on *DB2 UDB for OS/390 Version 6 SQL Reference*, SC26-9014 for details.

Foreign key definition on DB2

Refer to B.7, “JCL to alter tables for foreign keys” on page 221 where the job CIPROSFK contains the converted DDL as input to the sample program DSNTIAD. Figure 38 contains an example of FOREIGN KEY creation with ALTER TABLE.

```
//PH01S01 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DB2X)
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -
      LIB('DB2V610X.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *

ALTER TABLE ANALYSIS_SPEC
            FOREIGN KEY (MTRL_NAME)
            REFERENCES MATERIAL (MTRL_NAME);
```

Figure 38. Foreign key creation using ALTER TABLE

5.2.1.9 Authorizations

Most of the authorization GRANT statements on database objects work in DB2 as they are extracted from the table Oracle export. The syntax of the GRANT statement is very similar in both databases.

The script ***ddlgrnt.sh*** (see A.4, “ddlgrnt.sh” on page 198) converts each GRANT Oracle statement into DB2 syntax.

The shell script can be issued with the syntax:

```
ddlgrnt.sh tables.exp > grants.ddl
```

This script performs the following operations:

- Selects just the GRANT statements from the Oracle export file
- Converts the syntax of the GRANT statement
- Deletes the double quote symbols and adds a ";" character at the end of each statement
- Changes the table names that have to be changed with the new names, according to a list of local environment variables defined at the beginning of the script

Since the Oracle users and roles have to be mapped into DB2 for OS/390 users and groups (see 5.2.1.4, “Users, roles, and groups” on page 67), a list of environment variables for the conversion is provided at the beginning of the script.

In Figure 39 we show an example of a GRANT statement with the old Oracle and the new DB2 syntax.

```
Old Oracle GRANT statements:
GRANT DELETE ON "ANALYSIS_SPEC" TO "ROLE_CIPROS"
GRANT INSERT ON "ANALYSIS_SPEC" TO "ROLE_CIPROS"
GRANT SELECT ON "ANALYSIS_SPEC" TO "ROLE_CIPROS"
GRANT UPDATE ON "ANALYSIS_SPEC" TO "ROLE_CIPROS"
GRANT SELECT ON "ANALYSIS_SPEC" TO "ROLE_VIEW"

New DB2 GRANT statements:
GRANT DELETE ON ANALYSIS_SPEC TO PAOLOR2;
GRANT INSERT ON ANALYSIS_SPEC TO PAOLOR2;
GRANT SELECT ON ANALYSIS_SPEC TO PAOLOR2;
GRANT UPDATE ON ANALYSIS_SPEC TO PAOLOR2;
GRANT SELECT ON ANALYSIS_SPEC TO PAOLOR3;
```

Figure 39. Example of a GRANT statement

However, there are several differences especially with administrative authorizations. The administrative authorizations, together with the objects authorizations, can be extracted with an Oracle full export that can be obtained with the following command:

```
exp sys/sys full=y rows=n file=full.exp
```

Then, using the same script *ddlgrnt.sh* as follows:

```
ddlgrnt.sh full.exp > allgrants.ddl
```

It is possible to see and modify, when necessary, the GRANT statements in the *allgrants.ddl* file.

For a list of all the GRANT options, refer to *DB2 UDB for OS/390 Version 6 SQL Reference Version 6,* SC26-9014.

Grant statements on DB2

Refer to B.8, “JCL for synonym creation” on page 221 for examples. Job CIPROSSY contains the converted DDL as input to sample program DSNTIAD. For example, in Figure 40 on page 87 a GRANT is made to allow the CIPROS users to create aliases. CIPROS is then set as the current id. Then a CIPROS alias is created for each table. This allows tables to be accessed with a qualification of CIPROS.*. This is done to limit program changes since the SQL in the CIPROS code is qualified with CIPROS. For example, table ANALYSIS_METHOD is qualified as CIPROS.ANALYSIS_METHOD.

An alternative method is to create all tables with CIPROS as qualifier by assigning DBADM or higher authority to the creator PAOLOR3.

```
//*        STEP  1: CREATE CIPROS SYNONYMS
//PH01S01 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DB2X)
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -
      LIB('DB2V610X.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *

GRANT CREATEALIAS TO CIPROS;
SET CURRENT SQLID = 'CIPROS';
CREATE ALIAS ANALYSIS_METHOD FOR PAOLOR2.ANALYSIS_METHOD;
CREATE ALIAS ANALYSIS_SPEC FOR PAOLOR2.ANALYSIS_SPEC;
```

Figure 40. GRANT and ALIAS creation for CIPROS

Creating plans and packages

B.17, “JCL for binding with use of packages” on page 231 contains sample JCL to bind a plan for use with packages. Packages allow greater flexibility to programmers in complex systems. Each program has a DB2 package associated with it. If an individual program must be changed then only that package needs to have the bind operation performed. If packages are not used then every DBRM associated with a plan would have to be bound when a change occurs.

Grants on being able to execute the plan CIPROS1 are made to public and the batch jobs to run the main programs are able to access the tables. GRANTs to public are satisfactory for our project purposes. However, in any production environment you want to limit executions and accesses only to specific IDs or RACF groups through selective grants. Our GRANTs were issued in DB2 interactive (DB2I) SPUFI option. The GRANTs were coded as in Figure 41.

```
GRANT EXECUTE ON PLAN CIPROS1
      TO PUBLIC;
GRANT CREATE IN COLLECTION CIPROS1
       TO PUBLIC;
GRANT BIND, EXECUTE ON PACKAGE CIPROS1.*
       TO PUBLIC;
```

Figure 41. GRANT examples

Once the plan has been created and GRANTs are executed, binding the DBRM makes the programs code executable in DB2.

5.2.1.10 Check constraints

The CHECK CONSTRAINT statements are extracted with the Oracle export command, too. In DB2 the syntax is similar: ALTER TABLE ADD CHECK or ALTER TABLE ADD CONSTRAINT CHECK.

The script ***ddlchk.sh*** (see A.5, “ddlchk.sh script” on page 200) creates a DDL file for the creation of the check constraints.

Since in the CIPROS applications involved in our conversion project there are no CHECK CONSTRAINT definitions, for test purpose we have created a sample table SPEED_LIMIT as follows (using *SQLPlus* Oracle command line):

```
CREATE TABLE SPEED_LIMIT (ROAD CHAR(30), MILE INTEGER, LIMIT SMALLINT);
ALTER TABLE SPEED_LIMIT ADD CONSTRAINT SL_P1 PRIMARY KEY (ROAD, MILE);
ALTER TABLE SPEED_LIMIT ADD CONSTRAINT SL_C1 CHECK (LIMIT<=65);
ALTER TABLE SPEED_LIMIT ADD CHECK (LIMIT>0);
```

Note that the CHECK constraint can be defined, as in DB2, with or without an explicit constraint name. If a constraint name is not specified, a random unique name is generated by Oracle.

Then, an Oracle export has been issued with the command:

```
exp sys/sys file=speed_limit.exp rows=n tables=speed_limit
```

The shell script can be issued with the syntax:

```
ddlchk.sh speed_limit.exp> chk.ddl
```

This script performs the following operations:

- Select all the ALTER TABLE ... CHECK statements from the input export file
- Deletes all the double quote symbols and adds a ";" character at the end of each statement
- Changes the table names that have to be changed into the new names, according to a list of local environment variables defined at the beginning of the script

In Figure 42 an example of a CHECK constraint creation is shown:

```
Old Oracle ALTER TABLE ADD ... CHECK statements:
ALTER TABLE "SPEED_LIMIT" ADD  CONSTRAINT "SPEED_LIMITC1" CHECK (limit<=65)
ALTER TABLE "SPEED_LIMIT" ADD  CHECK (limit>=0)

New DB2 ALTER TABLE ADD ... CHECK statements:
ALTER TABLE SPEED_LIMIT ADD  CONSTRAINT SPEED_LIMITC1 CHECK (limit<=65);
ALTER TABLE SPEED_LIMIT ADD  CHECK (limit>=0);
```

Figure 42. Example of a CHECK CONSTRAINT definition

5.2.1.11 Synonyms and aliases

There are some differences between Oracle and DB2 in the way synonyms and aliases are used.

Oracle database uses only synonyms to create an alternative name for a database object. DB2 has both synonyms and aliases.

With Oracle synonyms can be either PRIVATE or PUBLIC. A PRIVATE synonym can be created by each user only for directly owned objects. A PUBLIC synonym can be created only by a DBA and is available to all users.

For example, if the table USER_NAME.MY_TABLE_WITH_LONG_NAME is often accessed by the USER_NAME user, the user can define a private synonym MYTAB for that table with the command:

```
CREATE SYNONYM MYTAB FOR USER_NAME.MY_TABLE_WITH_LONG_NAME;
```

If the user is also a DBA and the table is also accessed by other users, the user can create a public synonym for that table as follows:

```
CREATE PUBLIC SYNONYM MYTAB FOR USER_NAME.MY_TABLE_WITH_LONG_NAME;
```

Differences between Oracle and DB2 synonyms

The concept of SYNONYM in DB2 is similar to a PRIVATE SYNONYM in Oracle. To map a PUBLIC SYNONYM in Oracle a DB2 ALIAS can be defined with the CREATE ALIAS command.

In DB2 a SYNONYM can be created by all the users and it is local to the owner's schema, while an ALIAS can be created only by a SYSADM user, a SYSCTL user, or a user with a CREATE ALIAS privilege, and it is public to all users.

An ALIAS, in DB2 for OS/390 V6, is always a full-qualified name, so it can be referenced by all users only together with the creation schema. In our project we have created a CIPROS schema, and all the alias have been created as CIPROS.table_name.

Another difference between Oracle and DB2 is that in Oracle a synonym can be created even if the corresponding objects do not exist, and the deletion of the reference object does not delete the synonym. In DB2 for OS/390 V6 this is true only for the aliases. Differently from Oracle, if a table is dropped, all its synonyms are deleted also. In DB2, if an alias is created with the referenced object not existent, a warning message is issued.

The syntax of the CREATE SYNONYM in DB2 is the same as in Oracle. An example of the CREATE ALIAS command is the following:

```
CREATE ALIAS LOCTABLES FOR DB2SOMEWHERE.SYSIBM.TABLES;
```

where DB2SOMEWHERE is the location name of a DB2.

Let us clarify the difference between aliases and synonyms in DB2 for OS/390:

- SYSADM or SYSCTRL authority or the CREATE ALIAS privilege is required to define an alias. No authorization is required to define a synonym.
- An alias can be defined on the name of a table or view, including tables and views that are not at the current server. A synonym can only be defined on the name of a table or view at the current server.
- An alias can be defined on an undefined name. A synonym can only be defined on the name of an existing table or view.
- Dropping a table or view has no effect on its aliases. But dropping a table or view does drop its synonyms.
- An alias is a qualified name that can be used by any authorization ID. A synonym is an unqualified name that can only be used by the authorization ID that created it.
- An alias defined at one DB2 subsystem can be used at another DB2 subsystem when connected through DRDA. A synonym can only be used at the DB2 subsystem where it is defined.
- When an alias is used, an error occurs if the name that it designates is undefined or is the name of an alias at the current server. (The alias can designate an alias defined at another server if that alias represents a table or view at the other server.) When a synonym is used, this error cannot occur.

Automatic conversion of synonym definitions

The private and public synonyms are not extracted and written in a simple Oracle export. Their definition is only extracted in a full export that can be obtained with the following command:

```
exp sys/sys full=y rows=n file=full.exp
```

Unfortunately, with a full export all the synonyms defined in the database are exported in the output file.

For this reason it is better to start directly from the Oracle catalog tables SYNONYMS (public synonyms) and USER_SYNONYMS (private synonyms) to extract only what we need to migrate.

In our case, since the database area is well defined (we have created a file containing the list of the tables) and they belong to a single Oracle user (*cipros*), we can use two simple scripts ***ddlalias.sh*** and ***ddlsyn.sh*** (see A.6, “ddlalias.sh script” on page 201 and A.7, “ddlsyn.sh script” on page 202) to create two DDL files for the creation of the synonyms and the aliases used by the application.

Since the CIPROS application does not use private synonyms, in this section we analyze only the alias creation and the related *ddlalias.sh* script (the *ddlsyn.sh* script is very similar).

The shell script can be issued with the syntax:

```
ddlalias.sh tables.lst > alias.ddl
```

This script performs the following operations:

- Creates an SQL file containing the SELECT statement for the Oracle SYNONYMS table, using the input file tables.lst as the list of the PUBLIC synonyms that should be created. This SELECT statement creates at the same time the CREATE ALIAS DB2 statement for the DB2
- Creates and applies a *sed* script file to handle the last line of the SQL file
- Executes the *SQLplus* Oracle command line using the SQL file as a command file
- Adds a ";" character at the end of each statement
- Writes to the output file, as the first line, the SET CURRENT SQLID=CIPROS statement
- Changes the table names that have to be changed with the new names, according to a list of local environment variables defined at the beginning of the script

5.2.1.12 Views

The view definitions in Oracle and DB2 may contain some differences that have to be handled manually. For instance, a view definition may contain in the body of the subquery functions that are different between Oracle and DB2 (refer to 5.2.1.14, “Standard operators and functions” on page 94).

DB2 cannot create a view on a non-existing object (this can be done in Oracle with the FORCE option, a warning is returned by the database) and cannot replace an existing view (in Oracle this can be obtained with the CREATE OR REPLACE VIEW command).

The CHECK OPTION clause in Oracle can only be local. In DB2 for OS/390 V6 a WITH CASCADE CHECK OPTION or WITH LOCAL CHECK OPTION clause can be specified. Refer to *DB2 UDB for OS/390 Version 6 SQL Reference,* SC26-9014 for a complete description of the CREATE VIEW statement.

The view definitions can be extracted by a full Oracle export.

The script ***ddlview.sh*** (see A.8, “ddlview.sh script” on page 204) selects from an input file all the CREATE VIEW statements.

The shell script can be issued with the syntax:

```
ddlview.sh full.exp > views.ddl
```

Then, the *views.ddl* file can be edited to modify the statements according to the DB2 syntax.

In Figure 43 we show an example of a view definition using the Oracle DECODE function. In Figure 44 the corresponding DB2 CREATE VIEW command (using the CASE expression) is listed.

Note that in the two statements we use the catalog structures for the table authorizations (ALL_TAB_PRIVS for Oracle, SYSTABAUTH in DB2), which contain the same information but in different formats. Note also that the USER register is used to retrieve the authorizations of the user connected to DB2. In Oracle the ALL_TAB_PRIVS catalog view already contains only the information related to the user currently logged on.

```
CREATE VIEW TABLES_PERMS (TABLE_NAME, ACCESS_STRING) AS
SELECT
T.TABLE_NAME,
DECODE(PRIVILEGE,'SELECT','-','S') ||
DECODE(PRIVILEGE,'UPDATE','-','U') ||
DECODE(PRIVILEGE,'INSERT','-','I') ||
DECODE(PRIVILEGE,'DELETE','-','D') ACCESS_STRING
FROM
ALL_TAB_PRIVS T, ALL_OBJECTS O
WHERE
T.TABLE_NAME = O.OBJECT_NAME
AND
O.OBJECT_TYPE IN ('TABLE','VIEW')
AND
T.TABLE_SCHEMA='CIPROS';
```

Figure 43. Usage of CREATE VIEW with an Oracle DECODE instruction

```
CREATE VIEW TABLES_PERMS (TABLE_NAME,ACCESS_STRING) AS
SELECT TTNAME,
CASE SELECTAUTH WHEN 'Y' THEN 'S'
                WHEN 'G' THEN 'S' ELSE '-' END ||
CASE UPDATEAUTH WHEN 'Y' THEN 'U'
                WHEN 'G' THEN 'U' ELSE '-' END ||
CASE INSERTAUTH WHEN 'Y' THEN 'I'
                WHEN 'G' THEN 'I' ELSE '-' END ||
CASE DELETEAUTH WHEN 'Y' THEN 'D'
                WHEN 'G' THEN 'D' ELSE '-' end
FROM SYSIBM.SYSTABAUTH
WHERE TCREATOR='CIPROS' AND GRANTEE=USER;
```

Figure 44. Usage of CREATE VIEW with a DB2 CASE instruction

Views definition in DB2

Refer to B.9, “JCL for creation of CIPROS views” on page 222. Job CIPROSVW contains the converted DDL as input to sample program DSNTIAD. Figure 45 provides an example of view definition in DB2.

```
//*        STEP  1: CREATE CIPROS VIEWS                                  00005400
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                      00005500
//SYSTSPRT DD SYSOUT=*                                                   00005600
//SYSTSIN  DD *                                                          00005700
 DSN SYSTEM(DB2X)                                                       00005800
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -                                 00005900
      LIB('DB2V610X.RUNLIB.LOAD')                                       00006000
//SYSPRINT DD SYSOUT=*                                                   00006100
//SYSUDUMP DD SYSOUT=*                                                   00006200
//SYSIN    DD *                                                          00006300
CREATE VIEW CASES_GLOBAL
(CASE_NAME,
LOCATION,
CASE_OWNER,
CASE_TITLE,
DESCRIPTION,
PERIOD_NAME,
START_DATE,
END_DATE,
CASE_TIME_RES,
DATE_CHANGE,
DATA_STORED) AS
SELECT CASE_NAME,
LOCATION,
CASE_OWNER,
CASE_TITLE,
DESCRIPTION,
PERIOD_NAME,
START_DATE,
END_DATE,
CASE_TIME_RES,
TIMESTAMP_CHANGE,
DATA_STORED FROM CASES WHERE LOCATION = 'GLOBAL';
```

Figure 45. CIPROS view definition

5.2.1.13 Triggers

With DB2 for OS/390 V6, the syntax is similar to Oracle, but there are differences that may need to be addressed when migrating to DB2, refer to *ORACLE 7, The Complete Reference,* ISBN 0-07-882285-8 and *DB2 UDB for OS/390 SQL Reference Version 6,* SC26-9014.

The major differences are:

- With Oracle, the CREATE or REPLACE command can be specified while creating a trigger. In DB2, we have to DROP and then CREATE the trigger again.
- In Oracle, the trigger body is a PL/SQL block (with no DDL, ROLLBACK or COMMIT statements), in DB2 we can only specify a list of statements. If you need to migrate a complex PL/SQL trigger, you might want to use a stored procedure as the action of the trigger, written in a standard language like C or COBOL, or with the new SQL Procedures language (see 5.2.1.16, “User-defined functions and stored procedures” on page 103).
- With Oracle, you can link together the before and after actions with the OR boolean conjunction, in DB2 only one before and after action at a time is supported for each trigger.

- In DB2, when defining a trigger, you must specify it as either a FOR EACH ROW or a FOR EACH STATEMENT trigger. Oracle defaults to FOR EACH STATEMENT if nothing is specified. Another difference is that Oracle allows the FOR EACH STATEMENT to be specified on BEFORE triggers; this is not allowed in DB2 for OS/390 V6.
- In DB2, the NO CASCADE BEFORE clause must be specified for before triggers. It implies that the trigger does not allow other triggers to be activated.
- DB2 allows you to use the REFERENCING clause to specify correlation names for the rows modified by the trigger (both the old and the new values), while Oracle use the *old* and *new* keywords. In addition, with DB2, you can reference the transient tables containing all the affected rows before and after the trigger modified them.
- DB2 allows the WHEN clause with statement-level triggers, Oracle does not.

You can see an example of a DB2 trigger in Figure 46 on page 105. We have used SPUFI to run the single statements.

5.2.1.14 Standard operators and functions

Most of the Oracle built-in operators and functions can be mapped into DB2 ones. For a detailed description of the DB2 operator and functions refer to *DB2 UDB for OS/390 - SQL Reference Version 6,* SC26-9014.

Table 27 shows, for all the Oracle built-in operators, the corresponding item and the possible differences with the DB2 for OS/390 V6 equivalent operator.

Table 27. Comparison between Oracle and DB2 operators

Oracle operator	DB2 operator	Notes
+	+	
-	-	
*	*	
/	/	DB2 default is to truncate the result to the lower integer, Oracle default (in *SQLPlus* command line) is decimal value with 8 digits after decimal point. Use DOUBLE or DECIMAL function to cast the operators and then the result to the desired output.
||, CONCAT	||, CONCAT	- CONCAT function is preferable because the || operator can be wrongly parsed in some countries or in particular statements - Oracle performs an implicit data type conversion, DB2 concatenates only homogeneous and compatible data types
=	=	
!, ^	^	If possible, an alternate operator is preferable to the ^ (not) operator, since it can be wrongly parsed in some countries or in particular statements. For example, substitute '<>' for '^=', '<=' for '^>', and '>=' for '^<'
!=, ^=, <>	^=, <>	In Oracle, not all the forms are available on all platforms. In DB2, <> is preferable (see above)

Oracle operator	DB2 operator	Notes
>, <, >=, <=	>, <, >=, <=	
NOT	NOT	
AND	AND	
OR	OR	
ANY	ANY	Usable in DB2 only with subqueries, not with lists. With lists, the OR operator can be used. With =ANY (list), use IN
ALL	ALL	Usable in DB2 only with subqueries, not with lists. With lists, the AND operator can be used
IN, = ANY	IN, = ANY	= ANY can be used only with subqueries, not with lists. With lists, the OR operator can be used
NOT IN, != ALL	NOT IN, ^=ALL	^= ALL can be used only with subqueries, not with lists. With lists, NOT IN or the AND operator can be used
[NOT] BETWEEN	[NOT] BETWEEN	
[NOT] EXISTS	[NOT] EXISTS	
[NOT] LIKE [ESCAPE]	[NOT] LIKE [ESCAPE]	
IS [NOT] NULL	IS [NOT] NULL	
UNION [ALL]	UNION [ALL]	
INTERSECT		Not available in DB2. Use WHERE ... IN (...) or WHERE EXISTS (...)
MINUS		Not available in DB2. Use WHERE ... NOT IN (...) or WHERE NOT EXISTS (...)
(+) (outer join)	... OUTER JOIN ... ON ...	Specify the OUTER JOIN command for the outer joins.

Table 28 shows, for all the Oracle built-in functions, the correspondent item and the possible differences with the DB2 for OS/390 V6 equivalent function.

Table 28. Comparison between Oracle and DB2 functions

Oracle function	DB2 function	Notes
ABS	ABS, ABSVAL	
ACOS	ACOS	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
ADD_MONTHS		Not available. You can use the date/time DB2 arithmetic. For example: - in Oracle: SELECT ADD_MONTHS(hiredate,1) from EMPLOYEE - In DB2: SELECT hiredate + 1 MONTHS from EMPLOYEE
ASCII		Not available
ASIN	ASIN	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
ATAN	ATAN	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
ATAN2	ATAN2	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
AVG	AVG	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format.
CEIL	CEIL	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
CHARTOROWID	ROWID	Same syntax but different external formats for the ROWID pseudo-column, since DB2 uses external hexadecimal representation for ROWID columns
CHR		Not available
CONVERT		Not available
COS	COS	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
COSH	COSH	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
COUNT	COUNT	The explicit ALL argument is not supported (it is the default with the * argument)
DECODE	CASE	Different syntax. See the examples in Table 43 on page 92 and Table 44 on page 92
DUMP		Not available

Oracle function	DB2 function	Notes
EXP	EXP	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
FLOOR	FLOOR	
GREATEST		Not available
GREATEST_LB		Not available
HEXTORAW	X	Either use X'xxxx' to represent an hexadecimal value (for FOR BIT DATA columns, for example) or, if needed, cast functions (GRAPHIC, VARGRAPHIC) to translate hexadecimal values into graphic binary data
INITCAP		Not available. Use TRANSLATE function
INSTR	POSSTR	Oracle syntax is INSTR(*str1,str2,n,m*) where n is the starting position in *str1* and *m* is the m^{th} occurrence to be positioned. DB2 POSSTR function corresponds to INSTR(*str1,str2,1,1*)
INSTRB		Not available. The POSSTR command has to be used. Note that special rules are used in DB2 for mixed and double-byte strings (refer to *DB2 UDB for OS/390 Version 6 SQL Reference,* SC26-9014 for a complete description of the POSSTR function)
LAST DAY		Not available. You can use the DB2 built-in functions and date arithmetic to obtain the last day of a month.
LEAST		Not available
LEAST_LB		Not available
LENGTH	LENGTH	Oracle returns the effective data length, DB2 returns, except the VARCHAR fields, the maximum field length
LENGTHB		Not available. The LENGTH function performs a byte-count operation on the input argument, unless for GRAPHIC data types, for which the double-byte length is returned
LN(x)	LN, LOG	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
LOG(x,y)		Not available. You can either use LOG(y)/LOG(x) or LOG10(y)/LOG10(x)
LOWER	LOWER, LCASE	
LPAD		Not available. Use INSERT, LEFT, LOCATE, SPACE, REPEAT, POSSTR, REPLACE functions

Oracle function	DB2 function	Notes
LTRIM	LTRIM	With DB2 you can just remove the leading blanks. Use STRIP function to remove any type of character. Oracle LTRIM allows to specify a substitution string, with DB2 you can only specify a substitution single character. To perform a string substitution, you can use nested STRIP functions. For example: - Oracle: LTRIM('abcd','ab') - DB2: STRIP(STRIP('abcd',L,'a'),L,'b') Note that the result is the same only if you have only one occurrence of the substitution characters
MAX	MAX	
MIN	MIN	
MOD	MOD	
MONTHS_ BETWEEN		Not available. You can use the date/time DB2 arithmetic to build a UDF performing the same operation.
NEW_TIME		Not available.
NEXT_DAY		Not available. You can use the date/time DB2 arithmetic to build a UDF performing the same operation.
NLS_INITCAP		Not available. Use TRANSLATE function with SET CURRENT LOCALE LC_CTYPE command
NLS_LOWER		Not available. Use LOWER or LCASE functions with SET CURRENT LOCALE LC_CTYPE command
NLS_UPPER		Not available. Use UPPER or UCASE functions with SET CURRENT LOCALE LC_CTYPE command
NLSSORT		Not available. SET CURRENT LOCALE LC_CTYPE command before querying the database
NVL	NULLIF	
POWER	POWER	
RAWTOHEX		Use HEX function.
REPLACE	REPLACE	
ROUND	ROUND	The syntax is the same: ROUND(n,m). In DB2 m is not optional, as it is in Oracle. Moreover, Oracle allows the user to input dates to the ROUND function.
ROWIDTOCHAR	CHAR	DB2 CHAR function can convert ROWID columns to the external hexadecimal representation.
RPAD		Not available. Use INSERT, RIGHT, LOCATE, REPEAT, SPACE, POSSTR, REPLACE functions

Oracle function	DB2 function	Notes
RTRIM	RTRIM	With DB2 you can just remove the trailing blanks. Use STRIP function to remove any type of character. Oracle LTRIM allows to specify a substitution string, with DB2 you can only specify a substitution single character. To perform a string substitution, you can use nested STRIP functions.For example: - Oracle: RTRIM('abcd','cd') - DB2: STRIP(STRIP('abcd',T,'c'),T,'d') Note that the result is the same only if you have only one occurrence of the substitution characters.
SIGN	SIGN	
SIN	SIN	
SINH	SINH	
SOUNDEX		Not available
SQRT	SQRT	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
STDDEV	STDDEV	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
SUBSTR	SUBSTR	The DB2 SUBSTR function operates on a byte-count basis. If the input string is a mixed or double-byte string, the result will not necessarily be a properly formed mixed or double-byte data string
SUBSTRB		Not available. See SUBSTR
SUM	SUM	The explicit ALL argument is not supported (it is the default)
SYSDATE	CURRENT DATE	
TAN	TAN	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
TANH	TANH	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
TO_BINARY_INTEGER		
TO_CHAR(date)	CHAR	The CHAR cast function can be used to convert dates/times/timestamps strings from the installation format into another format. The valid formats are listed in Table 23 on page 73 and Table 25 on page 74
TO_CHAR(label)		Not available
TO_CHAR(num)	CHAR	WIth CHAR DB2 function, the output number format can not be specified

Oracle function	DB2 function	Notes
TO_DATE		DB2 provides a lot of functions to manipulate date-time columns. Refer to Table 26 on page 74 and to
TO_LABEL		Not available
TO_MULTI_ BYTE		Not available. Cast functions GRAPHIC, VARGRAPHIC or BLOB can be used to convert into double-byte columns
TO_NUMBER	SMALLINT, INT[EGER], FLOAT, DOUBLE, DEC[IMAL],	WIth DB2 functions, the input number format can not be specified
TO_SINGLE_ BYTE		Not available. Cast functions CHAR or CLOB can be used to convert into double-byte columns
TO_VARCHAR2		Not available
TRANSLATE	TRANSLATE	
TRUNC	TRUNC[ATE]	Not to be confused with TRUNCATE Oracle command. For numeric operands, the behavior and the considerations are similar to the ROUND functions. Moreover, as for the ROUND function, Oracle allows the user to input dates to the TRUNC function.
UPPER	UPPER, UCASE	
USER	USER	
USERENV		Not available
VARIANCE	VARIANCE	Different default output formatting. Use INT, CEIL, FLOOR, DECIMAL, FLOAT DB2 functions to obtain the desired output format
VSIZE	LENGTH	In DB2 the internal length, determined by the data type, is prompted

Due to the different storage management, data type definitions and data access, DB2 owns several additional built-in functions, as listed in Table 29. Some of them have been already introduced in Table 28 on page 96, since in some cases they can be used to map Oracle functions. For a detailed description and usage of these functions, refer to *DB2 UDB for OS/390 Version 6 SQL Reference,* SC26-9014.

Table 29. Additional DB2 functions

Additional DB2 for OS/390 V6 functions	Description
BLOB	Returns a BLOB representation of its argument
CHAR	Returns a fixed-length character string representation of its arguments
CLOB	Returns a CLOB representation of its argument
COALESCE	Returns the first argument in a set of arguments that is not null
COUNT_BIG	Same as COUNT, except the result can be greater than the maximum value of an integer
DATE	Returns a date derived from its argument
DAY	Returns the day part of its argument
DAYOFMONTH	Similar to DAY
DAYOFWEEK	Returns an integer in the range of 1 to 7, where 1 represents Sunday
DAYOFYEAR	Returns an integer in the range of 1 to 366, where 1 represents January 1
DAYS	Returns an integer representation of a date
DBCLOB	Returns a DBCLOB representation of its argument
DEC[IMAL]	Returns a decimal representation of its argument
DEGREES	Returns the number of degrees for an argument that is expressed in radians
DIGITS	Returns a character string representation of a number
DOUBLE [PRECISION]	Returns a double precision floating-point representation of its argument
FLOAT	Same as DOUBLE
GRAPHIC	Returns a GRAPHIC representation of its argument
HEX	Returns a HEX representation of its argument
HOUR	Returns the hour part of its argument
IFNULL	Returns the first argument in a set of two arguments that is not null
INSERT	Returns a string that is composed of an argument inserted into another argument at the same position where some number of bytes have been deleted
INT[EGER]	Returns an integer representation of its argument
JULIAN_DAY	Returns an integer that represents the number of days from January 1, 4712 B.C.
LEFT	Returns a string that consists of the specified number of left-most bytes of a string

Additional DB2 for OS/390 V6 functions	Description
LOCATE	Returns the starting position of the first occurrence of a string within another string optionally passing the position from where to start the search
LOG10	Returns the base 10 logarithm of an argument
MICROSECONDS	Returns the microsecond part of its argument
MIDNIGHT_SECONDS	Returns an integer in the range of 0 to 86400 that represents the number of seconds between midnight and the argument
MINUTE	Returns the minute part of its argument
MONTH	Returns the month part of its argument
POSSTR	Returns the starting position of the first occurrence of a string within another string searching from the beginning
QUARTER	Returns an integer in the range of 1 to 4 that represents the quarter of the year for the date specified in the argument
RADIANS	Returns the number of radians for an argument that is expressed in degrees
RAISE_ERROR	Raises an error in the SQLCA with the specified SQLSTATE and error description
RAND	Returns a double precision floating-point random number
REAL	Returns a single precision floating-point representation of its argument
REPEAT	Returns a character string composed of an argument repeated a specified number of times
RIGHT	Returns a string that consists of the specified number of right-most bytes of a string
ROWID	Returns a row ID representation of its argument
SECOND	Returns the second part of its argument
SMALLINT	Returns a small integer representation of its argument
SPACE	Returns a string that consists of the number of blanks the argument specifies
STRIP	Returns the characters of a string with the blanks (or specified character) at the beginning, end, or both beginning and end of the string removed
TIME	Returns a time derived from its argument
VALUE	Same as COALESCE
VARCHAR	Returns the varying-length character string representation of its argument
VARGRAPHIC	Returns the graphic string representation of its argument
WEEK	Returns an integer that represents the week of the year
YEAR	Returns the year part of its argument

5.2.1.15 Distinct user-defined types

DB2 offers the possibility to create user-defined types (UDT). They can be useful to create types corresponding to missing Oracle data types, or to create your own types that more closely match application needs and the values being stored.

We show in the following example how to create a UDT:

```
CREATE DISTINCT TYPE SPEED AS INTEGER WITH COMPARISONS
```

UDTs are strongly typed. This means that a UDT, such as SPEED, cannot be compared to an INTEGER, even though it is based on the integer type. To do such a comparison, two casting functions (SPEED(integer) and INTEGER(speed) are automatically generated during the distinct type creation. Moreover, the comparison operators (=,<,>,^ and so on) for the new type are created.

For further information on user-defined types, refer to *DB2 UDB for OS/390 Version 6 SQL Reference,* SC26-9014.

5.2.1.16 User-defined functions and stored procedures

Both user-defined functions (UDFs) and stored procedures are available in Oracle and DB2 and are defined with CREATE FUNCTION and CREATE PROCEDURE statements.

Up to DB2 Version 5, the difference consisted in the language used to create them: PL/SQL in Oracle, an external language (ASSEMBLER, C/C++, COBOL, PLI) in DB2. This implied the re-writing of Oracle PL/SQL functions and procedures into one of the languages listed above and then the use of SQL CALL statement.

With DB2 for OS/390 V6 and V5 at September 1999 refresh level, a new language called ***SQL Procedures*** can be used to create stored procedures.It provides functions comparable to PL/SQL Oracle procedure language, and it offers a lot functions that can be useful especially in conversion projects from other databases:

- DB2 stored procedures can be written entirely in SQL Procedures language
- SQL Procedures language is an ISO extension of SQL defined by the SQL Persistent Storage Module (PSM) standard
- SQL Procedures language is standard across the DB2 family of products and can be used to access local and remote DB2 databases

In addition, the DB2 Software Development Kit (SDK), distributed with the products belonging to the DB2 family, includes a new tool, the ***DB2 Stored Procedure Builder*** (SPB). SPB provides a graphical user interface for developing in Java or SQL Procedures language across the DB2 servers in the various platforms. It runs either stand-alone or embedded in most of developing graphical environments, such as Microsoft Visual Basic, Microsoft Visual Studio, IBM VisualAge for Java, on Windows 95-98-NT operating systems.

For details and examples on coding DB2 stored procedures utilizing these two new functions, refer to the very recent redbook *Developing Cross-Platform DB2 Stored Procedures: SQL Procedures and the DB2 Stored Procedure Builder*, SG24-5485.

See also 7.5.1.3, “Stored procedures” on page 175.

5.2.1.17 Packages

In Oracle, a package is a collection of procedures, functions, variables, constants, exceptions, and cursors. They allow multiple procedures to use same objects, such as variables and cursors.

The *package* specifies the functions and the procedures used and their parameters (it is the *declaration* of the package).

The *package body* specifies the PL/SQL to be executed when the package function or the package procedure is executed by an application (it is the *source* of the package).

In DB2, the term *package* refers to a database object that includes information required to execute SQL statements associated with the source file of an application program. A package is generated by pre-compiling a source file or by binding a precompiler-generated bind file using the BIND command.

For these reasons, the CREATE PACKAGE statement is not available on DB2. The Oracle packages should be converted into stand-alone functions or procedures, using the CREATE PROCEDURE statement for each one of the Oracle package procedures.

5.2.1.18 Sequences

Sequences are used by some Oracle applications to assign unique numbers automatically. For instance, we can create a table EMPLOYEE and a sequence EMPLOYEE_ID and then use the sequence to insert a new record in the table, as described with the following commands:

```
create table EMPLOYEE (name VARCHAR2(30) NOT NULL,
                       location VARCHAR2(20),
                       dep VARCHAR2(20),
                       id NUMBER(5) NOT NULL);

create sequence EMPLOYEE_ID increment by 1 start with 1;

insert into EMPLOYEE (name, location, dep, id) values
('ROMINA CORDELLA','BUILDING M-93','FINANCIAL',EMPLOYEE_ID.NextVal);
```

In DB2 for OS/390 V6 the CREATE SEQUENCE command is not available. A user-defined function, associated to a before-insert trigger, can be used to obtain the same result. If NEXT_EMP_ID is the user-defined function which calculates and returns the next available value of *ID* field in *EMPLOYEE* table, the trigger can be created as in Figure 46.

```
create table EMPLOYEE (name VARCHAR(30) NOT NULL,
                       location VARCHAR(20),
                       dep VARCHAR(20),
                       id INTEGER NOT NULL);

CREATE TRIGGER EMP_TRIG NO CASCADE BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AS NEWROW
FOR EACH ROW MODE DB2SQL
SET NEWROW.ID = NEXT_EMP_ID();
```

Figure 46. Example of a trigger to create a sequence number

The NEXT_EMP_ID function should contain a CASE statement similar to the following:

```
CASE
WHEN (SELECT COUNT(ID) FROM EMPLOYEE) = 0 THEN 1
ELSE (SELECT MAX(ID) + 1 FROM EMPLOYEE)
END
```

If you are inserting more than one row at the same time, this trigger will insert the same value for all the inserted rows, unless the SCRATCHPAD option in the user-defined function is used. Refer to *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004 for further information.

5.3 Data layout

The cross relating of the source and target data sources through data layout is a once only task discussed in 3.3.1.5, “Data layout” on page 36 that includes the following tasks:

- Source tables
- Source columns
- Target tables
- Target columns

5.3.1 Source tables

As explained in 5.2.1.6, “Table definition conversion” on page 75, the main differences between Oracle and DB2 in the definition of tables are related to the following items:

- Table names length
- Column names length
- Index names length
- Foreign key names length
- NULL definition for primary key fields

These differences do not affect the migration path of data from Oracle to DB2 tables. In any case, an inventory of the source tables and their possible differences from DB2 requirements is highly recommended, before starting any automatic conversion/transcodification procedures.

Refer to 5.2.1.6, “Table definition conversion” on page 75 for the migration path of the table definitions and to Chapter 6, “Data conversion” on page 109 for the conversion of data.

5.3.2 Source columns

In the conversion of table definitions, column data types have to be converted into DB2 valid data types, in order to maintain the structure and the contents of source data in the target environment.

As shown in 5.2.1.5, “Data types comparison” on page 69, each Oracle data type has a DB2 corresponding data type.

From a data point of view, some problems can arise for the following situations:

- NUMBER: source data out of DB2 range and precision
- LONG: temporary space for the data download/load; bad performance; data conversion; limitation on *SQLPlus* interface
- LONG RAW: temporary space for the data download/load; bad performance; data conversion; limitation on *SQLPlus* interface

Refer to Chapter 6, “Data conversion” on page 109 for the detailed path used for the conversion of data.

5.3.3 Target tables

Once you have successfully migrated their definitions, target tables can be loaded with source data.

We used two data migration methods:

- Usage of data flat files to download data from Oracle; after transferring them (using *ftp)* to the S/390 where DB2 is installed, load them into DB2 tables with the LOAD utility
- Usage of IBM DataJoiner for AIX to access simultaneously both Oracle and DB2 tables

In the first case, we have created, using *SQLPlus*, one file for each table. The data file layout has been determined starting from the table definition. In the second case, once two nicknames for each table have been created (one for the Oracle table, the other for the DB2 table), a direct *insert/select* statement between the two tables can be used. Refer to Chapter 6, “Data conversion” on page 109 for a detailed description of the used methods.

5.3.4 Target columns

In general, non-binary, non-long and non-varying character fields can be simply migrated using the first method described in 5.3.3, “Target tables” on page 106.

The other data types have to be handled with specific actions. Some of the possible migration paths are described in Chapter 6., “Data conversion” on page 109.

5.4 Cross reference

A cross reference of fields to source tables and columns to target tables and columns needs to be built so that any field reference can be changed to the equivalent in DB2. Input for the cross reference is from 3.3.1.4, “Inventory” on page 35 and 3.3.1.5, “Data layout” on page 36.

Oracle and DB2 feature incompatibilities

The major differences between the two databases can be summarized as related to the following:

- Name limit
- Views
- Triggers
- UDT and UDF

Chapter 6. Data conversion

The data conversion implementation plan in 3.3.3.1, “Data conversion plan” on page 37 details how this conversion takes place, with what tools and resources, from the methodology point of view.

In this chapter we describe the contents of the tasks included in our data conversion plan:

- Clean data
- Unload data from Oracle
- Create programs to prepare data for file transfer and DB2 format
- Create partitioned data sets on OS/390 for the data
- Transfer data from the AIX system to the OS/390 system
- Reformat data for DB2
- Test data for correct format
- Load data into DB2

Most of the considerations reported in this chapter on DB2 are meant for inexperienced users.

6.1 Clean data

Normally, before transferring the data from one platform to the other, the data needs to checked with some tests for corruption. The policy decided upon in section 3.2, “Stage one — defining the strategy” on page 14 will be implemented as the data cleansing plan.

In our project we did not include data cleansing, because the data was checked for validity and correctness when packaging the application prior to the start of the project.

6.2 Unloading data from Oracle

In this section we describe the methods we used to unload data from the source Oracle database.

6.2.1 Character, numeric, and date data types

The Process and Laboratory segments of CIPROS Oracle database contain only CHAR, VARCHAR2, DATE, NUMBER and NUMBER(n) data types.

This is a very common situation: In this case, we can use Oracle *SQL*Plus* (called *SQLPlus* from now on) to download data from the Oracle database into flat files. *SQLPLus* is a front-end product to the SQL language in Oracle, which, by means of format commands, also allows you to manipulate data and spool it on flat files. The output is usually formatted in columns, as we can see in the example reported in Figure 47.

```
cipros@BALTIC /home/cipros > sqlplus ciprosusr/ciprospwd

SQL*Plus: Release 3.3.3.0.0 - Production on Thu Jul  8 11:45:41 1999

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.


Connected to:
Oracle7 Server Release 7.3.3.0.0 - Production Release
PL/SQL Release 2.3.3.0.0 - Production

SQL> spool file.out
SQL> select * from data_type;

DATA_T DESCRIPTION
------ ----------------------------------------
REAL   Real value
TEXT   Text Value

SQL> spool off
```

Figure 47. Example of SQLplus usage

The file *file.out* contains the following lines:

```
SQL> select * from data_type;

DATA_T DESCRIPTION
------ ----------------------------------------
REAL   Real value
TEXT   Text Value

SQL> spool off
```

The commands *columns, set linesize, set pagesize, set feedback* allow you to manipulate the output. For example, if you have records longer than the default *SQLPlus* line length, your lines will be split into more than one line; in this case we can use the *set linesize* command to increase the *SQLPlus* line length.

Unfortunately, we could not use the "*" symbol to select all the columns from the tables, because in some cases (DATE columns, for example) we need to manipulate the output format to handle differences. So, we have created a script that gets the table description automatically from Oracle catalog and, with an *awk* script, generates the SELECT statement with the concatenation of the columns.

When the script concatenates columns, we lose the automatic formatting in columns of *SQLPlus*, so we need to re-create the column layout in the output files.

To do this, we have used the RPAD function. With this function we could fill in, with blanks, the space to the right of each column, up to the exact maximum length of each field. We used the DECODE function in order to fill NULL fields with blanks.

If you have tables with no special handling of columns required, like dates, you can create a simple *SQLPlus* file *mytable.sql* containing the following commands:

```
clear columns
clear breaks
set pagesize 50000
set linesize 200
set feedback off
set heading off
set echo off
set space 0
set newpage 0
spool lll.lst
select * from MYTABLE;
spool off
quit
```

And then you can issue the command:

```
sqlplus ciprosusr/ciprospwd @mytable.sql
```

The script ***download.sh*** (see A.9, “download.sh script” on page 205) uses the *SQLPLus* Oracle command line and the DESCRIBE, RPAD, DECODE, SELECT commands to download onto flat files the source Oracle data (one file for each table).

The shell script can be issued with the syntax:

```
download.sh tables.lst
```

where *tables.lst* is a file containing just the list of the downloaded tables.

This script performs the following operations:

- Scans each table *TABLE_NAME* listed in the input file
- Extracts the table definition with the DESCribe command
- Creates a file *TABLE_NAME.dsc* containing, for each field of the table, the length of the columns, according to their data type
- Computes the total record length of the table using the *count.awk* awk script file reported in A.9.1, “count.awk script” on page 207
- Creates a *TABLE_NAME.sql* file containing the select statement, built also using the *desc.awk* awk script file reported in A.9.2, “desc.awk script” on page 207
- Runs the *TABLE_NAME.sql* file and spools the output onto the *TABLE_NAME.dat* file
- Closes the loop on each table of the input file

After unloading the data files, we have transferred them to the OS/390 system for the DB2 loading operation. Refer to 6.5, “Transferring data from AIX to OS/390” on page 118.

For variable length character fields (VARCHAR), we cannot use the DB2 LOAD utility unless the length of the data itself is written in its binary representation in the first two bytes of each VARCHAR field of each record. Refer to 6.6, “Reformatting data for DB2” on page 119.

Note: The *download.sh* script would also download the RAW data, not present in our application, since *SQLPlus* performs an implicit conversion to the hexadecimal representation of RAW data columns. But then a REXX program on S/390 must be written to decode the hexadecimal data in the file into its binary character representation. After such conversion the DB2 LOAD utility can be used.

6.2.2 Other data types and exceptions

Unfortunately, not all the data can be downloaded to a flat file, transferred to the S/390 and loaded into DB2 without some loss of information during this operation. This can happen because of:

- The special data types used in table definitions
- The differences between the two operating systems
- The conversions made by *ftp*

6.2.2.1 Long and non-spoolable fields

One limitation is due to the maximum length of a spoolable line allowed in Oracle *SQLPLus* command line. The maximum length is platform-dependent. This value can be changed with the SET MAXDATA parameter of *SQLPlus* Oracle utility, but the limit with AIX is 60000 bytes.

The ARRAYSIZE parameter of the SET statement affects the maximum line length, too. It allows to change the size of the batch of rows that *SQLPlus* fetches at one time. The combination of the two parameters must not overflow the buffer length of the *SQLPLus* product.

Another limitation comes from the maximum LINESIZE parameter for the SET command of *SQLPlus* product, that is, 32767 bytes. Moreover, the LONG parameter of the SET command, used to increase the maximum spoolable length of LONG fields, has the same limit of 32767 bytes.

These limitations in *SQLPlus* constitute a problem if your tables contain LONG fields and the content of the LONG field is longer than 32767.

Instead, if the data contained in the LONG fields of your tables is shorter than 32767 bytes, we can use *SQLPLus* as in the following example:

We have created and inserted a table with the following statements issued from a SQL*plus* command line:

```
CREATE TABLE LTAB (L LONG);
INSERT INTO LTAB VALUES('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
```

Then, we have created a file *mylongtab.sql* containing the following lines:

```
set pagesize 50000
set linesize 500
set long 500
set feedback off
set heading off
set echo off
set space 0
set newpage 1
spool LTAB.DAT
select * from LTAB;
spool off
quit
```

Then, we have issued the SQL script with the following command:

```
sqlplus ciprosusr/ciprospwd @mylongtab.sql
```

Since the LONG fields are variable length character fields, they have to be managed as VARCHAR. Refer to 6.6, “Reformatting data for DB2” on page 119. If the page size in the table space which contains your DB2 table is large enough to contain the Oracle data, you may also define in DB2 the LONG field as a VARCHAR.

Another limitation is constituted by the maximum result of a string concatenation, that is, 2000 bytes. If you are using the *download.sh* script described in 6.2.1, “Character, numeric, and date data types” on page 109, you will have problems with tables whose record length is more than 2000 bytes.

In these cases, you can spool one on more columns on different files and then use the *paste* AIX command to join vertically the columns in a unique file.

Let us look at an example of string concatenation with maximum length of 2000 bytes.

We have created a table with the following definition:

```
CREATE TABLE LONGTAB (FIELD1 CHAR(50) NOT NULL,
                      FIELD2 VARCHAR2(1000),
                      LONG_FIELD VARCHAR2(1000));
```

We have creates two SQL scripts as follows:

```
rem Script longtab_1.sql
clear columns
clear breaks
set pagesize 50000
set linesize 1050
set feedback off
set heading off
set space 0
set newpage 1
spool LONGTAB_1.DAT
select ''
 || rpad(DECODE(FIELD1,NULL,' ',FIELD1),50)
 || rpad(DECODE(FIELD2,NULL,' ',FIELD2),1000)
from LONGTAB;
spool off
quit
```

```
rem Script longtab_2.sql
clear columns
clear breaks
set pagesize 50000
set linesize 1000
set feedback off
set heading off
set space 0
set newpage 1
spool LONGTAB_2.DAT
select ''
|| rpad(DECODE(LONG_FIELD,NULL,' ',LONG_FIELD),1000)
from LONGTAB;
spool off
quit
```

We can execute the two SQL files with the following command to obtain the output data files LONGTAB_1.DAT and LONGTAB_2.DAT (*n* should be 1 and 2):

```
sqlplus ciprosusr/ciprospwd @longtab_n.sql
```

After deleting the first heading line in both files and the last empty line (if any), we can paste vertically the two files into the LONGTAB.DAT file with the following AIX command:

```
paste -d"\0" LONGTAB_1.DAT LONGTAB_2.DAT > LONGTAB.DAT
```

Note: Since also the *vi* editor has a limitation on the displayable line length (2048 bytes), you may not see the file with this editor. You can either use the CDE system editor, if you have it installed in your environment, or the *cat* command.

6.2.2.2 Binary fields

Care must be taken when managing binary fields.

The short binary data, such as RAW(n) or CHAR(n) FOR BIT DATA, can be unloaded by using the *download.sh* script described in 6.2, “Unloading data from Oracle” on page 109 and transferred using *ftp*, since the *SQLPlus* interface performs an implicit conversion or RAW data into hexadecimal representation.

Unfortunately, the LOAD utility does not allow to load binary data from a file containing its external hexadecimal representation.

You can write a simple batch program (REXX or C, or the language you prefer) to decode the hexadecimal strings into binary strings.

If the size of your table is not large, you can also INSERT it into the DB2 table. We have tested this with the following example.

We have created in the Oracle database, from an *SQLPlus* command line, a table called RAWTAB as follows:

```
CREATE TABLE RAWTAB (FIELD RAW(255));
```

The corresponding DB2 table is created, for instance from SPUFI, as follows:

```
CREATE TABLE RAWTAB (FIELD CHAR(255) FOR BIT DATA);
```

The INSERT statements can be extracted directly with *SQLPLus* by creating a file *extr_ins.sql* containing the following lines:

```
clear columns
clear breaks
set pagesize 50000
set linesize 1000
set feedback off
set space 0
set newpage 1
spool INSRAW
select 'INSERT INTO RAWTAB VALUES(X''' || FIELD || ''');' from RAWTAB;
spool off
quit
```

and then by executing the *extr_ins.sql* script as follows:

```
sqlplus ciprosusr/ciprospwd @extr_ins.sql
```

After deleting the heading and trailing lines, the INSRAW file can be transferred to the OS/390 system and used, for example, with SPUFI.

We can use also DataJoiner to map both tables and then use the INSERT/SELECT concatenated statements to migrate the data. Refer to 6.9, “Loading data into DB2 using DataJoiner” on page 127.

Note: Both using a data file and DataJoiner, DB2 fills in the trailing space of CHAR FOR BIT DATA fields with blanks.

As an example, we inserted into Oracle RAWTAB table a record as follows:

```
insert into rawtab values(hextoraw('ab10ab10ffffff'))
```

After migrating the data to DB2 using one of the two methods above, the content of the RAWTAB DB2 table, as selected from SPUFI, is as follows:

```
SELECT HEX(FIELD) FROM RAWTAB;
---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+
AB10AB10FFFFFF4040404040404040404040404040404040404040404040404040
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
SNE612I DATA FOR COLUMN HEADER  COLUMN NUMBER 1 WAS TRUNCATED
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+
```

6.3 File transfer and format programs

After the data has been downloaded into a file, we used the TCP/IP File Transfer Protocol (*ftp)* to transfer the data files to the OS/390 system. The TCP/IP services were already installed on S/390 system. On the AIX system, TCP/IP is included in the base operating system. The configuration of the systems for the transmission implies:

- The definition of an *ip* address and a hostname (*BALTIC*, in our case), provided centrally by the network administrator, together with the domain name, the router address, the network mask and the nameserver address
- The definition of a hostname for the OS/390 system in */etc/hosts* file (*mvsftp*, in our case), corresponding to its *ftp* address.

Refer to *AIX Version 4 System User's Guide: Communications and Networks*, SC23-2545 and to *OS/390 eNetwork Communications Server: IP Configuration*, SC31-8513 for detailed information on network configuration and set up.

To perform an automatic transfer from AIX to OS/390 systems and vice versa, we used the *.netrc* network configuration file of *cipros* user, located in its home directory (*/home/cipros*).

It contains the following lines:

```
machine mvsftp login paolor3 password pr3pwd
```

This file needs to have read/write/execution permission only for the owner, and neither for the group *cipros* user belongs to, nor for other users. To inhibit the usage from the other users, you can issue the command:

```
chmod 600 /home/cipros/.netrc
```

At the end, after creating a partitioned data set PAOLOR3.CIPROS.DATA on OS/390, refer to 6.4, “Creating a PDS on OS/390” on page 117, we have created a file *put.ftp* containing the ftp commands for the file transfer, as follows:

```
asc
put ANALYSIS_METHOD.DAT 'paolor3.cipros.data(ANMETHOD)'
put ANALYSIS_SPEC.DAT   'paolor3.cipros.data(ANSPEC)'
put APPL_DATA.DAT       'paolor3.cipros.data(APPLDATA)'
...
...
...
put UNIT_ISLAND.DAT     'paolor3.cipros.data(UNITISL)'
put USRID_CONFIG.DAT    'paolor3.cipros.data(USRIDCON)'
put VALUE_TYPE.DAT      'paolor3.cipros.data(VALUETYP)'
```

The files *TABLE_NAME.dat* have been created by the script *download.sh* described in 6.2, “Unloading data from Oracle” on page 109, but a new naming convention for the data files has to be used on the OS/390 side, since the data set name qualifiers cannot be longer than 8 characters.

With the *asc* option, an automatic conversion between ASCII and EBCDIC is performed by the *ftp* protocol. We were able to use this options for all our files since they all contained only ASCII (or externally ASCII) data.

The *put.ftp* file can be executed from an AIX shell by the *cipros* user with the command:

```
ftp mvsftp < put.ftp
```

6.4 Creating a PDS on OS/390

You can use ISPF option 3.2 to create the PDS that is going to receive in OS/390 the data transmitted from AIX. In our project we transmitted files that are all of the same length. Therefore, we created a fixed format PDS.

You should calculate your own space requirements when allocating the PDS (or any data set). Request enough space in the initial space allocation to contain all of the expected data. Only the initial allocation is provided to you. Do not depend on OS/390 to allocate default extents to hold initial data. Multiple extents may affect performance, especially if they are not allocated in contiguous storage on traditional devices, and later on you might run out of space interrupting the transmission at the most unsuitable time. It might be worthwhile creating a small procedure to calculate the allocation for all data objects needed.

Be sure to use block sizes that are efficient for the DASD device you will be using. Your installation may have CLISTS, or ISPF menu options, that will do this calculation for you. See your ISPF system programmer about this. The other option in OS/390 is to use system determined block sizes. OS/390 will calculate

the optimum block size for the device the data will reside on. This is done by entering 0 as the block size.

Note: A quick way to allocate a PDS (or any OS/390 data set) is to use another one as a model. In ISPF option 3.2, from the Data Set Utility option, enter the name of a known PDS. If possible, one that is resident on the device you plan to use for your PDS. When the “Data Set Information” panel is displayed press enter. This returns you to the “Data Set Utility” initial panel. However, the information from the previous operation is saved. Type “A” on the command line, for allocation of a new data set, and type in the data set name on either the lines provided for ISPF libraries or the one labeled for "Other...". A panel will be displayed titled “Allocate New Data Set”. On this panel will be the information from the data set you chose as a model. Change any parameters you need to. Be sure to allocate enough “directory blocks” to hold what will become the index associated with the data. When you have made changes to your satisfaction, press enter.

6.5 Transferring data from AIX to OS/390

Oracle on AIX uses ASCII data encoding, our existing DB2 on OS/390 uses EBCDIC. This, of course, presents us with the a known problem. However, this was easily solved because in the project we used File Transfer Protocol (FTP) to move the data from the client to the OS/390 server. During the transfer process we utilized the Character Data Conversion (CDC) facility of FTP. When file transfer is initiated the correct Coded Character Set Identifier (CCSID) is specified and conversion is achieved by FTP. Refer also to *DB2 UDB for OS/390 Version 6 SQL Reference*, SC26-9014 and *DB2 UDB for OS/390 Version 6 Installation Guide*, GC26-9008 for considerations on ASCII CCSID support and coexistence provided by DB2. We decided to convert all data because of some restrictions when accessing DB2 tablespaces with different CCSID.

As shown in 6.3, “File transfer and format programs” on page 116, after configuring the network layer between the two servers and creating the files *.netrc* and *put.ftp*, we were able to transfer the data files with the following command executed by *cipros* user from an AIX shell:

```
ftp mvsftp < put.ftp
```

The data, when stored on the OS/390 system is now in EBCDIC representation and ready for use by DB2 LOAD utility. But we need to first execute the REXX program that we coded to establish the 2 binary bytes VARCHAR length field, see 6.6, “Reformatting data for DB2” on page 119.

Originally we intended to add the 2 byte length field when the data was extracted from Oracle. However, placing 2 bytes of binary data in the extracted record using a C language program proved too difficult to do. This left us with the alternative of changing the data on the OS/390 side after transmission. The binary field could be dealt with easily using any of the well known mainframe languages. We chose REXX because it could be coded quickly, the amount of data was limited, and the talent was immediately available to write the program. The same kind of program could be written in COBOL or PLI, for example, and optimized for performance.

Using FTP, the transmitted files on OS/390 were received on PDS where each member represented the data for one table. This approach is feasible since we were using small amounts of data. However, the receiving data set need not be a PDS and could be defined as a sequential file large enough to hold any large table. For very large tables the receiving sequential data set could also be specified as a tape data set if not enough disk space is available.

6.6 Reformatting data for DB2

The DB2 LOAD utility will load data from a flat-file format either from a partitioned data set or a regular flat file. The file must include length information for those fields that become VARCHAR columns in a DB2 table. Reformatting the data for DB2 means adding 2 bytes of binary data to the beginning of such a field. These 2 bytes contain a binary number representing the length of the data within the field. The DB2 LOAD utility requires that this field be present for VARCHAR data. DB2 also requires that the 2 byte binary field be at the beginning of the data to be considered VARCHAR but it is not, itself, counted as data to be loaded. For example, if a variable character field able to hold up to 10 bytes is to be loaded, the entire field will be 12 bytes long. 2 bytes for the binary length field and 10 bytes for the data. The position indicated to DB2 as the beginning of the field to become VARCHAR for the LOAD, is the beginning of the 2 bytes of binary data.

The data unloaded from the Oracle tables and transmitted to OS/390 is stored in a PDS whose members represent the data for the individual tables that have been transmitted. For example, the data for table PIPELINE has been stored in member named PIPELINE. Since member names for a PDS may only be 8 characters long, abbreviation is necessary. The data stored in the PDS is used as input for a program process to determine the length of the variable character data and create and populate the binary bytes.

The data for our project was processed by a REXX program. It needs positional input describing where in its input file records it will find the beginning of each field and what kind of data will be encountered. Leading and embedded blanks are significant data and will be included in the resulting count of variable character positions. Trailing blanks are not counted.

The REXX program also outputs a file with the positions and lengths of the fields in the reformatted output data file. This file is useful when setting up the LOAD utility commands. The positions indicated by the REXX program are those the LOAD utility will expect when it reads the REXX converted data as input. The REXX program writes the complete input file plus any needed binary bytes to the output file.

We decided to pass to OS/390 records whose fields were the maximum length for the data type they held. Therefore, if the field was a VARCHAR(10) column in the Oracle database table then the field in the record transmitted to OS/390 was 10 bytes long. The extra bytes are filled with blanks. Integer fields varied in size from 2 to 10 bytes. Floating decimal fields were transmitted at their maximum size of 41 bytes. Timestamp fields were transmitted as 26 bytes. All of these fields would have been defined to the REXX program as fixed fields. The program recognizes fixed fields as identified by an “F”. Variable fields are identified by a “V”. Refer to B.19, “JCL for the conversion of data using REXX program” on page 234 for the example JCL and an explanation of the nomenclature.

Table DATA_TYPE, for example, consists of 2 columns of VARCHAR data as shown by the following CREATE TABLE SQL statement in Figure 48

```
CREATE TABLE DATA_TYPE (DATA_TYPE VARCHAR(6) NOT NULL,
            DESCRIPTION VARCHAR(40))
            IN CIPROS.CPRSBASE;
```

Figure 48. Create table example

The input data from the Oracle table looks like Figure 49.

```
REAL Real value

TEXT Text Value
```

Figure 49. Transmitted Oracle data

The position indication input file for the REXX program is like Figure 50.

```
//CTLIN    DD *
1   6    V         DATA_TYPE (DATA_TYPE VARCHAR(6)
7   40   V         DESCRIPTION VARCHAR(40))
```

Figure 50. REXX input control file

The "//CTLIN DD *" is a JCL statement indicating the data represented by file CTLIN follows instream in the job. Each line represents one input record to the program. The first record's layout indicates that starting in position 1 there are 6 bytes of data that are variable. The second record indicates that starting in position 7 there are 40 bytes of variable data.

Note: REXX utilizes main storage for its processing and reads in all of its input. The maximum limit may be reached quickly when processing a table with a large number of rows. Our tables held small amounts of data and were appropriate for the REXX solution. For a large amount of data you will want to design and write a program in COBOL or PLI using the REXX program as a model.

The program, reported in B.19, "JCL for the conversion of data using REXX program" on page 234, was run as an OS/390 batch job. The first step sets up the REXX program in a temporary load file. The subsequent steps process each member of the source data in turn.

You will notice in the JCL that the REXX output is variable. This is done to save space. When you calculate the potential length of the output file you need to count three things: the length of the data, the extra 2 bytes needed for each variable character field, and 4 bytes for the system generated lengths attached to a variable file's records. For example, if your file has 4 original fields and two are fixed in length and 2 will be variable and each of the 4 is 10 bytes long then the output length is 48. 40 for the 4 actual fields, 4 for the binary length data for the 2 variable character fields and 4 for the system generated length for the variable blocked file record.

Null columns are dealt with by the REXX program in this way: If the null or empty column is VARCHAR it will have its length set to 1 and be filled with spaces

(x'40'). If it is fixed, it will be padded with spaces. The NULLIF verb will have to be utilized to correctly load these fields. See Figure 51.

```
LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES
     INTO TABLE PAOLOR2.ENGINEERING_UNIT
          (ENG_UNT POSITION(1) VARCHAR,
           ENG_UNT_TYPE POSITION(13) VARCHAR
            NULLIF (ENG_UNT_TYPE = ' '),
           DESCRIPTION POSITION(30) VARCHAR)
     ENFORCE CONSTRAINTS
```

Figure 51. Example of LOAD statement with a NULLIF parameter

You can find a description of the output and its use in 6.8, "Loading data into DB2 using the LOAD utility" on page 121.

6.7 Checking data for the correct format

After converting the data to include the length for VARCHAR columns, you should check the results. The obvious reason for this is that no matter how careful you have been in calculating the positions and lengths for the input, errors will be introduced. Examining the output now is a quality control point that is well worth the effort.

Quick visual scanning of the output will tell you if any fields have been misplaced. You will see problem indications like the 2 byte binary data in a location that is not at the beginning of the appropriate field, or fields that seem too short or long. This can indicate that the calculation for the position of the beginning of the file was in error or that the input was actually longer or shorter than expected. If your data is too short ensure you have allowed for all the potential length fields (the 2 bytes binary data) and for the 4 bytes of system generated length for variable blocked records. Correct your counts or JCL and resubmit the job.

This may seem like a tedious task but it is really important to make sure to have correct data before the DB2 loads are run.

6.8 Loading data into DB2 using the LOAD utility

The DB2 LOAD utility is normally run as an OS/390 batch job. Another possibility is to run it from the DB2I panels. See the LOAD job in Appendix B.15, "JCL for first job to LOAD CIPROS tables" on page 228 for the sample JCL that we used during the project.

When loading data into a table you can also use the INSERT SQL statement from an application program. The LOAD utility is more efficient than the INSERT program. This is especially true for large amounts of data. Among other things, with LOAD, you may specify LOG NO, avoiding the overhead of logging all the inserted records, and just doing a quick COPY of the table space at the end. The LOAD utility also checks for referential integrity during its operation when the parameter ENFORCE CONSTRAINTS is specified.

If referential integrity is needed, enforcing referential integrity and detecting referential integrity errors at the time the various tables are loaded can be less time consuming than using the CHECK DATA utility after all the data has been

loaded. If tables are loaded in the proper order based on their primary and foreign keys definitions this will work well.

We suggest that you graphically lay out the referential integrity relationships across the tables as reported in Figure 52.

```
DESIGNATION                                          MEANING

                    CIPROS01                         STORAGE GROUP NAME

                    CIPROS                           DATABASE NAME

CPRSBASE   CPRSLAB  CPRSLOGE   CPRSREAD   CPRSPRDS   TABLESPACE NAMES

DATA_TYPE  SAMPLE   APPL_DATA  READING    CASES      TABLE NAMES
.          .        .          .          .
.          .        .          .          .
```

Figure 52. Sample database graphical layout

Just include a listing of all tables with foreign key reference. This will help in determining the order of loading the tables. An example is shown in Figure 53. In this approach the table with the foreign key is indented below the table to which it refers to. Do this for each table. Tables with no references will have no indented lines below them. Here you can see readily that both ANALLYSIS_METHOD and COMPOS_TYPE need to be loaded before METHOD_RESULT.

```
ANALYSIS_METHOD
        METHOD_RESULT
COMPOS_TYPE
        METHOD_RESULT
```

Figure 53. Sample reference listing

If you do not enforce referential integrity at load time with the ENFORCE CONSTRAINTS DISCARDS 0 option of the LOAD utility, tables may be loaded in any order. However, when you do check referential integrity any missing data will cause errors. Therefore, you will still have to load the data for all affected tables. The consequence of this approach is that if there are integrity problems you will have a bigger mess to clean up since you may now have more dependent tables to deal with at the same time. DB2 can be directed to delete offending rows. This may have a ripple effect through various tables depending on how interlaced the foreign key structure is. Your table spaces will now be in *check pending status* and in need of extra work.

The description of the data to DB2 is contained in the LOAD statement parameters. The POSITION parameter in the LOAD statement should match the position in the input file of the various fields. For example, if you state position 1 begins a fixed length character data field of 3 bytes DB2 will expect to find 3 bytes of character data there to place into a CHAR(3) column in the specified table. If the next position, which would be 4, represents the beginning variable character data DB2 should find the 2 byte binary length field not the beginning of the actual data. DB2 will expect to find the beginning of the data in the third byte, or position 6.

As mentioned in 6.6, “Reformatting data for DB2” on page 119 the description output from the REXX program is valuable when setting up the positional input for the DB2 LOAD. In our example JCL this output is named, for example, CIPROS.DATATYPE.CTLOUT. DATATYPE is the abbreviated name representing the table DATA_TYPE. The actual converted data is found in the file named CIPROS.DATATYPE.DATAOUT. You can see that the positional description found in CTLOUT matches the locations of the various fields in DATAOUT.

Examples of these two files are in Figure 54.

```
CTLOUT:
* Control file describing converted data (DATAOUT)
* (same format as CTLIN)
1 6 V
9 40 V
DATAOUT:
..REAL  ..Real value
..TEXT  ..Text Value
```

Figure 54. CTLOUT file

The two dots in front of each readable piece of data represent the 2 binary length bytes since these fields are to become VARCHAR columns.

If we set the data set profile for DATAOUT to "hex on" under ISPF we would be able to view the values, in hexadecimal of course, for the length. See Figure 55.

```
...REAL  ..Real value
00DCCD4400D8894A89A84444444444444444444444444444444
049513000A9513051345000000000000000000000000000000
 ------------------------------------------------
..TEXT  ..Text Value
00ECEE4400E8AA4E89A84444444444444444444444444444444
043573000A3573051345000000000000000000000000000000
```

Figure 55. Hex version of DATAOUT file contents

You may notice that the word "TEXT" above is 4 bytes long. However, we are sure you have also noticed that there are 6 positions represented in the file. As we mentioned, the REXX program moves all of the input record’s fields to the output record. DB2 will load only the 4 bytes "TEXT" as indicated by its length of 4. The same is true for the other field.

The DB2 LOAD statement for DATA_TYPE taken out from its JCL context is reported in Figure 56.

```
LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES
     INTO TABLE PAOLOR2.DATA_TYPE
          (DATA_TYPE POSITION(1) VARCHAR,
           DESCRIPTION POSITION(9) VARCHAR)
```

Figure 56. Sample LOAD statement

Notice that position 1 and position 9 are where DB2 looks for the data and that it l finds the 2 byte length information there with the data following. The parameter INDDN(CPRSRECS) refers the LOAD to the input file DD card which describes the input file for the LOAD. LOG NO tells DB2 not to log the loaded records. The implication here is that the log cannot be used for recovery. However, this is not much of a problem since we are loading the table for the first time. To recover before the LOAD is completed we would use the LOAD again. If any rows had

been loaded we would have to delete them before reloading due to the RESUME YES parameter specified for our segmented table space. If the rows are not deleted the LOAD abends for duplicate keys. RESUME YES is needed for segmented table spaces with more than one table in each table space. If REPLACE were used, data of tables loaded previously would be destroyed.

Table DATA_TYPE has, in our project, a referentially dependent table TAG. Therefore, DATA_TYPE must be loaded before TAG. DATA_TYPE is not dependent on any other table and may be loaded at anytime in the loading sequence as long as it is before TAG.

If the field to be loaded is known to have null then the NULLIF condition should be coded. For example, table TAG has several columns that may be null. The input to the LOAD will be all spaces. So, you must code the NULLIF condition as shown in Figure 57. Notice the parameter ENFORCE CONSTRAINTS.

```
LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES
     INTO TABLE PAOLOR2.TAG
   (TAG_NAME POSITION(1) VARCHAR,
       ...
       ...
       ...
    STATUS POSITION(73) VARCHAR NULLIF (STATUS = ' '),
HI_VALUE POSITION(85) FLOAT EXTERNAL(41) NULLIF (HI_VALUE = ' '),
LO_VALUE POSITION(126) FLOAT EXTERNAL(41) NULLIF (LO_VALUE = ' '),
    DATA_TYPE POSITION(167) VARCHAR,
    ENG_UNT POSITION(175) VARCHAR,
    DESCRIPTION POSITION(187) VARCHAR,
    CREATION_DATE POSITION(229) TIMESTAMP EXTERNAL(26)
                NULLIF (CREATION_DATE = ' '),
    DISCONN_DATE POSITION(255) TIMESTAMP EXTERNAL(26)
             NULLIF (DISCONN_DATE = ' '))
     ENFORCE CONSTRAINTS
```

Figure 57. Sample LOAD statement with NULLIF

With the REXX output of a length of 1 and spaces for the VARCHAR column, it can be assumed that null should be the result. The float and timestamp columns have corresponding fixed fields in the REXX file. It can be assumed that spaces mean null and the NULLIF condition will establish this in the table.

The obvious caution here is for a character column where space may be valid. If this is so, is a nullable column the best solution? If null is acceptable use some other character unacceptable as a value for the column as the indicator for NULLIF.

The EXTERNAL(n) parameter should be coded on the LOAD statement for integer, float and timestamp data types. This tells the LOAD utility what size field to expect.

Refer to *DB2 UDB for OS/390 Version 6 Utility Guide and Reference*, SC26-9015 for details on the LOAD Utility.

Pending statuses

Occasionally a LOAD or other utility job will fail. When this happens the utility places the table space in a pending status. The status depends upon what was going on at the time. The statuses you are most likely to see are RECP, recovery pending; CHKP, check pending; COPY, copy pending (when LOG NO is specified); RBDP, index rebuild pending;. A list of these and other statuses along

with a discussion of them and various solutions can be found in *DB2 UDB for OS/390 V6 Command Reference*, SC26-9006 and the *DB2 UDB for OS/390 V6 Administration Guide,* SC26-9003.

To begin recovery from a status that prevents further work, such as copy pending, you need to see if any utility is still running against the table space and what kind of pending status it is in. In several instances the utility execution may be restarted after correcting the situation.

Figure 58 shows the command to be executed under DB2 interactive (DB2I), option 7, to verify anomalies in the status of the database objects. Use caution when executing commands from the command processor since you might leave the cursor positioned for execution on the wrong line.

```
-DIS DATABASE(CIPROS) SPACENAM(*) RESTRICT
```

Figure 58. Sample DISPLAY DATABASE command

The DISPLAY command will show the CIPROS table spaces with restricted statuses. The table spaces in the correct read/write (R/W) status will not display because of the RESTRICT parameter.

The next step is to see if any utilities are still running. Figure 59 shows the related command.

```
-DIS UTILITY(*)
```

Figure 59. Sample DISPLAY UTILITY command

If any utility is running with your ID, and it is certainly recommended, you need to terminate it before DB2 lets you run a recovery utility for the table space. Figure 60 shows the TERMINATE command.

```
-TERM UTILITY(DSNTEX)
```

Figure 60. SampleTERMINATE UTILITY command

Once this is done you will be able to execute a recovery job using the appropriate utility statement. Appendix B, “Sample DB2 for OS/390 jobs” on page 215 has several sample jobs for recovery, rebuild, check or copy executions.

Submit the job with the correct command and utility and, when the job is complete, the status will be reset by DB2 to read/write (RW).

For example, the LOAD utility, coded with LOG NO, leaves the table space in COPY pending status. To relieve this situation you will have to run an image copy of the table space. B.11, “JCL for REORG, RUNSTATS and COPY of CIPROS table spaces” on page 223 shows a job with steps that take image copies.

DB2 keeps track of the names of data sets used to store the image copy made during any job. You will have to change the file name each time you run an image copy. Using some sort of numbering scheme or generation data group that makes this easier for you to keep track of it, as well.

Hang situation

Occasionally, our prerelease copy of DB2 V6 would hang. When this condition occurred the only solution was to take drastic measures. Sometimes cancelling the job that was running against a table space was enough. Some other times we had to stop and restart the DB2 subsystem and cancel the TSO userids of those involved. You may or may not be able to execute these commands yourself. Usually the computer console operator can execute the cancel command if you cannot. He or she can also execute the stop and start commands of the DB2 subsystem, as well.

The userid cancel command is in Figure 61.

```
/c u=paolor1
```

Figure 61. Cancelling a TSO user

If cancelling the userid is not enough to remove it from the system you can use the FORCE command. See Figure 62.

```
force u=paolor1
```

Figure 62. Forcing a TSO userid from the system

The START and STOP commands for the DB2 subsystem must be entered from the console, or console like function (SDSF command option or started task.) The format for the stop command is in Figure 63.

```
=db2x stop db2 mode (force)
```

Figure 63. The STOP DB2 command

The '=DB2X' part of the command tells the operating system which instance of DB2 is to be stopped. This is very important since there may be more than one in your system. In our environment we had six. The MODE(FORCE) tells DB2 to stop regardless of what is running. If you use the regular stop command DB2 will wait until there is no other activity before stopping.

You should watch the system log in SDSF to see the progress of the shutdown and for the indication from DB2 that it is ready for restart. When this message is displayed you can enter the start DB2 command shown in Figure 64.

```
=db2x start db2
```

Figure 64. The START DB2 command

Notice that the DB2 instance must be specified if you have more than one.

If everything else fails the OS/390 command reported in Figure 65 will bring all DB2 address spaces down. Hopefully you will not need these commands at all.

```
P IRLMprocname
```

Figure 65. The STOP IRLM proc OS/390 command

6.9 Loading data into DB2 using DataJoiner

DataJoiner may represent significant time and effort savings if it is available to you. It can be used to move data directly from Oracle on a client, such as a RISC 6000, directly to a DB2 database on a server machine running OS/390. Connectivity can be achieved through TCP/IP. SQL is written to select the data from the various tables of the Oracle database and insert it into the corresponding tables of the DB2 database. Of course, before using DataJoiner the database design will have to be implemented in DB2 on the server. That is, the same jobs that we used to create the storage group, table spaces, tables, indexes, primary and foreign keys, and so forth will have to be run to use this approach as well.

6.9.1 Installing and configuring DataJoiner for AIX

In our project, we have used an F50 RISC machine, whose hostname is *sky,* as the gateway machine for the connection to both Oracle for AIX and DB2 for OS/390 servers.

On this machine, DB2 UDB for AIX V5.2 and Oracle 8.0.4 are installed. Using Oracle 8.0.4 as client for our Oracle 7.3 server did not present any compatibility problem.

Refer to Figure 66 for the DataJoiner configuration used.

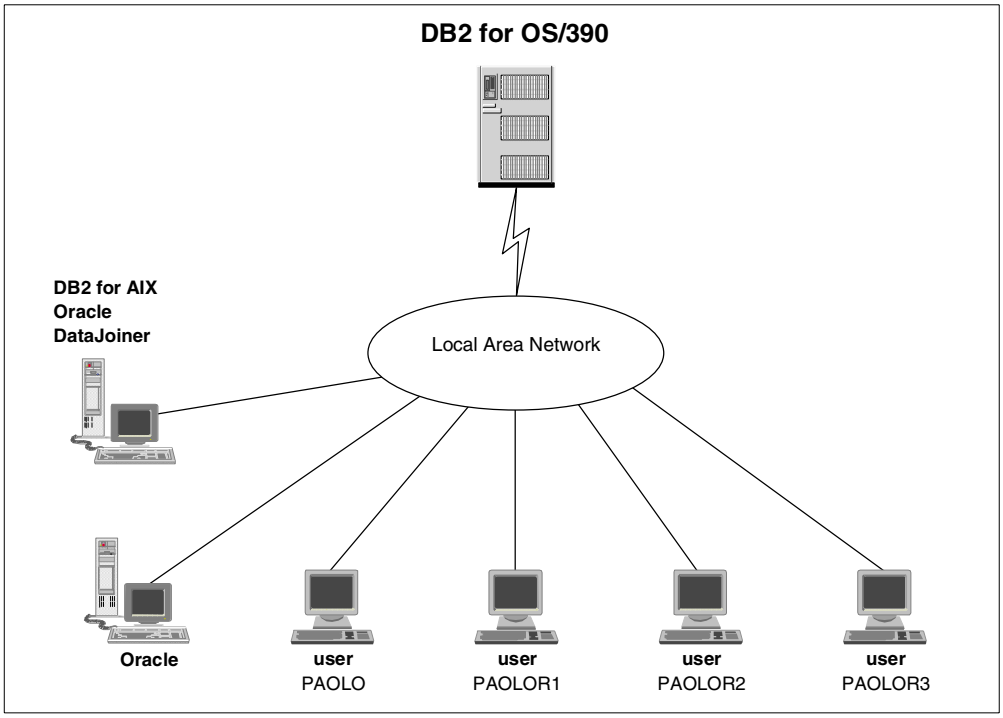


Figure 66. DataJoiner configuration

Oracle is required for the client access to the Oracle server. You may decide to install DataJoiner on the same machine where your Oracle server is installed.

DataJoiner Version 2.1.1 on AIX is the version that we used in our project. For more detailed information on the installation and configuration of DataJoiner, refer to *DataJoiner Implementation and Usage Guide,* SG24-2566 and to *Mining Relational and Nonrelational Data with IBM Intelligent Miner for Data Using Oracle SPSS and SAS As Sample Data Sources*, SG24-5278.

6.9.1.1 Installing and configuring the base product

After inserting the DataJoiner distribution media in the appropriate drive, you can install the product as follows:

- From an AIX shell and with *root* user, run the command:

```
smitty install_latest
```

- Input your installation device identifier (for example, */dev/cd0*)
- On the next screen, select the components you want to install (with F4 key), or select *all_latest*

After installing DataJoiner base product, the configuration includes the following steps:

- Create an AIX group for the DataJoiner instance (for example, *djinst*) using SMIT or with the AIX command:

```
mkgroup -A djinst
```

- Create an AIX user for the DataJoiner instance (for example, *djinst*), belonging to the *djinst* group previously created, using SMIT or with the AIX command:

```
mkuser pgrp=djinst djinst
```

- Create a DataJoiner instance for the *djinst* user previously defined, with the following command:

```
/usr/lpp/djx_01_01_0000/instance/db2instance djinst
```

The three commands previously described have to be issued by *root* user.

At this point, you can login to the AIX system as the *djinst* user and add the following line in the *.profile* file (running in a Korn shell):

```
. /home/djinst/sqllib/db2profile
```

where */home/djinst* is the *djinst* home directory.

Open the *db2profile* file in the *sqllib* subdirectory of *djinst* home directory (*/home/djinst*) and modify the lines as follows:

```
DB2COMM=TCPIP
export DB2COMM
...
DJXCOMM='db2ra drda drdaIP net8'
export DJXCOMM
```

Execute the *djinst .profile* file, for example, with the command:

```
. $HOME/.profile
```

At this point, the DataJoiner can be started by the *djinst* user with the command:

```
db2start
```

and a DataJoiner database can be created, for example, with the following command:

```
db2 create database DJDB
```

6.9.1.2 Configuring DataJoiner to access Oracle

In this section we describe the configuration of DataJoiner for the connection to the Oracle CIPROS database.

Configuration of DataJoiner Data Access Modules

Login as *root* and add the following lines in the */.profile* file:

```
ORACLE_HOME=/oracle8/product/8.0.4
LIBPATH=/usr/lpp/djx_02_01_01/lib
export ORACLE_HOME LIBPATH
```

where */oracle8/product/8.0.4* is the Oracle home directory.

then, execute the *.profile* file with the command:

```
. /.profile
```

Run the djxlink.sh script to create the Data Access Module for Oracle 8 RDBMS (*net8*).

```
cd /usr/lpp/djx_02_01_01/lib
./djxlink.sh
```

Note that this script creates all the Data Access Modules available with the DataJoiner release you are running. In our case, it creates also, for example, the *drdaIP* module for DB2 for OS/390 connection via TCP/IP.

Also the *SQLNet* Data Access Module is created. *SQL*Net,* as it is properly named but generally called *SQLNet*, is the Data Access Module of Oracle 7 RDBMS. If you are installing DataJoiner on the same machine where your Oracle 7 is installed, you have to use *SQLNet* instead of *net8* in the following configuration operations and commands.

Configuration of Oracle connection (SQLNet)
Login as *djinst* and add the following lines in his *.profile* file (*home/djinst/.profile*):

```
ORACLE_BASE=/oracle8
ORACLE_HOME=/oracle8/product/8.0.4
TNS_ADMIN=$ORACLE_HOME/network/admin
export ORACLE_HOME ORACLE_BASE TNS_ADMIN
LIBPATH=/usr/lpp/djx_02_01_01/lib
export LIBPATH
```

and run again his *.profile* file, as shown in 6.9.1.1, “Installing and configuring the base product” on page 128.

If you have installed DataJoiner on your Oracle machine, you are now able to connect to the Oracle RDBMS with the *djinst* user.

Since in our project the CIPROS Oracle database is installed on another machine (*BALTIC*), we had to configure *SQLNet* to reach the CIPROS database from the DataJoiner machine (*sky*). Refer to the Oracle documentation *ORACLE7 The Complete Reference*, G. Koch and K. Loney, Oracle Press, ISBN 0-07-882285-8 for detailed information on the Oracle Network (*SQLNet*) configuration.

With the Oracle owner user (*oracle*, in our case) we have added the lines listed in Figure 67 on page 131 in the *listener.ora* file of the *$ORACLE_HOME/network/admin* directory of CIPROS (BALTIC) server (if the *listener.ora* does not exist, create a new one):

```
LISTENER =
  (ADDRESS_LIST =
        (ADDRESS= (PROTOCOL= IPC)(KEY= cipros))
        (ADDRESS= (PROTOCOL= IPC)(KEY= PNPKEY))
        (ADDRESS= (PROTOCOL= TCP)(host= baltic)(port= 1525))
  )
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME= cipros.)
      (ORACLE_HOME= /home/oracle/app/oracle/product/7.3.3)
      (SID_NAME = cipros)
    )
  )
STARTUP_WAIT_TIME_LISTENER = 0
CONNECT_TIMEOUT_LISTENER = 10
TRACE_LEVEL_LISTENER = OFF
```

Figure 67. Example of LISTENER definition in listener.ora file

where:

- *port=1525* is referred to the port, in the file */etc/services*, indicated either with *listener* or with *orasrv* names
- *cipros* is the name of the Oracle instance (ORACLE_SID variable of your instance owner *.profile* file) you are connecting to
- *baltic* is the hostname of the Oracle server machine (as it is in the */etc/hosts* file)
- */home/oracle/app/oracle/product/7.3.3* is your Oracle home directory (ORACLE_HOME variable of your instance owner *.profile* file)

Then, we started the Oracle *SQLNet listener* as follows:

```
lsnrctl start
```

On the DataJoiner machine (*sky*, in our case), we have added the lines listed in Figure 68 in the *tnsnames.ora* file of *$ORACLE_HOME/network/admin* directory (if the *tnsnames.ora* does not exist, create a new one):

```
baltic =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL= TCP)(host= baltic)(port= 1525))
    (CONNECT_DATA = (SID = cipros))
  )
```

Figure 68. Example of tnsname definition in tnsnames.ora file

We could test the connection between the two machines by executing, for example, the following command from the *djinst* user on the DataJoiner machine:

```
sqlplus ciprosusr/ciprospwd@baltic
```

The *quit* or *exit* commands can be executed to exit the SQLPlus command line.

Refer also to *Understanding SQL*Net Release 2.3*, A42484-1

Configuration of DataJoiner mappings and nicknames
Now, you can create the *server mapping* for your Oracle database.

First you need to connect to the DataJoiner database (with *djinst* user, from an AIX shell), as follows:

```
db2 connect to djdb
```

Then, you run the CREATE SERVER MAPPING statement with the following command:

```
db2 create server mapping from <SERVER NAME> to node <NODE NAME> \
type <DATABASE SERVER TYPE> version <VERSION NUMBER> protocol \"<DAM>\"
```

where:

- SERVER NAME is a unique name of your choise (*oradb*, in our case)
- NODE NAME is the *SQLNet* name used in *listener.ora* file (*baltic*, in our case)
- DATABASE SERVER TYPE is *oracle*
- VERSION NUMBER is the version of your Oracle database (*8.0,* in our case)
- DAM is the Data Access Module you are using (*net8*, in our case)

In our environment, the command was the following:

```
db2 create server mapping from oradb to node baltic type oracle \
version 8.0 protocol \"net8\"
```

With the following command, create the *user mapping* for the Oracle connection:

```
db2 create user mapping from djinst to server oradb \
authid ciprosusr password ciprospwd
```

where:

- *djinst* is the local user
- *oradb* is the server mapping previously created
- *ciprosusr* is the Oracle CIPROS user
- *ciprospwd* is his password

At the end, a locale nickname for a remote Oracle table can be created with the following command:

```
create nickname <NICKNAME> for <SERVERNAME.REMOTEUSER.TABLE>
```

that is, in our case, for example:

```
db2 create nickname OCASES for oradb.cipros.CASES
```

In Appendix A.10, “nick.sh script” on page 208 we provide a shell script which creates a list of CREATE NICKNAME statements for the creation of all the nicknames for our CIPROS tables, starting from a file containing the list of the tables. The script creates also the DB2 nicknames (refer to 6.9.1.3, “Configuring DataJoiner to access DB2 for OS/390” on page 133).

You can test the connection by selecting from the OCASES nickname, as follows:

```
db2 "select * from OCASES"
```

6.9.1.3 Configuring DataJoiner to access DB2 for OS/390

If you have already configured DataJoiner for Oracle database, you should already have the *drdaIP* Data Access Module (see 6.9.1.2, “Configuring DataJoiner to access Oracle” on page 129), otherwise you need to run, with *root* user, the following commands:

```
cd /usr/lpp/djx_02_01_01/lib
./djxlink.sh
```

Configuration of DB2 for OS/390 TCP/IP services

Refer to *Wow! DRDA Supports TCP/IP: DB2 Server for OS/390 and DB2 Universal Database*, SG24-2212.

Configuration of DataJoiner TCP/IP services for DB2

On the DataJoiner machine (*sky*), we need to enable the communication TCP/IP services for the connection to the remote DB2 for OS/390, using DRDA architecture.

To start you need to know the ip address related to the DB2 for OS/390 server. You can also define a hostname for it and insert it into local */etc/hosts* file (with *root* user), as we show in the following example:

```
9.12.2.35       wtsc63oe        wtsc63oe.itso.ibm.com
```

Then login to the DataJoiner machine as the *djinst* user and catalog a TCPIP node for your remote DB2 for S/390 as follows:

```
catalog tcpip node wtsc63oe remote 9.12.2.35 server 33340
```

The node name (*wtsc63oe*, in our case) can be chosen by you. We used the same hostname of the remote machine, as defined in */etc/hosts* file.

The server number (33340, in our case), must be the same you have defined as the listener port number on your DB2 for OS/390 environment. Refer to “Configuration of DB2 for OS/390 TCP/IP services” on page 133.

Then you can catalog the database in the local database directory, as follows:

```
catalog database db63 as db63 at node wtsc63oe authentication dcs
```

where the first *db63* database name is a pointer to the local DCS database directory (see next command), the second *db63* database name is an alias by which the database has to be known in the local server and *wtsc63oe* is the node name previously defined.

Now you can create an entry in the local DCS database directory with the command:

```
catalog dcs database db63 as db2x
```

where *db63* is the database alias previously defined in the CATALOG DATABASE command and *db2x* is the target database name by which the DB2 subsystem is known to the OS/390 system and which must match the location name stored in the BSDS of the DB2 for OS/390 subsystem.

You can now try to test the connection to the remote DB2 server with the command:

```
db2 connect to db63 user paolor3 using pr3pwd
```

where *paolor3/pr3pwd* are valid user/password on OS/390 system.

Configuration of DataJoiner mappings and nicknames
Now you can create the *server mapping* for your DB2 database.

First you need to reconnect to the DataJoiner database (with *djinst* user, from an AIX shell), as follows:

```
db2 connect to djdb
```

Then you can run the CREATE SERVER MAPPING statement with the following command:

```
db2 create server mapping from mvs63 to node \"wtsc63oe\" \
database \"DB2X\" type db2/mvs version 6.1.0 protocol \"drdaIP\"
authid paolor3 password pr3pwd
```

where:

- *mvs63* is the server name of your choice
- *wtsc63oe* is the node name previously defined
- *DB2X* is the target database name (the same that we used in the CATALOG DCS DATABASE statement in section “Configuration of DataJoiner TCP/IP services for DB2” on page 133
- *db2/mvs* is the type)
- 6.1.0 is the version number of the DB2 database
- *drdaIP* is the Data Access Module
- *paolor3/pr3pwd* are valid user and password on the OS/390 system

With the following command, create the *user mapping* for the Oracle connection:

```
db2 create user mapping from djinst to server \"MVS63\" \
authid paolor3 password pr3pwd
```

where:

- *djinst* is the local user
- *MVS63* is the server mapping previously created
- *ciprosusr* is the Oracle CIPROS user
- *ciprospwd* is his password

At the end, a locale nickname for a remote DB2 table can be created with the following command:

```
create nickname <NICKNAME> for <SERVERNAME.REMOTEUSER.TABLE>
```

that is, in our case, for example:

```
db2 create nickname TAG for mvs63.cipros.TAG
```

Note: In the previous command, *cipros* is the name of the schema the table belongs to. In our project, we have created all the tables with the *paolor2* user, then we have created an alias in the *cipros* schema for all the tables, as we have described in 5.2.1.11, “Synonyms and aliases” on page 89.

For example, you can test the connection by selecting from the TAG nickname, as follows:

```
db2 "describe select * from TAG"
```

In Appendix A.10, “nick.sh script” on page 208 we provide a shell script which creates a list of CREATE NICKNAME statements for the creation of all the nicknames for our CIPROS tables, both for Oracle and DB2 databases, starting from a file containing the list of the tables.

6.9.2 Using DataJoiner to migrate data from Oracle to DB2

As we described in 6.9.1, “Installing and configuring DataJoiner for AIX” on page 127 and using the command file created by the *nick.sh* script described in A.10, “nick.sh script” on page 208, in our DataJoiner machine (*sky)* we have created two sets of nicknames for CIPROS tables.

The first set, the names of the tables starting with an "O", is constituted by the nicknames of Oracle tables.

The second set, the names of the tables are the same of the DB2 table names, is constituted by the nicknames of DB2 tables.

Note: For some Oracle nicknames the total length of the nickname (“O” + the table name) was greater than 18. In our *nick.sh* script we have defined a list of environment variables (at the beginning of the script) containing the old and the new, shorter nicknames of Oracle tables.

At this point, we can use the INSERT/SELECT statement to migrate data from Oracle to DB2 tables, as in the following example; DataJoiner will provide for us the data conversion:

```
db2 "insert into TAG select * from OTAG"
```

Since DB2 performs the referential integrity check during INSERT statements, the order used for inserting data using DataJoiner must follow the right order provided by the referential constraints.

6.9.3 Exceptions in using DataJoiner to migrate data

In this section we describe some specific cases for which DataJoiner either should be used in a different way or cannot be used to migrate data from Oracle to DB2.

6.9.3.1 Large objects

DB2 large objects (CLOB, BLOB, DBCLOB) are not supported at the moment by DataJoiner. If your Oracle database uses LONG or LONG RAW data, you should verify that the length of the data fits into a standard DB2 VARCHAR or VARCHAR FOR BIT DATA field.

In case of LONG field mapped into DB2 VARCHAR data type, DataJoiner can be used to migrate data with a simple INSERT/SELECT statement.

If you have LONG RAW Oracle data mapped into DB2 VARCHAR FOR BIT DATA, you can use the EXPORT/IMPORT DataJoiner utility to migrate data from Oracle to DB2 table.

We show how to do this in the following example.

We have created in the Oracle database a table called LRAWTAB as follows:

```
CREATE TABLE LRAWTAB (LRAW LONG RAW);
```

Let us suppose that the source data in the Oracle table is not longer than 1000 bytes. A corresponding DB2 table (with *paolor3* user) was created as follows:

```
CREATE TABLE LRAWTAB (LRAW VARCHAR(1000) FOR BIT DATA);
```

Then, we have created the DataJoiner nicknames for the two tables as follows:

```
db2 create nickname OLRAWTAB for oradb.cipros.LRAWTAB
db2 create nickname LRAWTAB for mvs63.paolor3.LRAWTAB
```

The EXPORT statement can be executed as follows:

```
db2 "export to exp.ixf of ixf messages exp.txt select * from olrawtab"
```

Note: The EXPORT utility truncates the LONG field to 32700 bytes.

Then, the IMPORT utility towards the DB2 nickname can be executed as follows:

```
db2 "import from exp.ixf of ixf messages imp.txt insert into olrawtab"
```

As an example, we inserted into Oracle LRAWTAB table a record with the following command, issued on the Oracle machine from a *SQLPlus* command line:

```
insert into lrawtab values(hextoraw('1234567890'));
```

After migrating the data to DB2 using EXPORT/IMPORT utilities, the content of the LRAWTAB DB2 table is as follows (we used SPUFI to select from the table):

```
 SELECT HEX(SUBSTR(LRAW,1,255)) FROM LRAWTAB;                                  00270799
---------+---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+---------+
12345678904040404040404040404040404040404040404040404040404040404040404040
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE612I DATA FOR COLUMN HEADER  COLUMN NUMBER 1 WAS TRUNCATED
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Note: As for CHAR FOR BITA DATA type, DB2 fills in the trailing space of VARCHAR FOR BIT DATA fields with blanks. Furthermore to show the content of the table in hexadecimal external format using SPUFI, we had to truncate the LRAW field to 255 bytes, maximum length supported by the HEX function.

6.9.3.2 Long indexes

In 5.2.1.7, “Indexes and primary keys conversion” on page 80 we discussed the different limitations in index creation between Oracle and DB2. These limitations also apply to DataJoiner.

If your tables have indexes longer than 255 bytes, the nickname cannot be created.

In our project we have solved the problem by creating a view on the table.

In this example, RTDB_LIST table in the Oracle CIPROS database has the following structure:

```
SQL> desc rtdb_list
Name                           Null?    Type
------------------------------ -------- ----
CODE                           NOT NULL VARCHAR2(256)
ENV_NAME                       NOT NULL VARCHAR2(8)
TAG_NAME                                VARCHAR2(40)
ENG_UNT                                 VARCHAR2(10)
```

The index is constituted by the columns CODE and ENV_NAME for a total length of 264 bytes.

Since our data does not require 256 bytes for the CODE column, we have created the DB2 table by defining only 246 bytes for the CODE column length. The other fields of the table remained unchanged.

Then we have created a view on the Oracle table with the following command using *SQLPlus*:

```
create view rtdb_view as select * from rtdb_list;
```

After that, on the DataJoiner system we have created the two nicknames, one on the RTDB_VIEW Oracle view, the other one on the RTDB_LIST DB2 table, as follows:

```
db2 create nickname ORTDB_LIST for oradb.cipros.RTDB_VIEW
db2 create nickname RTDB_LIST for mvs63.cipros.RTDB_LIST
```

At this point, we can INSERT/SELECT between the two nicknames with the command:

```
db2 "insert into RTDB_LIST select * from ORTDB_LIST"
```

Chapter 7. Application conversion

In this chapter we discuss the first five tasks of our application conversion plan as described in 3.3.3.2, “Application conversion plan” on page 38. That section details how this conversion takes place, with what tools and resources:

- Proof of concept iterative process
- Programs for pilot
- Program redesign
- Program preparation
- Program conversion

7.1 Proof of concept iterative process

The proof of concept iterative process takes the application programs through all the steps needed for the conversion until they are ready for cutover. During the pilot, the iterative loop is executed a series of times until success is achieved and the conversion is complete. The actual execution times are reviewed and used for future estimates. Section 3.3.2, “Iterative tasks” on page 36 describes the use of this process during our project. Figure 69 on page 142 represents the iterative process. The iterative tasks for a conversion proof of concept include:

- Convert application programs
- Review program code
- Run tests
- Performance tuning
- Change control

7.1.1 Convert application programs

The activity related to the application programs conversion starts when the application code is at the state where it will precompile, compile, link, bind and run cleanly with SQL statements. The tasks for this activity are:

- Programs for pilot, 7.2, “Programs for pilot” on page 143 shows how we determined our pilot program
- Program redesign, 7.3, “Program redesign” on page 150 shows how we redesigned areas of our program
- Program preparation, 7.4, “Program preparation” on page 162 shows how we prepared and ran our programs
- Program conversion, 7.5, “Program conversion” on page 174 shows how we converted the programs for our project

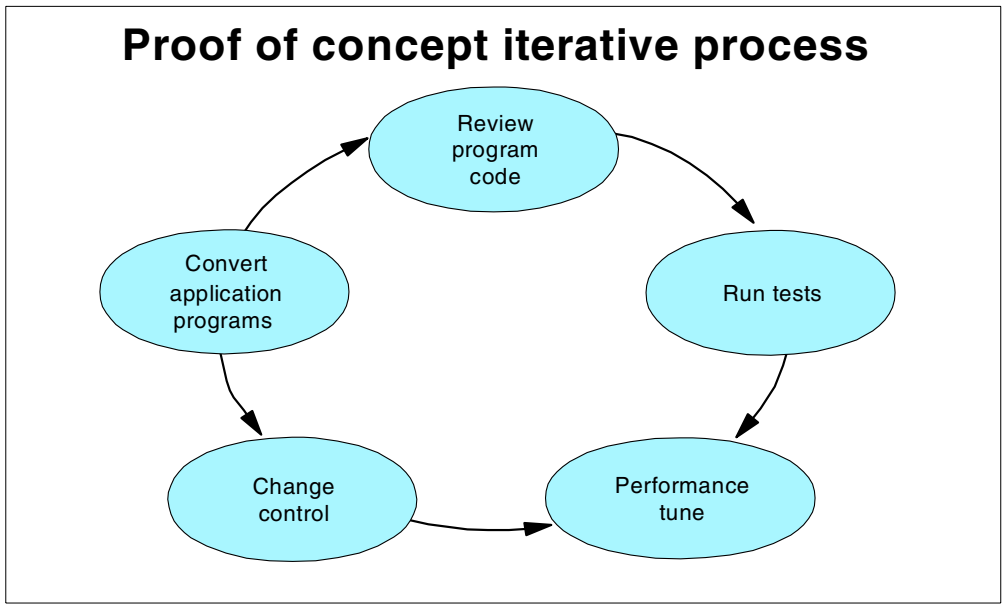


Figure 69. Proof of concept iterative process

7.1.2 Review program code

The SQL to be used in the programs may be tested with Explain or checked by the database administrator before it is system-tested. A few samples should suffice because function and performance are about to be automatically tested in 7.1.3, “Run tests” on page 142 and 7.1.4, “Performance tuning” on page 142.

7.1.3 Run tests

The run test task verifies that the program still produces the same results using the target database as it does using the source database. To do this effectively means finding the right test data. Section 8.1, “Testing” on page 183 discusses how we ran tests for our project.

7.1.4 Performance tuning

Data collected from the source system should show the CPU and elapsed time of the source system but this is not really suitable. The same tests with the same data need to be run on the both systems and the resources taken compared. It may not be necessary to do this for every possibility, but the critical transactions and important batch jobs do need to be thoroughly tested. Also, stress tests of the system ensure that the system will perform well when needed. This involves sending many transactions for processing at the same time. Section 8.2, “Performance tuning” on page 189 discusses how we performed some performance tuning for this project.

7.1.5 Change control

During stage two, any changes to the RDBMS or programs need to be done in a controlled manner. Section 8.3, “Change control” on page 189 discusses how we performed some change control for this project.

7.2 Programs for pilot

The program inventory process identifies and evaluates the source application modules needed for the target application modules. The Laboratory and Process batch application consisted of two main programs (lab_in.c and rtd_in.c). Several functions in various libraries were used. See Chapter 2, “Project scenario” on page 5 and Chapter 3, “Conversion process” on page 13 for more detail about the source application and how we determined our pilot application. The inventory process covered the areas of .c C non-SQL source code, .pc C SQL code and .h include files.

Section 7.2.1, “Source program inventory summary” on page 143 describes our source program inventory. Section 7.2.2, “Target program environment” on page 147 describes how we mapped the AIX source application environment to the OS/390 target application environment.

7.2.1 Source program inventory summary

A variety of program inventories were conducted. Table 30 provides a sample list of functions by .c/pc file.

Table 30. List of functions for each source file

.c/.pc files	List of functions for each .c/.pc file					
lab_in.c	elabora_lab	inserisci_lab	select_mtrl_lab			
sqlins.pc	CPRS_i_instrument	CPRS_i_reading	CPRS_i_reading_text	CPRS_i_sample_point	CPRS_i_sample	CPRS_i_tag
sqlerr.pc	CPRS_sql_error					
util.c	CPRS_cerca_stanza	CPRS_read_stripstr	CPRS_to_day	CPRS_open_file	CPRS_readfile	mod_str

In Figure 70 we report a sample sorted list of program functions.

```
build_lab_code
build_u_stmt
CPRS_build_log_path
CPRS_cerca_stanza
...
```

Figure 70. Sorted list of functions

In Table 31 we provide an example of a list of all files by file type.

Table 31. List of all files ordered by type

.h Include files	.c C non-SQL source files	.pc C SQL source files	.exp files
bridge.h	bridge.c	sqlerr.pc	cprsbase.exp
cbridge.h	corertd.c	sqlins.pc	cprssql.exp
ccprsrtd.h	dinexec.c	sqllog.pc	
...	...	...	

In Table 32 we show a sequence of calls to nested subroutines.

Table 32. Application nested subroutines calls

Application	Subroutines			
lab_in.c	CPRS_initbr			
		CPRS_read_cfg		readcfg.c
		CPRS_read_cfg_file		
			CPRS_cerca_stanza	util.c
	CPRS_sql_connect			
		CPRS_sql_error		sqlerr.pc
...	...	...	...	...

In Table 33 we show the list of nested header file calls by application.

Table 33. Application nested header files calls

Application	Header files					
lab_in.c						
	"lab.h"					
		<stdio.h>				
		"mbridge.h"				
			"cbridge.h"			
				<stdio.h>		
				"cprscat.h"		
					"cprs_msg.h"	
						<limits.h>
						<nl_types.h>

We identified the AIX source modules that we would use in the target environment. The AIX command "ls -R" run in the CIPROS home directory was used to generate a directory list with subdirectories which contain the AIX source modules for the Process and Laboratory Data. See Figure 71 for the related example.

```
cipros /home/cipros/ciproso> ls -R

aix
include
msg
source

./aix:
lab_in
rtd_in

./include:
bridge.h
cbridge.h
ccprsrtd.h
...

./msg:
en_US

./msg/en_US:
cprs.msg
cprs_msg.h

./source:
bridge.c
corertd.c
cprsbase.o
...
sqlerr.pc
sqlins.pc
sqllog.pc
...
```

Figure 71. AIX source module list

Figure 72 represents the program inventory summary of the AIX source modules and includes C include files (.h), C non-SQL source files (.c), C SQL source files (.pc), C make files to precompile, compile and link the C main application programs, and the message source (.msg).

.

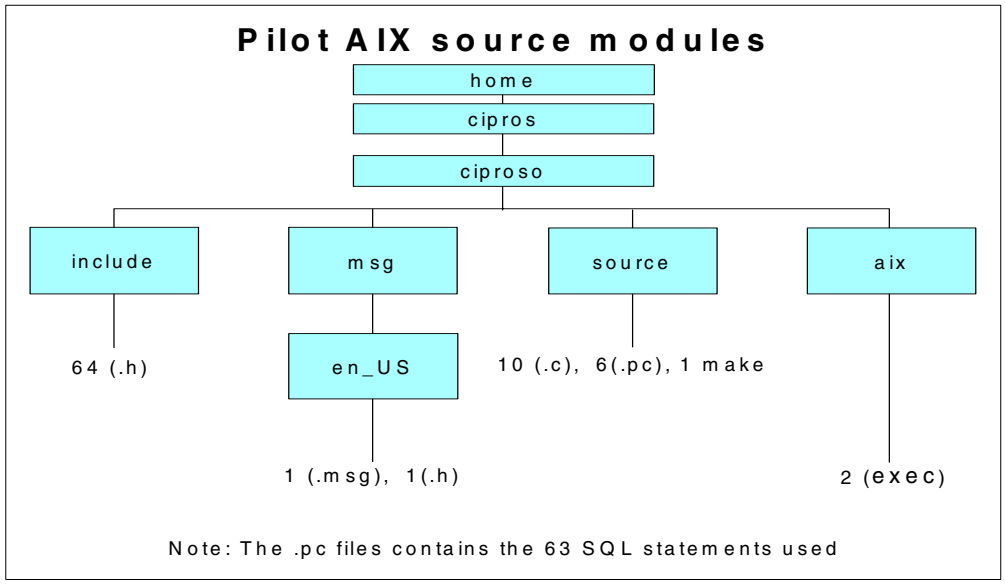


Figure 72. Process and Laboratory data source modules

Table 34 is a detailed program inventory of the Oracle ProC (.pc) files. The result of the inventory was used as a basis for application redesign efforts. In addition, the inventory was used to estimate the application development hours needed for the .pc module conversion effort.

Table 34. SQL modules inventory

Module	SQL count	Comments	Design Hours	Code Hours	Test Hours	Total hours
sqlerr.pc	0	Redesign effort - medium Code length -short Conversion-error message	8	8	8	24
sqlins.pc	24	1 to 1 - medium Code length - long Conversion-error code, pointer, date	24	16	8	48
sqllog.pc	10	Redesign effort - medium Code length-medium Conversion-error code, pointer, date, system call	24	16	8	48
sqlsel.pc	9	1 to 1 - easy Code length-medium Conversion-indicator, pointer	16	8	8	32
sqlupd.pc	10	1 to 1 - easy Code length-medium Conversion-error code, pointer, date	16	16	8	40
sqlutil.pc	10	Redesign effort-medium Code length - medium Conversion - dual, date, dynamic	24	16	8	48

7.2.2 Target program environment

In this section we describe analogies and differences when dealing with the program libraries on the OS/390 environment. Please also refer to Appendix D, “OS/390 TSO tools and tips” on page 257 for some tips on TSO usage for the first time user.

7.2.2.1 Mapping the program libraries

The target modules are located in partitioned data sets library on the OS/390 system. OS/390 partitioned data sets are not the same as the hierarchical file system structure in AIX. The AIX source module directory structure needs mapping to the OS/390 target module partitioned data set. Partitioned data sets have three qualifiers in their name, then a member name. The source module AIX directory structure with several levels is mapped to the three levels nomenclature structure for partitioned data sets. The next three figures show the three levels as reported in the target program environment on OS/390 through the ISPF/PDF Data Set List Utility.

Figure 73 shows the search at data set name level CIPROSN.C.

```
Menu  RefList  RefMode  Utilities  Help
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                            Data Set List Utility
Option ===>

   blank Display data set list              P Print data set list
       V Display VTOC information          PV Print VTOC information

Enter one or both of the parameters below:
   Dsname Level . . . CIPROSN.C
   Volume serial  . .

Data set list options
   Initial View . . . 1  1. Volume          Enter "/" to select option
                         2. Space           /  Confirm Data Set Delete
                         3. Attrib          /  Confirm Member Delete
                         4. Total

When the data set list is displayed, enter either:
  "/" on the data set list command field for the command prompt pop-up,
  an ISPF line command, the name of a TSO command, CLIST, or REXX exec, or
  "=" to execute the previous command.

 F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

Figure 73. Screen 1 for OS/390 target module Data Set List Utility

Figure 74 shows the output with the list of PDS with the specified two-level qualifier and the selection of a specific data set.

```
Menu  Options  View  Utilities  Compilers  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
DSLIST - Data Sets Matching CIPROSN.C                          Row 1 of 14
Command ===>                                                 Scroll ===> PAGE

Command - Enter "/" to select action                Message          Volume
------------------------------------------------------------------------------
         CIPROSN.C.BASEDLL                                              SBOX09
         CIPROSN.C.DBRMLIB                                              SBOX09
         CIPROSN.C.DCLGEN                                               SBOX09
         CIPROSN.C.ENVARS                                               SBOX09
         CIPROSN.C.ENVAR1                                               SBOX09
/        CIPROSN.C.INCLUDE                                              SBOX09
         CIPROSN.C.JCL                                                  SBOX02
         CIPROSN.C.LOADLIB                                              SBOX09
         CIPROSN.C.OBJ                                                  SBOX09
         CIPROSN.C.PARMS                                                SBOX09
         CIPROSN.C.SOURCE                                               SBOX09
         CIPROSN.C.SQLDLL                                               SBOX09
         CIPROSN.C.TEST                                                 SBOX09
         CIPROSN.C.VARS                                                 SBOX09
***************************** End of Data Set list *****************************
 F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cancel
```

Figure 74. Screen 2 for OS/390 target module Data Set Utility List

Figure 75 shows the list of members contained in the specified data set:

```
Menu  Functions  Confirm  Utilities  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
DSLIST           CIPROSN.C.INCLUDE                    Row 00001 of 00064
Command ===>                                            Scroll ===> DATA
           Name    Prompt       VV MM     Changed     Size  Init   Mod   ID
_________ BRIDGE
_________ CBRIDGE
_________ CCPRSRTD
_________ CDINEXEC
_________ CFINDEU
_________ CFINDMTR
_________ CFINDTYP
_________ CFINDVAL
_________ CLOGEV
_________ CMAPPING
_________ CPRSCAT
_________ CPRSMSG
_________ CPRSRTD
_________ CPRSSQL
_________ CREADCFG
_________ CSQLDEL
_________ CSQLFET
_________ CSQLINS
 F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

Figure 75. Screen 3 for OS/390 target module Data Set Utility List

7.2.2.2 Mapping the editing functions

ISPF/PDF is a powerful and comprehensive set of tools for the application programmer. The *OS/390 V2R7.0 TSO/E Command Reference,* SC28-1969-02, and *OS/390 V2R7.0 TSO/E User's Guide,* SC28-1968-01, cover the functionalities in detail. In Table 35 we report a brief summary of comparable functions between AIX and ISPF.

Table 35. AIX to OS/390 system utility mapping

Tool	AIX	OS/390
Editor	vi	ISPF Editor
List	ls	ISPF Data Set Utility Dslist
New directory/PDS	mkdir	ISPF Data Set Utility Data Set
New file/member	touch, vi	For a PDS open the member as its name in edit. For a new file, create it in the data set utility and open the file as its name in edit.
Move	mv	ISPF Data Set Utility Move/Copy
Copy	cp	ISPF Data Set Utility Move/Copy
Delete	rmdir, del	ISPF Data Set Utility Data Set

7.2.2.3 Mapping the module names

OS/390 partitioned data sets and their members have an 8 character name length limit. The AIX source module names must be changed to names of 8 characters or less.

Table 36 represents a subset of the program name cross reference.

Table 36. Program name cross reference

AIX module name in AIX hierarchical file structure	**OS/390 module name in OS/390 partitioned data set member**
sqlins.pc	sqlins
inslab.c	inslab
cprs_msg.h	cprsmsg
. . .	. . .

Note: The members in a partitioned data set do not have file extensions similar to the AIX file extensions of .pc, .h, .c etc..

The ISPF/PDF Data Set utility was used to allocate partitioned data sets and members on OS/390 for the AIX source module files. The AIX source module files were transferred to the OS/390 target module partitioned data sets via FTP.

Figure 76 represents the inventory summary of the OS/390 target modules for the Process and Laboratory Data application. It represents only a relationship diagram because PDS member attributes are different. We describe the SQL related members in 7.4.4, “JCL for precompile, compile, link, bind, and run” on page 166.

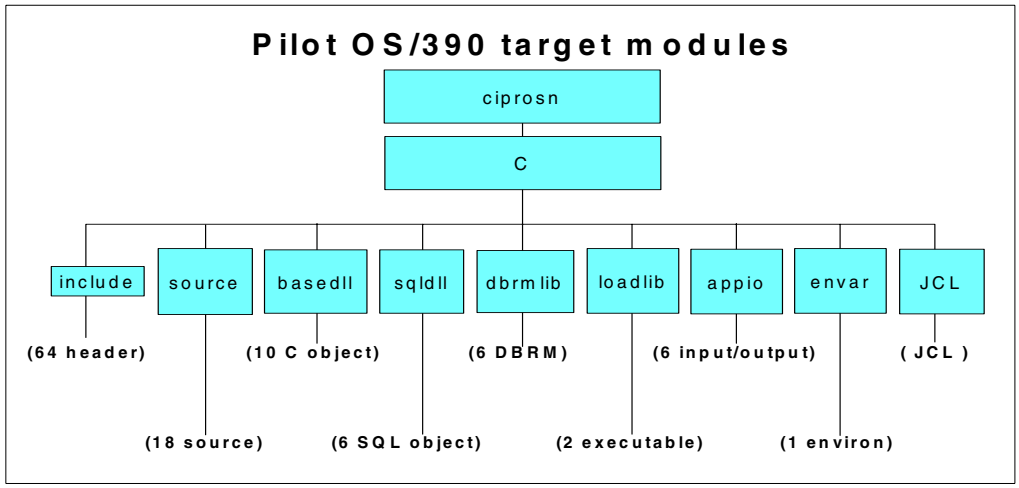


Figure 76. Process and Laboratory Data OS/390 target modules

7.3 Program redesign

Some Oracle ProC source program modules in the Process and Laboratory Data application required redesign. Table 34 on page 146 identifies which Oracle ProC files required redesign. The areas of redesign are as follows:

- Pointers
- TYPDEF
- Host variables
- Error and message handling
- File handling
- Name length limitation
- Functions

Before we detail the areas of redesign, an explanation of the prototype application and prototype JCL to test the redesign is given. In addition we show sample Oracle source code and sample DB2 target code.

7.3.1 Prototype application

In order to prototype the areas of redesign, we used the DB2 SAMPLE database and sample Phone application. Chapter 5., “Database conversion” on page 61 discusses the installation and setup of the DB2 SAMPLE database. The Phone application has a structure similar to the Process and Laboratory Data Oracle ProC module structure. For example, both the Phone application and the Laboratory and Process Data application are batch programs, use the C language and use embedded static SQL. The *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004 describes the DB2 sample applications and the environments under which each application runs. It also provides information on how to use the applications. The *DB2 UDB for OS/390 Version 6 Installation Guide*, GC26-9008 contains detailed information about the sample applications.

The Phone sample application C source code is located in the on-line sample library included with the DB2 product at *prefix*.SDSNSAMP(DSN8BD3). The program lists employee telephone numbers and updates them as requested. It also contains calls to C functions. The JCL member name for the Phone application is *prefix*.SDSNSAMP(DSNTEJ2D).

A partitioned data set was allocated for our prototype environment. It contained the JCL member in PAOLOR1.JCL.CNTL(DSNTEJ2D). Figure 77 shows how the JCL was altered to suit our system environment. The partitioned data set that contained the Phone application member is PAOLOR1.C.SOURCE (DSN8BD3). To precompile the Phone application in our prototype environment, we added the high level qualifier DSN8610.VPHONE to eliminate the message "SQLCODE -204: PAOLOR1.VPHONE is an undefined name".

7.3.2 Prototype JCL

During our redesign phase, we created JCL to precompile and compile the PDS members that contained SQL based on the sample JCL for the Phone application DSN610.QPP.SDSNSAMP(DSNTEJ2D). This would be similar to creating a **make** file on AIX to precompile and compile the source files (Oracle ProC .pc, C .c and header .h). We did this to test syntax, the host variable declarations, the include libraries, the initialization of the host variables and the SQL statements.

```
 //CPRSPCPL JOB (999,POK),'PAOLOR1',CLASS=A,MSGCLASS=T,
// NOTIFY=PAOLOR1,TIME=1440,REGION=0M
/*JOBPARM L=999,SYSAFF=SC63
//JOBLIB  DD  DSN=CEE.SCEERUN,DISP=SHR
 //        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR
 //*
 //*        STEP001: PREPARE C PROGRAM
 //*
 //STEP001  EXEC DSNHC,MEM=SQLINS,
 // COND=(4,LT),
 // PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',
 // PARM.C='/LSEARCH(''CIPROSN.C.INCLUDE'') OPTF(DD:OF)',
 // PARM.LKED='AMODE=31,RMODE=ANY,MAP'
 //PC.DBRMLIB   DD DSN=CIPROSN.DBRMLIB.DATA(SQLINS),
 //             DISP=SHR
 //PC.SYSLIB    DD DSN=CIPROSN.C.INCLUDE,
 //             DISP=SHR
//PC.SYSIN     DD DSN=CIPROSN.C.SOURCE(SQLINS),
 //             DISP=SHR
 //C.OF         DD *
   SOURCE LIST MARGINS(1,72) NESTINC(255) LONGNAME
 //LKED.SYSLMOD DD DSN=CIPROSN.RUNLIB.LOAD(SQLINS),
 //             DISP=SHR
 //LKED.RUNLIB  DD DSN=CIPROSN.RUNLIB.LOAD,
 //             DISP=SHR
 //LKED.SYSIN   DD *
      INCLUDE SYSLIB(DSNELI)
      INCLUDE RUNLIB(SQLINS)
 //*
```

Figure 77. Prototype JCL

7.3.3 Sample Oracle source code

Figure 78 represents a sample of the original CIPROS Oracle code. It is a ProC (.pc) file that contains C with embedded static SQL. This file contains all the SQL UPDATE statements for the Process and Laboratory application. Each SQL statement is contained in a C function. There is a one-to-one mapping of C function with an SQL statement.

```
#include "sqlupd.h"
EXEC SQL INCLUDE SQLCA;

int CPRS_u_sample_point(pr_sample_point rtb,char *msg,int cmt)
{
    int rc=CPRS_ORA_OK;

    EXEC SQL UPDATE SAMPLE_POINT
    SET  FACILITY_NAME=:rtb->facility_name,DESCRIPTION=:rtb->description
    WHERE  SAMPLE_PT_NAME=:rtb->sample_pt_name;

    if (((rc=CPRS_sql_error(CPRS_NO,msg))==0) && cmt==CPRS_YES)
        EXEC SQL COMMIT WORK;
   return(rc);
}
```

Figure 78. Sample Oracle source code

7.3.4 Sample DB2 target code

Figure 79 represents a sample of the target CIPROS DB2 code. It is a member in a PDS that contains C with embedded static SQL. This example contains all the SQL UPDATE statements for the Process and Laboratory application. Each SQL statement is also contained in a C function. There is a one-to-one mapping of C function with an SQL statement.

```
#include "sqlupd.h"
EXEC SQL INCLUDE SQLCA;

/* Include host variable declarations created by DCLGEN */
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL INCLUDE HOSTVARS;
EXEC SQL END DECLARE SECTION;

int CPRS_u_sample_point(pr_sample_point rtb,char *msg,int cmt)
 {
  int rc=CPRS_ORA_OK;

  /* Initialize values of host variable structure to NUL*/
  memset(DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_data,'\0',
    sizeof(DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_data));
  memset(DCLSAMPLE_POINT.DESCRIPTION.DESCRIPTION_data,'\0',
    sizeof(DCLSAMPLE_POINT.DESCRIPTION.DESCRIPTION_data));
  memset(DCLSAMPLE_POINT.FACILITY_NAME.FACILITY_NAME_data,'\0',
    sizeof(DCLSAMPLE_POINT.FACILITY_NAME.FACILITY_NAME_data));

 /* Determine length of VARCHAR host variable */
 DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_len =
  strlen(rtb->sample_pt_name);
 DCLSAMPLE_POINT.DESCRIPTION.DESCRIPTION_len =
  strlen(rtb->description);
 DCLSAMPLE_POINT.FACILITY_NAME. FACILITY_NAME_len =
  strlen(rtb->facility_name);

 /* Initialize host variables with values
    from pointer parameter */
 strncpy(DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_data,
  rtb->sample_pt_name,
  DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_len );
 strncpy(DCLSAMPLE_POINT.DESCRIPTION.DESCRIPTION_data,
   rtb->description,
  DCLSAMPLE_POINT.DESCRIPTION.DESCRIPTION_len);
 strncpy(DCLSAMPLE_POINT.FACILITY_NAME.FACILITY_NAME_data,
  rtb->facility_name,
  DCLSAMPLE_POINT.FACILITY_NAME.FACILITY_NAME_len);

/* Execute SQL */
 EXEC SQL UPDATE SAMPLE_POINT
 SET   FACILITY_NAME=:DCLSAMPLE_POINT.FACILITY_NAME,
  DESCRIPTION=:DCLSAMPLE_POINT.DESCRIPTION
 WHERE  SAMPLE_PT_NAME=:DCLSAMPLE_POINT.SAMPLE_PT_NAME;

 /* Error handling */
 if (((rc=CPRS_sql_error(CPRS_NO,msg))==0) && cmt==CPRS_YES)
  EXEC SQL COMMIT WORK;
 return(rc);
}
```

Figure 79. Sample DB2 target code

7.3.5 Pointers

C supports some data types and storage classes with no SQL equivalents, for example, pointers. You cannot use pointers as host variables. See *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004.

The Oracle source modules used pointers to structures as host variables in the embedded SQL statements. Figure 80 represents the header (.h) file that contains the definition for the pointer to the TYPDEF C structure.

```
#ifndef SQLTYPE_H
#define SQLTYPE_H
.
.
.
#define TABLE_SAMPLE_PT "SAMPLE_POINT"
typedef struct{
char sample_pt_name[CPRS_LENSPT+1];
char description[CPRS_LENDESCR+1];
char facility_name[CPRS_LENFAC+1];
} r_sample_point;

typedef r_sample_point *pr_sample_point;
#define R_SAMPLE_POINT_SIZE sizeof(r_sample_point)
.
.
.
```

Figure 80. Example 1. Source module header file

Figure 81 represents the header (.h) file that contains the function declaration for the function with the pointer.

```
#ifndef SQLUPD_H
#define SQLUPD_H
.
.
.
int CPRS_u_sample_point(pr_sample_point,char *,int);
.
.
.
```

Figure 81. Example 2. Source module function declaration

Figure 82 represents the Oracle ProC (.pc) file that contains the function and the Oracle embedded SQL using a TYPEDEF pointer to a structure as a host variable using an Oracle implicit host variable declaration.

```
#include "sqlupd.h"
.
.
.
int CPRS_u_sample_point(pr_sample_point rtb,char *msg,int cmt)
{
.
.
.
EXEC SQL UPDATE SAMPLE_POINT
SET  FACILITY_NAME=:rtb->facility_name,DESCRIPTION=:rtb->description
WHERE  SAMPLE_PT_NAME=:rtb->sample_pt_name;
.
.
.
}
```

Figure 82. Example 3. Source module SQL statement

Several changes had to be made to the DB2 target module because the host variable could not be defined as a pointer. See A.11, “gendcl.sh script” on page 209, A.12, “genmemset.sh script” on page 211, A.13, “genstrcpy.sh script” on page 212 for sample scripts we used to make the code changes. See 7.3.7, “Host variables” on page 156 for a detailed discussion about host variables and DCLGEN. See Figure 79 on page 153 for an example of the target code without the use of a pointer as a host variable. Below is a list of the changes that we made.

- DCLGEN was used to create the host variables.
- The host variable structures members had to be initialized to NUL using the C function *memset* to make sure the host variable was clean. The function *memset* cannot be used with a structure if it is an integer, char or float, only if it is a varchar.

```
memset(DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_data,'\0',
    sizeof(DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_data));
```

- The length of the VARCHAR host variable length n had to be determined using the C function strlen.

```
DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_len =
 strlen(rtb->sample_pt_name);
```

- The host variables had to be initialized from the pointer input parameter with values by using atoi for ascii to integer, atof for ascii to float, strcpy for a char of length n+1 and strncpy for a varchar of length n. If an integer or float was returned to the calling program, sprintf was used to set the return variable.

```
strncpy(DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_data,
  rtb->sample_pt_name,
  DCLSAMPLE_POINT.SAMPLE_PT_NAME.SAMPLE_PT_NAME_len );
```

- The SQL statement was executed with the DCLGEN host variable structure, not a pointer.

```
EXEC SQL UPDATE SAMPLE_POINT
SET   FACILITY_NAME=:DCLSAMPLE_POINT.FACILITY_NAME,
 DESCRIPTION=:DCLSAMPLE_POINT.DESCRIPTION
WHERE  SAMPLE_PT_NAME=:DCLSAMPLE_POINT.SAMPLE_PT_NAME;
```

Note: C and SQL differ in the way they use the word null. The C language has a null character (NUL), a null pointer (NULL), and a null statement (just a semicolon). The C NUL is a single character which compares equal to 0. The C NULL is a special reserved pointer value that does not point to any valid data object. The SQL null value is a special value that means unknown and needs IS and IS NOT NULL for comparisons.

7.3.6 TYPEDEF

There are some C data types with no SQL equivalent. C supports some data types and storage classes with no SQL equivalents, for example, TYPEDEF. See *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004.

Figure 82 on page 155 represents the Oracle ProC (.pc) file that contains the function and the Oracle embedded SQL using a TYPEDEF pointer to a structure as a host variable using an Oracle implicit host variable declaration. Figure 80 on page 154 contains the TYPEDEF declaration of the host variable as a pointer. See 7.3.5, “Pointers” on page 154 for a description of the code changes that were necessary for DB2 because TYPEDEF and pointers were used.

7.3.7 Host variables

Oracle can use an implicit declaration of a host variable. When this feature of Oracle is used, a host variable declaration section does not need to be used. See Figure 80 on page 154, Figure 81 on page 154, and Figure 82 on page 155 for an example of the source module’s use of implicit host variable declaration.

DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide, SC26-9004 states that you must explicitly declare each host variable before its first use in an SQL statement.

Figure 79 on page 153 represents how we changed the Oracle implicit use of host variable declaration to explicitly declared host variable without pointers. For a detailed discussion of the changes that were made in the code, see 7.3.5, “Pointers” on page 154.

We used DCLGEN, the declarations generator supplied with DB2 to produce a host variable DECLARE statement for C so that we did not need to code the statement ourselves. See *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004. DCLGEN generates a table declaration and host variable declaration and puts it into a member of a partitioned data set that you can include in your program. See Figure 83. The host variable declarations are based on the table declaration.

```
 /************************************************************************/
 /* DCLGEN TABLE(READING)                                                */
 /*        LIBRARY(CIPROSN.C.DCLGEN(READING))                            */
 /*        ACTION(REPLACE)                                               */
 /*        LANGUAGE(C)                                                   */
 /*        APOST                                                         */
 /* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS         */
 /************************************************************************/
  EXEC SQL DECLARE READING TABLE
           ( TAG_NAME                      VARCHAR(20) NOT NULL,
             TIMESTAMP                     TIMESTAMP NOT NULL,
             VALUE                         FLOAT,
             VALIDITY_FLAG                 INTEGER
           ) ;
/************************************************************************/
 /* C DECLARATION FOR TABLE READING                                      */
 /************************************************************************/
 struct
   { struct
       { short int  TAG_NAME_len;
         char       TAG_NAME_dataÝ20¨;
       }            TAG_NAME;
     char       TIMESTAMPÝ27¨;
     double     VALUE;
     long int   VALIDITY_FLAG;
   } DCLREADING;
 /************************************************************************/
```

Figure 83. Sample DCLGEN

If you want to include the C host variable declarations from a member of a partitioned data set, add the following SQL statement in the source code where you want to embed the statements:

```
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL INCLUDE HOSTVARS;
EXEC SQL END DECLARE SECTION;
```

HOSTVARS is the file containing the host variables.See also Figure 79 on page 153.

We created the DCLGEN for each table, then combined all the individual files into one file on AIX using the following AIX command:

```
cat * > hostvars
```

B.14 contains the JCL used to create the DCLGEN and the sample DCLGEN. Below are some points we discovered during our project:

- You cannot nest EXEC SQL INCLUDE statements.
- Do not use C #include statements to include SQL statements or C host variable declarations
- Host variable cannot be defined as hostvar[length_name+1]. The length must be numeric.
- The same column names can be in different tables because DCLGEN host variable names are table_name.column_name. The table_name and column_name combination makes the host variable name unique.
- DCLGEN maps C host variable data types to DB2 host variable data types based on the table definitions for you. This reduces data type matching errors between the host variable and the table definition.
- DCLGEN uses varchar with no null (n), and a char with a null (n+1).
- DCLGEN handles the Oracle DATE to DB2 TIMESTAMP mapping

7.3.8 Error and message handling

This section describes differences and analogies in handling application and system messages.

7.3.8.1 Oracle and DB2 error handling (SQLCA)

Oracle and DB2 for OS/390 V6 have a similar management of exceptions, with the command, available in both RDBMS:

```
EXEC SQL WHENEVER [ SQLERROR | SQLWARNING | NOT FOUND]
[CONTINUE | GOTO ]
```

Oracle has also DO and STOP commands which can be specified for an exception.

The CIPROS function that handles the error codes returned by the RDBMS does not use the standard SQL exception handling. It is instead based on a series of if/then/else C statements in order to choose the proper behavior of the program according to the result of the previous SQL statement.

A mapping between the *sqlcodes* used by the application is listed in Table 37.

Table 37. Comparison of main SQL error codes

Oracle sqlcodes	DB2 sqlcodes	Description
-1403	100	NOT FOUND
-2291	-530	Integrity constraint violation
-1	-803	Duplicate record
-1427	-811	More than one row returned
0	0	Successful execution

The SQLCA is used in both cases to retrieve from the RDBMS/C language interface both the return code and the return message of the SQL statements.

In Figure 84 on page 159 an example of C listing to access the Oracle SQLCA is shown. In our code this function is a specific C function, *CPRS_sql_error*, which is issued after the execution of the SQL statement:

```
#define SQLCA_STORAGE_CLASS
#include <sqlca.h>
...
char msg[500];
...
strncpy(msg,sqlca.sqlerrm.sqlerrmc,500);
sqlca.sqlerrm.sqlerrmc[sqlca.sqlerrm.sqlerrml]='\0';
...
if(sqlca.sqlcode == 0)
         {
        strcpy(msg,"OK!!!");
        printf("OK!!!\n");
         }
else if(sqlca.sqlcode == -1403)
        printf("NOT FOUND!\n");
else if(sqlca.sqlcode == -1)
        printf("DUPLICATE KEY!\n");
...
```

Figure 84. Example of C access to Oracle SQLCA

C programs for DB2 for OS/390 V6 have to use the DSNTIAR subroutine to retrieve the sqlcode and the return message from SQLCA, as shown in Figure 85 on page 160. In our code this function is the specific C function *CPRS_sql_error* which is issued after the execution of the SQL statement.

```
#pragma linkage(dsntiar, OS)

#define data_len 100
#define data_dim 5

char msg[500];
char msg_temp[100];
int lrecl = data_len, i;

struct error_struct {
        short int error_len;
        char error_text[data_dim][data_len];
        } error_message = {data_dim * data_len};

extern short int dsntiar(struct sqlca        *sqlca,
                         struct error_struct *msg,
                         int                 *len);
...
dsntiar(&sqlca,&error_message,&lrecl);

for (i=0;i<data_dim;i++)
        {
       memset(msg_temp, '\0', sizeof(msg_temp));
       sprintf(msg_temp,"%.100s",error_message.error_text[i]);
       strcat(msg,msg_temp);
        }
if(sqlca.sqlcode == 0)
         {
        strcpy(msg,"OK!!!");
        printf("OK!\n");
         }
else if(sqlca.sqlcode == -1403)
        printf("NOT FOUND!\n");
else if(sqlca.sqlcode == -1)
        printf("DUPLICATE KEY!\n");
...
```

Figure 85. Example of C access to DB2 SQLCA

For a complete reference to DB2 and Oracle SQLCA/SQLDA/ORACA structures, refer to *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004 and to *Oracle7 Programmer's Guide to the Pro*C Precompiler,* Part No. A16458-1.

7.3.8.2 Message handling

CIPROS programs use the AIX National Language Support (NLS) catalog facility to handle system messages.

The *NLcatopen* and *catgets* functions are used to open the message catalog file and retrieve the proper message string.

In our project we decided not to use this functionality. Instead, we handled the message file as a simple flat file that is read with *fopen* and *fgets* standard functions.

Refer to 7.3.9, “File handling” on page 161 for further details on file management

7.3.9 File handling

In order to comply with data sets and members management on the OS/390 side (for example, in OS/390 you use "."(dot) instead of "/" (slash) as the directory separator) the *fopen* functions have to be reviewed.

Functions like *fgets* or *fclose* and all the file functions that use the FILE* pointer do not need to be changed, since they refer to the already open file.

Since in our programs the file name is built by concatenating the path and the file name, and all our files have been mapped into members of a partitioned data set, the *fopen* function has been converted accordingly.

In Figure 86 an example of C management of files on AIX is shown:

```
...
FILE *fd;
char file[100], filepath[70], filename[30];
...
strcpy(filepath,"/home/cipros/rtdb/data");
strcpy(filename,"VALTAG_1999080718.DAT");

sprintf(file,"%s%s%s",filepath,"/",filename);
...
fd=fopen(file,"w");
...
```

Figure 86. File management of C for AIX

In Figure 87 we show the equivalent example of file management on OS/390:

```
...
FILE *fd;
char file[100], filepath[70], filename[30];
...
strcpy(filepath,"CIPROSN.C.TEST");
strcpy(filename,"V080718D");
...
sprintf(file,"%s%s%s%s",filepath,"(",filename,")");
...
fd=fopen(file,"w");
...
```

Figure 87. File management of C for OS/390

Note: If the complete file path is passed to the *fopen* function without enclosing it in single quotes, the *userid* running the program is appended to the path before the specified data set. In our example in Figure 87 on page 161, the file ’*userid*.CIPROSN.C.TEST(V080718D)’ is opened. If the partitioned data set does

not exist, a variable block length PDS with default settings is allocated by the program. For details, refer to *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004.

7.3.10 Name length limitation

Because of the DB2 limit of 18 characters for names of tables, columns, indexes, etc.., we changed the names of a few tables, columns and indexes to comply. We had to change the names in any place where these tables, columns, and indexes were referenced in the target module code. See Chapter 5, “Database conversion” on page 61 for information on how to find the database object names for tables, columns and indexes that incurr in this limitation.

See Figure 88 for an example of the AIX script we used to find and change the names of these objects in our target code.

```
old=OLD_TABLE_NAME
new=NEW_TABLE_NAME
for i in *.c *.h
do
echo $i
sed "s/$old/$new/g" $i > $i.new
done
```

Figure 88. Table name script

7.3.11 Functions

Two functions in the application had to be mapped from the AIX C environment to the OS/390 C environment.

On OS/390 the *__getenv()* (double underscore getenv) function was used instead of the AIX *getenv().* getenv() returns the address of the environment variable value string that has been copied into a buffer, whereas __getenv() returns the address of the actual value string in the environment variable array.

Also, on OS/390 the __dlgth() function was used instead of the AIX dayligth(). On OS/390 the __dlght() function is needed to access the thread-specific value of daylight.

7.4 Program preparation

This section details how we performed the program preparation consisting on precompile, compile, bind, link and run of our application.

7.4.1 Resources

The following manuals provided us with guidance in creating our JCL:

- *OS/390 V2R6.0 C/C++ User’s Guide*, SC09-2361-03
- *DB2 UDB for OS/390 Version 6 Application Programming and SQL*, SC26-9004

We used the following DB2 and C samples for guidance in creating our JCL for:

- DSN610.NEW.SDSNSAMP(DSNTEJ2D) JCL to precompile, compile, pre-linkedit, linkedit. DSNHC for precompile, IKJEFT01 for bind and run under TSO.
- DSN610.NEW.SDSNSAMP(DSNTIJMV) JCL to precompile (DSNHPC), compile (EDCDC120), pre-linkedit (EDCPRLK), and link (IEWL) a C program
- SYS1.PROCLIB(DSNHC) JCL to precompile (DSNHPC), compile (CBCDRIVER), pre-linkedit (EDCPRLK) and linkedit (IEWL) a C program
- SYS1.PROCLIB(EDCC) JCL to compile a C program (CBCDRIVER)

7.4.2 General program preparation process

The general program preparation process we used is detailed in the *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide V6*, SC26-9004. Figure 89 is a summary diagram of the DB2 process.

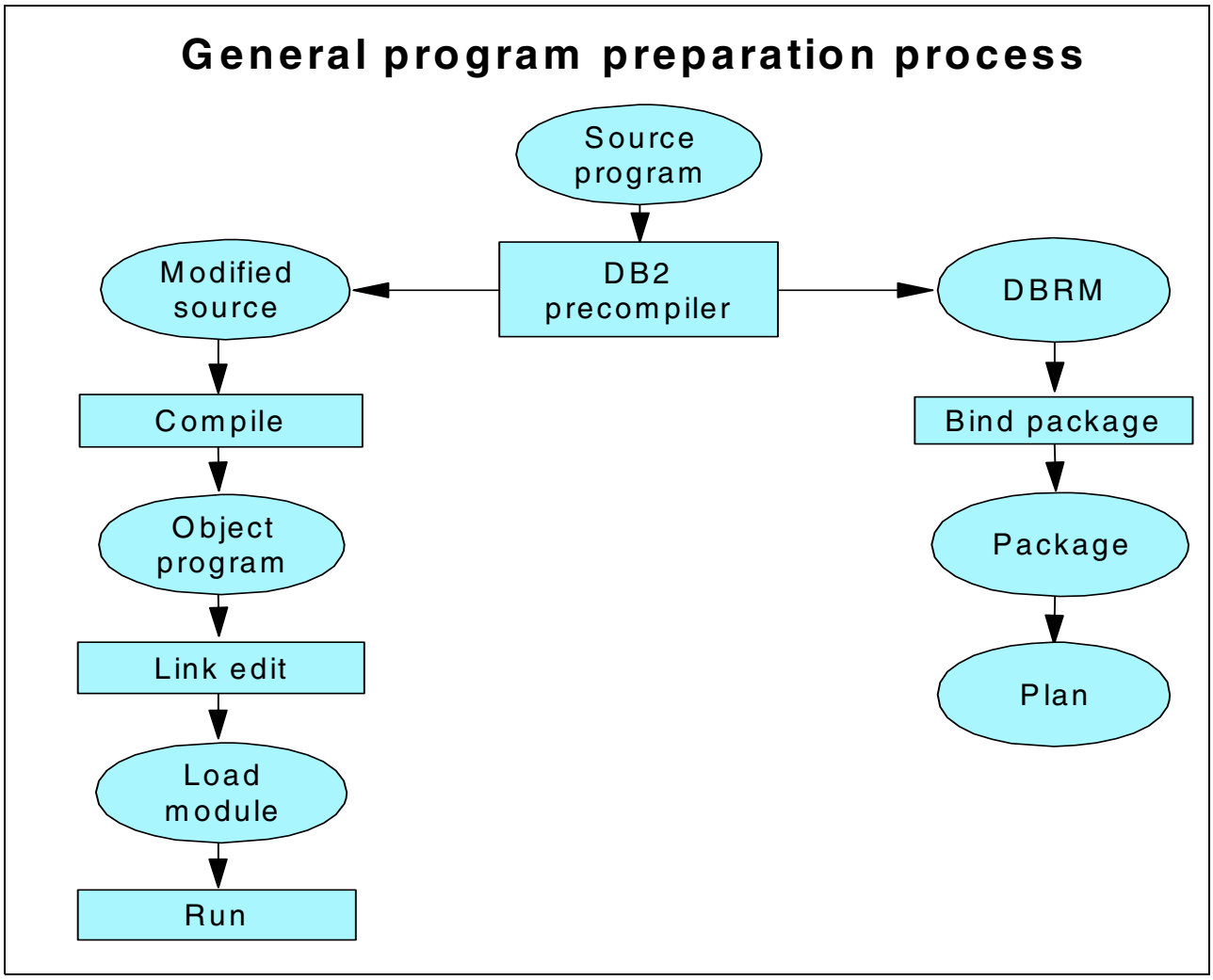


Figure 89. General program preparation process

7.4.3 Source makefile

A *makefile* has been used to build the compiling steps for the creation of the precompiled sources, the dynamic link libraries and the main executable of the programs.

It includes the standard *proc.mk* file provided by Oracle PROC interface in the directory *$ORACLE/precomp/demo/proc* in order to execute the precompile steps of the *.pc* files containing SQL statements.

In Figure 90 you can see an example of the *proc.mk* file:

```
SHELL=/bin/sh

INCLUDE=../include
CC=xlc
CFLAGS=-I. -I$(INCLUDE) -I$(ORACLE_HOME)/precomp/public
LDFLAGS=-L$(ORACLE_HOME)/lib -lsql -lsqlnet -lncr -lclient -lcommon -lgeneric -lepc
-lnlsrtl3 -lc3v6 -lcore3 -lclntsh -lm
CPPFLAG=
PROFILING=-O
SOURCELIST=-qflag=i:i -qinfo
DEBUGFLS=$(PROFILING) $(SOURCELIST) $(CPPFLAG)

CONFIG=./oracle.cfg
PROC=$(ORACLE_HOME)/bin/proc

$(EXE): $(OBJS)
      $(CC) $(LDFLAGS) -o $@ $@.o $(OBJS) $(DEBUGFLS) -DAIX -bloadmap

# Suffix rules
.SUFFIXES:  .o .c .pc

.pc.c:
      $(PROC) $(PROFLAGS) config=$(CONFIG) include=$(INCLUDE) iname=$*.pc

.c.o:
      $(CC) $(CFLAGS) -c $*.c $(DEBUGFLS)  -DAIX

.pc.o:
      $PROC) $(PROFLAGS) config=$(CONFIG) include=$(INCLUDE) iname=$*.pc
      $(CC) $(CFLAGS) -c $*.c $(DEBUGFLS) -DAIX
```

Figure 90. proc.mk file

- The *oracle.cfg* file contains some configuration information for the Oracle PROC precompiler, such as:

```
RELEASE_CURSOR=YES
CODE=ANSI_C
AUTO_CONNECT=YES
MAXOPENCURSORS=100
```

- The indentation of configuration stanzas must be done with tabulation characters (as in the C *makefile* files)
- the proc.mk file has to be included into the *makefile* file with the following *include* statement:

```
include ./proc.mk
```

Let us give an example of the usage of a makefile. We have not used our original makefile because it contains statements and links to CIPROS programs and applications which are not within the scope of work of our project.

As in our application, you may have a dynamic link library (DLL) constituted by only SQL functions (we call it *libsql.a*) and a DLL constituted by standard and user defined C functions (*libstd.a*).

The first library is constituted by *file1.pc* and *file2.pc*. The second library is constituted by *file3.c* and *file4.c*.

Your main program is called *mainprg.c*. In Figure 91 you can see the *makefile* file to compile your program:

```
include ./proc.mk

LIBP=-L.
LIBSTD=-lstd
LIBSQL=-lsql

EXECMD=$(CC) $(LDFLAGS) -o $@ $@.o $(LIBP) $(LIBSTD) $(LIBSQL) $(DEBUGFLS)

sql.o:./sql.exp file1.o file2.o
      ld -o sql.o file1.o file2.o $(LDFLAGS) \
      -bE:./sql.exp \
      -bM:SRE \
      -e Entry_SQL_Function -lc -lm

std.o:.std.exp file3.o file4.o
      ld -o std.o file3.o file4.o $(LDFLAGS) \
      -bE:.std.exp \
      -bM:SRE \
      -e Entry_STD_Function -lc -lm $(LIBSQL)

libsql.a: sql.o
      rm -f libsql.a
      ar vro libsql.a sql.o

libstd.a: std.o
      rm -f libstd.a
      ar vro libstd.a std.o

mainprg: mainprg.o
$(CC) -o $@ $@.o $(LIBP) $(LIBSTD) $(LIBSQL) $(DEBUGFLS)
```

Figure 91. Example of a makefile file

- The indentation of configuration stanzas must be done with tabulation characters
- The files sql.exp and std.exp contain a lists of all the external functions used by the main program and contained into the corresponding source files
- The environment variables CC, LDFLAGS and DEBUGFLS are defined into the *proc.mk* file
- The Entry_SQL_Function and Entry_STD_Function functions are entry points of the executable output file
- For the execution of the *make* statements, you can simply run the command (from an AIX shell):

```
make stanza_name
```

First, the object code of the two libraries must be compiled, as follows:

```
make sql.o
make std.o
```

Then, the two dynamic libraries must be compiled executing the ar command, as in the following statements:

```
make libsql.a
make libstd.a
```

At the end, the main program can be compiled activating with the command:

```
make mainprg
```

With the provided example we cannot cover all the possible and more optimized facilities that the C language and the AIX programming interface offer. This is just one of several possible alternatives. Refer to *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*, SC23-2533 for details on AIX programming interface.

7.4.4 JCL for precompile, compile, link, bind, and run

We used the OS/390 V2R6.0 C language compiler. AIX uses a makefile for precompile, compile, link and run. OS/390 uses JCL to precompile, compile, link and run the application. See Appendix B, “Sample DB2 for OS/390 jobs” on page 215 for the BASECOMP, SQLCOMP, CGLOB, and RTDIN/LABIN JCL described in this section.

Figure 92 shows a summary of our program preparation process. The boxes in the figure represent the JCL PDS members that we used for that process. For example, the BASECOMP PDS member contained the JCL to compile non-SQL C source code. The circles in Figure 92 represent either input or output PDS to the JCL. For example, the SOURCE PDS members contained C source code. In general, our process consisted of several steps:

1. Step 1a. (JCL BASECOMP) was to compile non-SQL source code and Step 1b.(JCL SQLCOMP) was to precompile/compile SQL source code.
2. Step 2 (JCL CGLOB) was to compile the 2 main application programs and link the non-SQL, SQL and application object code.
3. Step 3 (Interactive BIND) was to BIND the DBRM’s for each of the SQL members.
4. Step 4a.(RTDIN) and 4b.(LABIN) was to run the executable application code.

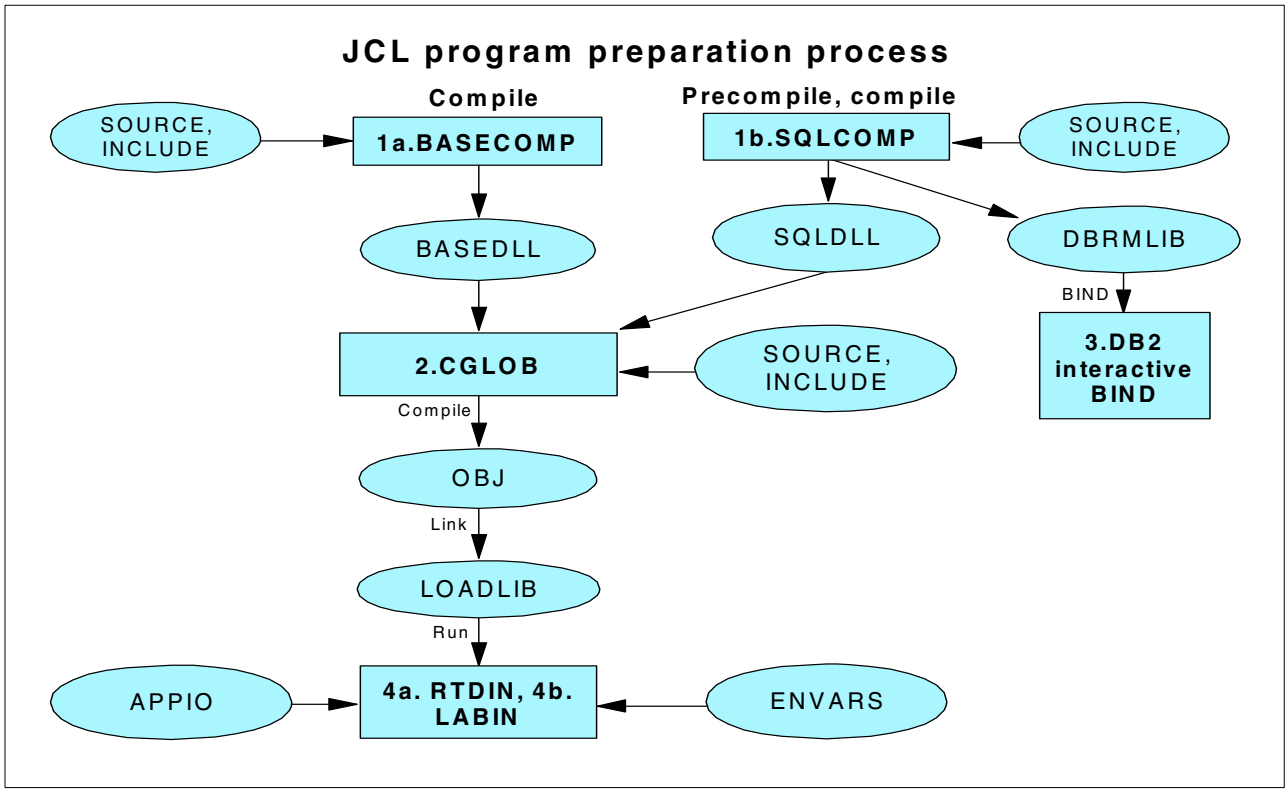


Figure 92. JCL program preparation process

Table 38 lists the PDS members containing the JCL.

Table 38. Program preparation JCL PDS members

Members of JCL PDS	Purpose	Input PDS	Output PDS	Proc/ Program in JCL
BASECOMP	Compile non-SQL source	SOURCE, INCLUDE	BASEDLL	CBCDRIVER
SQLCOMP	Precompile, compile SQL source	SOURCE, INCLUDE	SQLDLL, DBRMLIB	DSNHPC
CGLOB	Compile, prelink, link 2 main applications	SOURCE, INCLUDE, BASEDLL, SQLDLL	OBJ, LOADLIB	EDCC
RTDIN	Run rtdin application	LOADLIB, ENVARS	APPIO	RTDIN
LABIN	Run labin application	LOADLIB, ENVARS, APPIO	APPIO	LABIN

Table 39 describes the PDS libraries needed for program preparation.

Table 39. Program preparation PDS libraries

PDS	Purpose
BASEDLL	Contains object code generated by BASECOMP JCL for non-SQL source code and serves as input for CGLOB JCL
DBRMLIB	Contains a DBRM for each SQL source code generated by SQLCOMP JCL and serves as input to the DB2 Interactive BIND process
ENVARS	Flat file containing input for environment variables used when the RTDIN and LABIN applications run
INCLUDE	Contains DCLGEN and include files used as input during the compile process in the BASECOMP, SQLCOMP and CGLOB JCL
JCL	Contains JCL to prepare the program
LOADLIB	Contains the executable code generated by the CGLOB JCL for the RTDIN and LABIN applications
OBJ	Contains the object code generated by the CGLOB JCL for the RTDIN and LABIN applications and serves as input to the CGLOB JCL link step to create the executable code
SOURCE	Contains non-SQL and SQL source code used as input during the compile process in the BASECOMP, SQLCOMP an d CGLOB JCL
SQLDLL	Contains object code generated by SQLCOMP JCL for SQL source code and serves as input for the CGLOB JCL
APPIO	Contains input/output files used when the RTDIN and LABIN application executables generated by CGLOB JCL run

7.4.5 Precompile

This section discusses the details of the JCL we used to precompile the SQL source code. This section references the SQLCOMP JCL in B.2, “JCL for SQL function precompile and compile” on page 216.

7.4.5.1 Precompile parameters

The sample code below is an excerpt from the SQLCOMP JCL. It lists the precompile parameter options that we used. This section discusses the parameters that we changed for our application.

```
//PC      EXEC PGM=DSNHPC,
//         PARM='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',
//         REGION=4096K
```

C Compiler

The C language is defined to the precompiler by the HOST parameter.

```
HOST(C)
```

Margin

The MARGINS parameter specifies that the SQL code statements will be in columns 1 through 72.

```
MARGINS(1,72)
```

SQLCA
The STDSQL parameter is set to NO to specify that the SQLCA will be included explicitly in the C program by using the statement "EXEC SQL INCLUDE SQLCA;".

```
STDSQL(NO)
```

7.4.5.2 Precompiler DD statements

This section discusses the precompile DD statements. The DD statements are:

- STEPLIB determines where the precompile EXEC PGM member DSNHPC is located.
- DBRMLIB determines where to put the DBRM output members as a result of the precompile
- SYSLIB determines where the source code INCLUDE members are located
- SYSIN determines where the source code member for the precompile is located

```
//STEPLIB  DD  DISP=SHR,DSN=DSN610B.SDSNEXIT.SC63
//         DD  DISP=SHR,DSN=DSN610B.SDSNLOAD
//DBRMLIB  DD DSN=CIPROSN.C.DBRMLIB(SQLLOG),
//            DISP=SHR
//SYSLIB   DD DSN=CIPROSN.C.INCLUDE,
//            DISP=SHR
//SYSIN    DD DSN=CIPROSN.C.SOURCE(SQLLOG),
//            DISP=SHR
```

7.4.6 Compiler

This section discusses the details of the JCL we used to compile the SQL source code. This section will reference the SQLCOMP JCL in B.2, “JCL for SQL function precompile and compile” on page 216.

7.4.6.1 Compiler parameters

The sample code below is an excerpt from the SQLCOMP JCL. It lists the compile parameter options that we used. This section discusses the parameters that we changed for our application.

```
//C       EXEC PGM=CBCDRVR,COND=(4,LT,PC),REGION=4096K,
//        PARM='/LSEARCH(''CIPROSN.C.INCLUDE'') OPTF(DD:OF)'
...
//C.OF         DD *
   SOURCE LIST MARGINS(1,72) NESTINC(255) LONGNAME RENT
```

Multiple parameters
The OPTF option directs the compiler to look for compiler options in the file specified by filename.

```
OPTF(DD:OF)
```

...

```
//C.OF         DD *
   SOURCE LIST MARGINS(1,72) NESTINC(255) LONGNAME RENT
```

- Margins

The MARGINS option specifies the columns in the input record that are to be scanned for input to the compiler.

```
MARGINS(1,72)
```

- Function name length

  The LONGNAME option generates untruncated and mixed case external names in the object module produced by the compiler for functions with non-C++ linkage.

  ```
  LONGNAME
  ```

- Reentrant

  The RENT option specifies that the compiler is to take code that is not naturally reentrant and make it reentrant.

  ```
  RENT
  ```

Include directive

Several areas were addressed to use the include user members. They are discussed below:

- The LSEARCH option directs the preprocessor to look for the user include files in the specified PDS library. Although the include PDS member does not end in ".h", you may specify the include directive in the source member as "#include "sqlupd.h".

  ```
  /LSEARCH(''CIPROSN.C.INCLUDE'')
  ...
  //SYSLIB   DD DSN=CIPROSN.C.INCLUDE,
  //           DISP=SHR
  ```

- If you alter the include member, remember to alter the source member or delete the object member so the compiler will re-compile. For example, add a space to the source member.
- The NESTINC option specifies the number of nested include files to be allowed in your source program.

  ```
  NESTINC(255)
  ```

7.4.6.2 Compiler DD statements

This section discusses the compile DD statements. The DD statements in Figure 93 are:

- STEPLIB determines where the compile EXEC PGM member CBCDRVR and C compiler are located.
- SYSLIB determines where the necessary C INCLUDE members are located
- SYSLIN determines where the modified source code member from the precompiler for the compile is located

```
//STEPLIB  DD  DSN=CBC.SCBCCMP,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//SYSLIB   DD  DSN=CEE.SCEEH.H,DISP=SHR
//         DD  DSN=CEE.SCEEH.SYS.H,DISP=SHR
//         DD  DSN=DSN610B.SDSNC.H,DISP=SHR
//         DD  DSN=GDDM.SADMSAM,DISP=SHR
//C.OF         DD *
  SOURCE LIST MARGINS(1,72) NESTINC(255) LONGNAME RENT
//SYSLIN   DD  DSN=CIPROSN.C.SQLDLL(SQLLOG),DISP=SHR
```

Figure 93. Compile DD statements

7.4.7 Link

This section discusses the details of the JCL we used to link the object code for the non-SQL, SQL and application code. This section will reference the CGLOB JCL in B.3, “JCL for compile, prelink and link of main programs” on page 217. This section discusses the compile DD statements. The DD statements are in Figure 94.

- SYSLMOD determines which PDS the executable code for the application will be placed after the link edit.
- SYSIN specifies the input PDS members for the link edit.

```
//LKED.SYSLMOD   DD DSN=CIPROSN.C.LOADLIB(RTDIN),DISP=SHR
//LKED.RUNLIB    DD DSN=CIPROSN.C.LOADLIB,DISP=SHR
//LKED.SYSIN     DD *
    INCLUDE SYSLIB(DSNELI)
    INCLUDE RUNLIB(RTDIN)
```

Figure 94. Link Edit JCL

7.4.8 BIND

This section discusses the process we used to interactively BIND the DBRM produced by the precompiler to a plan or package before the DB2 application can run. The bind process we used is detailed in the *DB2 for UDB for OS/390 Application Programming and SQL Guide V6*, SC26-9004. Figure 95 on page 172, Figure 96 on page 172 and Figure 97 on page 173 are sample screens and options we selected by using DB2 Interactive for the bind process.

Figure 95 displays our selection of option 5 in DB2 Interactive to perform the bind process.

```
 DB2I PRIMARY OPTION MENU          SSID: DB2X
COMMAND ===>  5

Select one of the following DB2 functions and press ENTER.

 1  SPUFI                  (Process SQL statements)
 2  DCLGEN                 (Generate SQL and source language declarations)
 3  PROGRAM PREPARATION    (Prepare a DB2 application program to run)
 4  PRECOMPILE             (Invoke DB2 precompiler)
 5  BIND/REBIND/FREE       (BIND, REBIND, or FREE plans or packages)
 6  RUN                    (RUN an SQL program)
 7  DB2 COMMANDS           (Issue DB2 commands)
 8  UTILITIES              (Invoke DB2 utilities)
 D  DB2I DEFAULTS          (Set global parameters)
 X  EXIT                   (Leave DB2I)




PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=RCHANGE
PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

Figure 95. IBIND screen one

Figure 96 displays our selection of option 4 in DB2 Interactive BIND to add a package.

```
 BIND/REBIND/FREE               SSID: DB2X
 COMMAND ===>   4

 Select one of the following and press ENTER:

  1  BIND PLAN             (Add or replace an application plan)

  2  REBIND PLAN           (Rebind existing application plan or plans)

  3  FREE PLAN             (Erase application plan or plans)

  4  BIND PACKAGE           (Add or replace a package)

  5  REBIND PACKAGE        (Rebind existing package or packages)

  6 REBIND TRIGGER PACKAGE (Rebind existing trigger package or packages)

  7  FREE PACKAGE          (Erase a package or packages)



 PF 1=HELP     2=SPLIT    3=END      4=RETURN    5=RFIND    6=RCHANGE
 PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

Figure 96. BIND screen two

Figure 97 displays the options and defaults that we chose to BIND the package. We specified "cipros1" as our collection id in option 2, "DBRM" as the package source in option 3, "sqlupd" as our member in option 4 and "ciprosn.c.dbrmlib" as the location of our DBRM.

```
 BIND PACKAGE                "DOWN    " is not active
COMMAND ===>

Specify output location and collection names:
 1  LOCATION NAME ..... ===>                    (Defaults to local)
 2  COLLECTION-ID ..... ===> cipros1            (Required)
Specify package source (DBRM or COPY):
 3   DBRM:         COPY:        ===> DBRM       (Specify DBRM or COPY)
 4  MEMBER    or  COLLECTION-ID ===> sqlupd
 5  PASSWORD  or  PACKAGE-ID .. ===>
 6  LIBRARY   or  VERSION ..... ===> ciprosn.c.dbrmlib
                                              (Blank, or COPY version-id)
 7  ........  --  OPTIONS ..... ===>           (COMPOSITE or COMMAND)
Enter options as desired:
 8  CHANGE CURRENT DEFAULTS? .. ===> NO         (NO or YES)
 9  ENABLE/DISABLE CONNECTIONS? ===> NO         (NO or YES)
10  OWNER OF PACKAGE (AUTHID).. ===>            (Leave blank for primary ID)
11  QUALIFIER ................. ===>            (Leave blank for OWNER)
12  ACTION ON PACKAGE .... .... ===> REPLACE    (ADD or REPLACE)
13  INCLUDE PATH?.............. ===> NO         (NO or YES)
14  REPLACE VERSION ........... ===>
                                              (Replacement version-id)
PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=RCHANGE
PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

Figure 97. BIND screen three

7.4.9 Run

This section discusses the details of the JCL we used to run the application executable code. This section will reference the RTDIN JCL in Appendix B.4, “JCL for running the main programs RTDIN and LABIN” on page 218. Below is a sample section of the RTDIN JCL.

IKJEFT01 is the EXEC PGM we used to run the application. We used *__getenv()* (double underscore getenv) instead of *getenv()* because getenv() returns the address of the environment variable value string that has been copied into a buffer, whereas __getenv() returns the address of the actual value string in the environment variable array. The MYVARS DD statement specifies in which PDS the environment is kept. The LIB statement specifies where the application executables are kept. The program we are running in this JCL is specified in the PROGRAM parameter.

```
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//MYVARS    DD  DSN=CIPROSN.C.ENVARS,DISP=SHR
...
DSN SYSTEM(DB2X)
RUN  PROGRAM(RTDIN) PLAN(CIPROS1)  -
     LIB('CIPROSN.C.LOADLIB') -
     PARM ('ENVAR("_CEE_ENVFILE=DD:MYVARS")/V060308D')
```

7.4.10 SDSF usage for output messages

Extensive use of SDSF was made to monitor the execution of our jobs and to verify the results of the output held in the JES2 output queue. Refer to D.2, “SDSF output messages” on page 262 for a demostration of the process.

7.5 Program conversion

This section will discuss the program conversion consisting on database programming methods, the SQL language and the special characters tabs and brackets.

7.5.1 Database programming methods

The scope of our project included embedded SQL and dynamic SQL programming methods. This section contains a detailed discussion of these methods. *See DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004 for details and sample code on the stored procedures and ODBC programming methods.

7.5.1.1 Embedded SQL

For most DB2 users, static SQL (embedded in a host language program and bound before the program runs) provides a straight-forward, efficient path to DB2 data. You can use static SQL when you know before run time what SQL statements your application needs to execute.

We used embedded SQL and OS/390 C for target modules. The reasons were that the programming method was most similar to the source, the skills available in the project team, the scope of effort of the project, and the solutions chosen for the architectural environment. For a more detailed discussion with sample code on our target embedded static SQL with C code, see 7.3, “Program redesign” on page 150.

7.5.1.2 Dynamic SQL

Dynamic SQL allows the applications to build SQL statements and prepare them at a run-time level. The major benefit is that there is no precompilation or binding required prior to the execution of the application. DB2 provides support for dynamic SQL statements such as PREPARE, EXECUTE, EXECUTE IMMEDIATE, and DESCRIBE.

CIPROS applications use only the EXECUTE IMMEDIATE statement. The syntax is identical in both Oracle and DB2, so we did not need to change our source code, as we show in the following example:

```
int rc=0;

EXEC SQL BEGIN DECLARE SECTION;
        struct { short len;
                 char  data[256];
               } stmt;
EXEC SQL END   DECLARE SECTION;

strcpy(statement,"UPDATE TAG SET STATUS='OK' WHERE TAG_NAME='T1000'");
memset(stmt.data,'\0',sizeof(stmt.data));
stmt.len=strlen(statement);
strncpy(stmt.data,statement,stmt.len);

EXEC SQL EXECUTE IMMEDIATE :stmt;
```

Refer to *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004, for further information on dynamic statements execution.

7.5.1.3 Stored procedures

There are two ways to implement stored procedures with DB2 UDB for OS/390 Version 6: external and internal SQL stored procedure. An external stored procedure and an internal SQL procedure differ in the way that they specify the code for the stored procedure. An external stored procedure definition specifies the name of the stored procedure program. An SQL procedure definition contains the source code for the stored procedure.

External stored procedures

First, an external stored procedure is a compiled program, stored at a DB2 local or remote server, that can execute SQL statements. A typical stored procedure contains two or more SQL statements and some manipulative or logical processing in a host language. A client application program uses the SQL statement CALL to invoke the stored procedure. See *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004 for a detailed discussion on external stored procedures.

Internal stored procedures

Second, an internal stored procedure is a stored procedure in which the source code for the procedure is in an SQL CREATE PROCEDURE statement. The part of the CREATE PROCEDURE statement that contains the code is called the procedure body.

Creating an SQL procedure involves writing the source statements for the SQL procedure, and defining the SQL procedure to DB2. There are two ways to create an SQL procedure:

- Use the IBM DB2 Stored Procedure Builder product to specify the source statements for the SQL procedure, define the SQL procedure to DB2 and prepare the SQL procedure for execution.
- Write a CREATE PROCEDURE statement for the SQL procedure. Then define the SQL procedure to DB2 and create an executable procedure.

The SQL Procedures language is used to write the CREATE PROCEDURE statement that contains the procedure body. A procedure body consists of a single simple or compound statement that you can include in a procedure body. Some types of statements that you can include in a procedure body are:

- Assignment statement
- CASE statement
- IF statement
- LOOP statement
- REPEAT statement
- WHILE statement
- Compound statement
- SQL statement

Refer to *Developing Cross-Platform DB2 Stored Procedures: SQL Procedures and the DB2 Stored Procedure Builder*, SG24-5485 for more information on how to develop stored procedures with DB2.

7.5.1.4 ODBC

DB2 Open Database Connectivity (ODBC) is IBM's callable SQL interface by the DB2 family of products. It is a 'C' and 'C++' application programming interface for relational database access, and it uses function calls to pass dynamic SQL statements as function arguments. It is an alternative to embedded dynamic SQL, but, unlike embedded SQL, it eliminates the need for precompiling.

DB2 ODBC is based on the Microsoft(TM) Open Database Connectivity (ODBC) specification, and the X/Open Call Level Interface specification. These specifications were chosen as the basis for the DB2 ODBC in an effort to follow industry standards and to provide a shorter learning curve for those application programmers already familiar with either of these data source interfaces. In addition, some DB2 specific extensions were added to help the DB2 application programmer to specifically exploit DB2 features.

ODBC lets you access data through ODBC function calls in your application. You execute SQL statements by passing them to DB2 through a ODBC function call. ODBC not only eliminates the need for precompiling but also for binding your application, and increases its portability. If you are writing your applications in Java, you can use Java Database Connectivity (JDBC) application support to access DB2. JDBC is similar to ODBC but is designed specifically for use with Java and is therefore a better choice than ODBC for making DB2 calls from Java applications. For more information on ODBC, refer to the *DB2 UDB for OS/390 Version 6 ODBC Guide and Reference*, SC26-9005.

7.5.2 SQL statements

The target code for our project included all the following standard SQL data manipulation language statements.

- INSERT
- SELECT
- UPDATE
- COMMIT
- ROLLBACK
- Indicator variable

Generally, the Oracle SQL source code mapped to the DB2 target code. We had to change the DB2 target EXEC SQL statement to handle the non-pointer host variables. See 7.3.5, “Pointers” on page 154. Below are samples of the source and target code EXEC SQL statements.

7.5.2.1 INSERT

Below is a sample of the Oracle embedded SQL with C SQL statement INSERT:

```
EXEC SQL INSERT INTO ANALYSIS_METHOD
(ANLY_METH_NAME,DESCRIPTION)
VALUES
(:rtb->anly_meth_name,:rtb->description);
```

Below is a sample of the DB2 embedded SQL with C SQL statement INSERT

```
EXEC SQL INSERT INTO ANALYSIS_METHOD
 (ANLY_METH_NAME,
  DESCRIPTION)
 VALUES
 (:DCLANALYSIS_METHOD.ANLY_METH_NAME,
  :DCLANALYSIS_METHOD.DESCRIPTION);
```

7.5.2.2 SELECT

Below is a sample of the Oracle embedded SQL with C SQL statement SELECT:

```
EXEC SQL SELECT MTRL_NAME
         INTO :mtrl_name :ind_mtrl_name
         FROM MAP_MATERIAL
         WHERE MTRL_ENV=:mtrl AND ENV_NAME=:env;
```

Below is a sample of the DB2 embedded SQL with C SQL statement SELECT:

```
EXEC SQL SELECT MTRL_NAME
INTO :MTRL_NAME :ind_mtrl_name
FROM MAP_MATERIAL
WHERE MTRL_ENV=:MTRL AND ENV_NAME=:ENV_NAME;
```

7.5.2.3 UPDATE

Below is a sample of the Oracle embedded SQL with C SQL statement UPDATE:

```
EXEC SQL UPDATE ANALYSIS_SPEC
  SET HI_VALUE=:rtb->hi_value, LO_VALUE=:rtb->lo_value
  WHERE  MTRL_NAME=:rtb->mtrl_name AND STD_RESLT_NAME=:rtb->std_reslt_name
```

Below is a sample of the DB2 embedded SQL with C SQL statement UPDATE:

```
EXEC SQL UPDATE ANALYSIS_SPEC
  SET  HI_VALUE=:DCLANALYSIS_SPEC.HI_VALUE,
   LO_VALUE=:DCLANALYSIS_SPEC.LO_VALUE
  WHERE   MTRL_NAME=:DCLANALYSIS_SPEC.MTRL_NAME AND
    STD_RESLT_NAME=:DCLANALYSIS_SPEC.STD_RESLT_NAME;
```

7.5.2.4 COMMIT

Below is a sample of the Oracle embedded SQL with C SQL statement COMMIT:

```
EXEC SQL COMMIT WORK;
```

Below is a sample of the DB2 embedded SQL with C SQL statement COMMIT:

```
EXEC SQL COMMIT WORK;
```

7.5.2.5 ROLLBACK

Below is a sample of the Oracle embedded SQL with C SQL statement ROLLBACK:

```
EXEC SQL ROLLBACK WORK;
```

Below is a sample of the DB2 embedded SQL with C SQL statement ROLLBACK:

```
EXEC SQL ROLLBACK WORK;
```

7.5.2.6 Indicator variables

Below is a sample of the Oracle embedded SQL with C SQL statement indicator Variable:

```
EXEC SQL SELECT TAG_NAME,ENG_UNT
 INTO :tag_name :ind_tag_name, :eng_unt :ind_eng_unt
 FROM RTDB_LIST
 WHERE CODE=:rtb->code AND ENV_NAME=:rtb->env_name;
```

Below is a sample of the DB2 embedded SQL with C SQL statement indicator variable:

```
EXEC SQL SELECT TAG_NAME, ENG_UNT
  INTO  :DCLRTDB_LIST.TAG_NAME :ind_tag_name,
  :DCLRTDB_LIST.ENG_UNT :ind_eng_unt
  FROM RTDB_LIST
  WHERE  CODE=:DCLRTDB_LIST.CODE AND
   ENV_NAME=:DCLRTDB_LIST.ENV_NAME;
```

7.5.3 Tabs

Tab characters (x'05') have to be changed to space (x'40'). You may change the tab characters to spaces in two ways:

1. On OS/390

   Edit the PDS member in ISPF. On the ISPF editor command line, as reported in Figure 98, enter "hex on" in the command line to see the hex values for the tab and space characters.

```
 File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
EDIT       CIPROSN.C.SOURCE(BRIDGE) - 01.00               Columns 00001 00072
Command ===> hex on                                          Scroll ===> CSR
****** **************************** Top of Data ******************************
000001 #include "bridge.h"
000002 #define _POSIX1_SOURCE 2
000003 #include <env.h>
000004
000005 int CPRS_initbr(
000006  char *pgm,
000007  char *filename,
000008  char *room,
000009  pr_cfg pcfg,
000010  pr_log_event plogev,
000011  pr_log_errlines plog_errlines,
000012  char *endmsg,
000013  int initlog)
000014 {
```

Figure 98. Turning hex on

You can now enter the command "c x'05' x'40' all" to change the tabs to space characters as reported in Figure 99. Then you can enter the command “hex off” .

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
EDIT       CIPROSN.C.SOURCE(BRIDGE) - 01.00               Columns 00001 00072
Command ===>  c x'05' x'40' all                             Scroll ===> CSR
****** **************************** Top of Data ******************************
------------------------------------------------------------------------------
000001 #include "bridge.h"
       78989A884789888848744444444444444444444444444444444444444444444444444444
       B95334450F299475B8F00000000000000000000000000000000000000000000000000000
------------------------------------------------------------------------------
000002 #define _POSIX1_SOURCE 2
       788889846DDECEF6EDEDCC4F44444444444444444444444444444444444444444444444444
       B4569550D762971D26493502000000000000000000000000000000000000000000000000
------------------------------------------------------------------------------
000003 #include <env.h>
       78989A884489A48644444444444444444444444444444444444444444444444444444444
       B95334450C555B8E00000000000000000000000000000000000000000000000000000000
------------------------------------------------------------------------------
```

Figure 99. Tabs to spaces

2. On AIX

   Use the vi editor command "%s/<tab>/<blank>/g" to change the tab characters to spaces.

7.5.4 Square brackets

In our environment, when using the ISPF editor to look at the C code we are using code page 037. The left bracket ’[’ x'AD' in code page 037 displays a "Y" with an accent over it and right bracket ’]’ x'BD' in code page 037 displays as a double quote ("). USS and the C compiler are setup to use code page 1047. In code page 1047 x'AD' is a left square bracket and x'BD' is a right square bracket. These are the correct hexadecimal values for left and right square brackets when working with C and the C compiler on OS/390 even though they look funny when editing them.

If you experience the same problem you may want to change this situation so you can see a left bracket or right bracket in C file by changing your PCOMM workstation keyboard mapping and resetting your ISPF terminal type to 3278A. In this case follow the instructions below.

Your PCOMM workstation V4.1 for windows emulator keyboard mapping can be changed so that when you type a [it writes a x'AD' character and likewise for] a x’BD’ character will be written. From your PCOMM emulator icon bar:

1. Choose keyboard MAP
2. Select the key on the keyboard where you want the [to be located
3. In Change Current Actions for Selected Key, click the right arrow for the base option
4. In "Select a Key-Action", choose y with an accent over it
5. The y with an accent over it will appear in the base option

6. Repeat steps 2 - 5 for the right bracket, but choose second double quotes in the Character selection area

To change your ISPF terminal type, enter option 0 Settings from the ISPF Primary Option Menu. Refer to Figure 100.

```
Menu  Utilities  Compilers  Options  Status  Help
------------------------------------------------------------------------------
                           ISPF Primary Option Menu
Option ===> 0

0  Settings      Terminal and user parameters           User ID . : PAOLOR1
1  View          Display source data or listings        Time. . . : 16:22
2  Edit          Create or change source data           Terminal. : 3278A
3  Utilities     Perform utility functions              Screen. . : 1
4  Foreground    Interactive language processing        Language. : ENGLISH
5  Batch         Submit job for language processing     Appl ID . : PDF
6  Command       Enter TSO or Workstation commands      TSO logon : IKJACCT
7  Dialog Test   Perform dialog testing                 TSO prefix: PAOLOR1
8  LM Facility   Library administrator functions        System ID : SC63
9  IBM Products  IBM program development products       MVS acct. : ACCNT#
10 SCLM          SW Configuration Library Manager       Release . : ISPF 4.5
11 Workplace     ISPF Object/Action Workplace


Enter X to Terminate using log/list defaults


 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Figure 100. Step one - set terminal type

On the ISPF settings screen, in the Terminal Characteristics area, select option 4 for terminal type 3278A as reported in Figure 101.

```
 Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                                ISPF Settings                      End of data
Command ===>
                                                                   More:   -
  /  Tab to point-and-shoot fields       General
     Restore TEST/TRACE options            Input field pad . . N
     Session Manager mode                  Command delimiter . ;
  /  Jump from leader dots
     Edit PRINTDS Command
  /  Always show split line
     Enable EURO sign

Terminal Characteristics
  Screen format   1  1. Data    2. Std     3. Max     4. Part

  Terminal Type   4    1. 3277       2. 3277A      3. 3278       4. 3278A
                       5. 3290A      6. 3278T      7. 3278CF     8. 3277KN
                       9. 3278KN    10. 3278AR    11. 3278CY    12. 3278HN
                      13. 3278HO    14. 3278IS    15. 3278L2    16. BE163
                      17. BE190     18. 3278TH

 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Figure 101. Step two - set terminal type

Chapter 8. Testing, change control, and tuning

In this chapter we briefly mention our test techniques, and the need for change control and possible tuning actions.

8.1 Testing

The test implementation plan in 3.3.3.3, “Test plan” on page 38 mentions how the test plan is to be implemented, with what tools and resources.

In this section we look at the tasks related to the test plan, which normally includes the following five levels of testing:

- Function
- Unit
- System
- User acceptance
- Performance source/target

In our test plan for a pilot subset of the application we basically concluded the project at a level equivalent to unit test.

For each level you need to define the contents:

- Input data
- Sequence of key steps
- Defined completion point
- Expected result (program function no change, data access code, error logic, data accuracy, error free compile, error free execution, acceptance of input, output format)
- Test script/driver (how test results, contents of results)
- Test data
- Output files
- Log of test
- Clean data
- How to verify input, output, function
- Functional, ensure that the overall process runs to normal completion with a small sample of data

Program redesign addressed the areas of the project where it was necessary to prototype and test sections of module code that had to be redesigned for use with DB2. See 7.3, “Program redesign” on page 150. In addition, we had to conduct unit, function and system test for these changes. See Figure 9 on page 29 for a diagram of our testing cycle.

8.1.1 Function tests

This section describes the function tests which we performed after the conversion completion.

The function tests have been executed by means of:

- System tools and database query tools (MVS ISPF editor, AIX *vi* editor, DB2 SPUFI, Oracle *SQLPlus*, DataJoiner), as far as the database definitions and the data are concerned
- The execution on specific test data of the final application, what concerns the final application itself

8.1.1.1 Database definition test

The database definitions (tables, relationships, other database objects) can be tested using standard DB2 and Oracle query tools.

As an example we show in Figure 102 the table definitions on Oracle (using *SQLPlus*) and in Figure 103 the final table definitions on DB2 (using DataJoiner), as retrieved from the catalog:

```
SQL> desc reading
 Name                           Null?    Type
 ------------------------------ -------- ----
 TAG_NAME                       NOT NULL VARCHAR2(20)
 TIMESTAMP                      NOT NULL DATE
 VALUE                                   NUMBER
 VALIDITY_FLAG                           NUMBER(3)
```

Figure 102. Oracle table definition test

The sqptype fields reported in Figure 103 are contained in SQLDA and are explained in the *DB2 UDB for OS/390 SQL Reference*, SC26-9014.

```
 $ db2 "describe select * from reading"

SQLDA Information

 sqldaid : SQLDA     sqldabc: 896  sqln: 20  sqld: 4
 sqltype  sqllen  sqlname.length   sqlname.data
 448          20               8   TAG_NAME
 392          26               9   TIMESTAMP
 481           8               5   VALUE
 497           4              13   VALIDITY_FLAG
```

Figure 103. DB2 table definition test (using DataJoiner)

We can access the catalog of the two RDBMS to retrieve the referential integrity definitions. Figure 104 and Figure 105 provide you with a comparison between the foreign key definitions on Oracle (using *SQLPlus*) and DB2 (using SPUFI). The related DDL is described in 5.2.1.8, “Foreign keys conversion” on page 84.

```
SQL> select constraint_name, r_constraint_name, delete_rule from
user_constraints where table_name='SAMPLE' and constraint_type='R';

CONSTRAINT_NAME                R_CONSTRAINT_NAME              DELETE_RULE
------------------------------ ------------------------------ ---------
SAMPLEF1                       MATERIALP1                     NO ACTION
SAMPLEF2                       SAMPLE_POINTP1                 NO ACTION
```

Figure 104. Oracle referential integrity definition test

The two semantically different delete rules reported in the example, NO ACTION for Oracle and R (RESTRICT) for DB2, have the same meaning in our case (do not delete rows from the parent table that have dependents in the dependent table of the referential constraint) because no self-referencing constraints are present. See *DB2 UDB for OS/390 SQL Reference*, SC26-9014 under DELETE rules for details.

```
 SELECT RELNAME, REFTBNAME, DELETERULE FROM SYSIBM.SYSRELS            00260199
   WHERE TBNAME='SAMPLE';                                              00260299
---------+---------+---------+---------+---------+---------+---------+---------+
RELNAME   REFTBNAME           DELETERULE
---------+---------+---------+---------+---------+---------+---------+---------+
MTRL$NAM  MATERIAL            R
SAMPLE$P  SAMPLE_POINT        R
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Figure 105. DB2 referential integrity definition test

8.1.1.2 Data migration test

Our test on data migration consisted of four steps:

1. Check of data files downloaded from Oracle (see 6.2, “Unloading data from Oracle” on page 109)
2. Check of data files on OS/390 side, after transferring to the OS/390 system
3. Check of data files after processing VARCHAR fields with REXX script (see 6.6, “Reformatting data for DB2” on page 119)
4. Check and comparison of table contents on both Oracle and DB2 tables

In case of usage of DataJoiner, only point 4 needs to be performed.

For example, in Figure 106 we show the content of a data file on AIX machine, after unloading from the table:

```
cipros@BALTIC /home/cipros/DATA/DOWNLOAD > cat SAMPLE.DAT
1       T_0001_SP           1999-06-03-08.00.00.000000ARABIAN_HEAVY  Tank 0001 sample
point
2       T_0002_SP           1999-06-03-08.00.00.000000ARABIAN_HEAVY  Tank 0002 sample
point
3       T_0003_SP           1999-06-03-08.00.00.000000ARABIAN_HEAVY  Tank 0003 sample
point
4       T_0004_SP           1999-06-03-08.00.00.000000ARABIAN_HEAVY  Tank 0004 sample
point
```

Figure 106. Content of a data file on AIX environment

In Figure 107 we show the content of a data file on AIX machine, after unloading from the table:

```
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 1       T_0001_SP           1999-06-03-08.00.00.000000ARABIAN_HEAVY  Tan
000002 2       T_0002_SP           1999-06-03-08.00.00.000000ARABIAN_HEAVY  Tan
000003 3       T_0003_SP           1999-06-03-08.00.00.000000ARABIAN_HEAVY  Tan
000004 4       T_0004_SP           1999-06-03-08.00.00.000000ARABIAN_HEAVY  Tan
****** **************************** Bottom of Data ****************************
```

Figure 107. Content of a data file on MVS environment

After processing the file with the REXX script for the insertion of two-bytes binary length for VARCHAR fields, the file contains the following lines, as we show in Figure 108 in its hexadecimal representation (only the left part of the file is shown):

```
****** ***************************** Top of Data ******************************


------------------------------------------------------------------------------
000001 1         T_0001_SP           1999-06-03-08.00.00.000000  ARABIAN_HEAVY
       F444444400E6FFFF6ED44444444444FFFF6FF6FF6FF4FF4FF4FFFFFF00CDCCCCD6CCCEE4
       10000000093D0001D27000000000001999006003008B00B00B0000000D1912915D851580
------------------------------------------------------------------------------
000002 2         T_0002_SP           1999-06-03-08.00.00.000000  ARABIAN_HEAVY
       F444444400E6FFFF6ED44444444444FFFF6FF6FF6FF4FF4FF4FFFFFF00CDCCCCD6CCCEE4
       20000000093D0002D27000000000001999006003008B00B00B0000000D1912915D851580
------------------------------------------------------------------------------
000003 3         T_0003_SP           1999-06-03-08.00.00.000000  ARABIAN_HEAVY
       F444444400E6FFFF6ED44444444444FFFF6FF6FF6FF4FF4FF4FFFFFF00CDCCCCD6CCCEE4
       30000000093D0003D27000000000001999006003008B00B00B0000000D1912915D851580
------------------------------------------------------------------------------
000004 4         T_0004_SP           1999-06-03-08.00.00.000000  ARABIAN_HEAVY
       F444444400E6FFFF6ED44444444444FFFF6FF6FF6FF4FF4FF4FFFFFF00CDCCCCD6CCCEE4
       40000000093D0004D27000000000001999006003008B00B00B0000000D1912915D851580
------------------------------------------------------------------------------
****** ************************** Bottom of Data ***************************
```

Figure 108. Data file after processing with REXX script

You can see that the two bytes before the second and fourth fields contain the two bytes hexadecimal representation of the length of the related data (0009 and 000D, respectively).

At the end, after loading the data into the table, the comparison between the table content is shown in Figure 109, Figure 110 (only the left part of the table is shown), Figure 111 on page 187 and Figure 112 on page 187:

```
SQL> select * from sample;

SAMPLE_NUM SAMPLE_PT_NAME       TIMESTAMP MTRL_NAME       DESCRIPTION
---------- -------------------- --------- --------------- ----------------------
------------------
         1 T_0001_SP            03-JUN-99 ARABIAN_HEAVY   Tank 0001 sample point
         2 T_0002_SP            03-JUN-99 ARABIAN_HEAVY   Tank 0002 sample point
         3 T_0003_SP            03-JUN-99 ARABIAN_HEAVY   Tank 0003 sample point
```

Figure 109. Oracle table content retrieved using SQLPlus

```
 SELECT * FROM CIPROS.SAMPLE;                                          00260099
---------+---------+---------+---------+---------+---------+---------+---------+
 SAMPLE_NUM  SAMPLE_PT_NAME        TIMESTAMP                   MTRL_NAME
---------+---------+---------+---------+---------+---------+---------+---------+
          1  T_0001_SP             1999-06-03-08.00.00.000000  ARABIAN_HEAVY
          2  T_0002_SP             1999-06-03-08.00.00.000000  ARABIAN_HEAVY
          3  T_0003_SP             1999-06-03-08.00.00.000000  ARABIAN_HEAVY
          4  T_0004_SP             1999-06-03-08.00.00.000000  ARABIAN_HEAVY
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Figure 110. DB2 table content retrieved using SPUFI

```
$ db2 "select * from osample"

SAMPLE_NUM  SAMPLE_PT_NAME       TIMESTAMP                  MTRL_NAME       DESCRIPTION
----------- -------------------- -------------------------- ---------------
----------------------------------------
          1 T_0001_SP            1999-06-03-08.00.00.000000 ARABIAN_HEAVY   Tank 0001
sample point
          2 T_0002_SP            1999-06-03-08.00.00.000000 ARABIAN_HEAVY   Tank 0002
sample point
          3 T_0003_SP            1999-06-03-08.00.00.000000 ARABIAN_HEAVY   Tank 0003
sample point
          4 T_0004_SP            1999-06-03-08.00.00.000000 ARABIAN_HEAVY   Tank 0004
sample point

  4 record(s) selected.
```

Figure 111. Oracle table content retrieved using SQLPlus

```
$ db2 "select * from sample"

SAMPLE_NUM  SAMPLE_PT_NAME       TIMESTAMP                  MTRL_NAME       DESCRIPTION
----------- -------------------- -------------------------- ---------------
----------------------------------------
          1 T_0001_SP            1999-06-03-08.00.00.000000 ARABIAN_HEAVY   Tank 0001
sample point
          2 T_0002_SP            1999-06-03-08.00.00.000000 ARABIAN_HEAVY   Tank 0002
sample point
          3 T_0003_SP            1999-06-03-08.00.00.000000 ARABIAN_HEAVY   Tank 0003
sample point
          4 T_0004_SP            1999-06-03-08.00.00.000000 ARABIAN_HEAVY   Tank 0004
sample point

  4 record(s) selected.
```

Figure 112. DB2 table content retrieved using DataJoiner

8.1.1.3 Application test

For the application test at a function level, we used a set of input files for the application, containing specific data which allow us to test each functionality of the application itself.

For example, since the Process bridge, while inserting data into the READING table, updates the duplicate records found in the input file, we tried to run the program twice with the same file, by changing something in the file before the second run. The expected result is that the first time the input data are inserted into the table, the second time the table is updated with the new input data.

To show this, first we select the contents of the table, as shown in Figure 113 (using SPUFI):

```
 SELECT * FROM CIPROS.READING;                                        00260099
---------+---------+---------+---------+---------+---------+---------+---------+
TAG_NAME             TIMESTAMP                                      VALUE  VALID
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Figure 113. Contents of READING table before the first execution of the program

Then we executed the program with the following input file:

```
****** **************************** Top of Data ********************
000001 FM0010.PVHVG,0,1999-06-03-08.00.00.000000,0
000002 FM0011.PVHVG,0,1999-06-03-08.00.00.000000,0
000003 FM0012.PVHVG,0,1999-06-03-08.00.00.000000,0
000004 FM0013.PVHVG,0,1999-06-03-08.00.00.000000,0
000005 FM0014.PVHVG,0,1999-06-03-08.00.00.000000,0
000006 FM0015.PVHVG,0,1999-06-03-08.00.00.000000,0
000007 FM0016.PVHVG,0,1999-06-03-08.00.00.000000,0
000008 FM0017.PVHVG,0,1999-06-03-08.00.00.000000,0
000010 PM0018.PVHVG,0,1999-06-03-08.00.00.000000,0
****** **************************** Bottom of Data ******************
```

The content of the table after the execution of the program is shown in Figure 114.

```
 SELECT * FROM CIPROS.READING;                                        00260099
---------+---------+---------+---------+---------+---------+---------+---------+
TAG_NAME            TIMESTAMP                                      VALUE  VALID
---------+---------+---------+---------+---------+---------+---------+---------+
FM0010.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0011.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0012.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0013.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0014.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0015.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0016.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0017.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
PM0018.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
DSNE610I NUMBER OF ROWS DISPLAYED IS 9
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Figure 114. Contents of READING table after the first execution of the program

After changing the VALUE field of the first record of the file into 55555, we re-submitted the program, and the result of a select after the end of the execution is shown in Figure 115.

```
 SELECT * FROM CIPROS.READING;                                        00260099
---------+---------+---------+---------+---------+---------+---------+---------+
TAG_NAME            TIMESTAMP                                      VALUE  VALID
---------+---------+---------+---------+---------+---------+---------+---------+
FM0010.PVHVG        1999-06-03-08.00.00.000000  +0.5555500000000000E+05
FM0011.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0012.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0013.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0014.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0015.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0016.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0017.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
FM0018.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
PM0018.PVHVG        1999-06-03-08.00.00.000000  +0.0                E+00
DSNE610I NUMBER OF ROWS DISPLAYED IS 10
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Figure 115. Contents of READING table after the second execution of the program

The test case described above should be repeated for all the basic functionalities of the application.

8.1.2 Unit test

This section describes the unit tests performed after the conversion completion.

8.1.2.1 Database definition test

We did not perform unit tests for the database definition. We considered that a function definition for all the database objects definition could be considered as a unit test.

8.1.2.2 Data migration test

From a data migration point of view, we performed some queries to check the validity of the loaded data, and made comparisons across the tables in order to check the consistency of each area of the data model.

An example is shown in Figure 116.

```
  SELECT ANLY_VALUE, SA.TIMESTAMP FROM SAMPLE_ANALYSIS SA, SAMPLE S,   00260099
    STANDARD_RESULT SR WHERE S.SAMPLE_PT_NAME=SA.SAMPLE_PT_NAME AND      00260199
    SR.STD_RESLT_NAME=SA.STD_RESLT_NAME AND SA.STD_RESLT_NAME            00260299
    ='D15' AND ANLY_VALUE>'0.5';                                         00260399
---------+---------+---------+---------+---------+---------+---------+---------+
ANLY_VALUE            TIMESTAMP
---------+---------+---------+---------+---------+---------+---------+---------+
0.801                 1999-06-03-08.00.00.000000
0.928                 1999-06-03-08.00.00.000000
0.827                 1999-06-03-08.00.00.000000
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Figure 116. Example of a query statement to check data consistency

8.1.3 System and user acceptance test

In our simplified environment the unit test was considered satisfactory also for system and user acceptance test.

8.2 Performance tuning

The performance plan in 3.3.3.4, “Performance plan” on page 39 describes how performance tuning takes place, with what tools and resources.

The tasks for a performance plan are:

- Database configuration
- System configuration
- Tools and utilities

8.3 Change control

The change control plan in 3.3.3.5, “Change control plan” on page 39 describes how the change control procedure takes place, with what tools and resources.

The tasks for a change control procedure are:

- Document Change Request
- Review change
- Review effect of change
- Approval of change

8.3.1 Change control overview

Recognizing that some changes can occur during a project, a formal change control procedure is required to accommodate them. Such a procedure allows users to make cost and benefit trade-offs based on analysis or requested changes. It also ensures that only approved changes are implemented. Changes are handled through the formal Project Change Control Procedure.

8.3.2 Change control procedure

The tasks defined and approved at each phase of the project served as a baseline for the next phase. Changes to this baseline are requested through the Project Change Control Procedure.

The following provides a detailed process to follow if a change to the project plan is required:

- A change request will be the vehicle for communication change. The change request must describe the change, the rationale for the change and the effect the change will have on the project.
- The designated team leader of the requesting party will review the proposed change and determine whether to submit the request to the other party.
- Both team leaders will review the proposed change and approve it for further investigation or reject it. If the investigation is authorized, the team leaders and project manager will approve the investigation. The investigation will determine the effect that the implementation of the change request will have on the project and the schedule.
- A written change request must be signed by both team leaders and the project manager to authorize implementation of the investigated changes.

Appendix A. Sample script functions

This appendix contains the sample script functions that have been written during the project for the Oracle AIX environment. They are referred to in the text of the book and contain in the comments section of each listing a short description of their functionalities.

A.1 ddltabs.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script ddltabs.sh
#
# Starting from an Oracle export file containing the table definitions,
# extracts the CREATE TABLE DDLs and converts them into DB2/MVS syntax
#
# Syntax: ddltabs.sh export_file
#
# The script writes the DDLs on standard output. If you want to save
# the DDLs, redirect the stdout to a file. Example:
#
# ddltabs.sh export_file > output_ddl_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Some environment variables have to be defined before launching the script:
#
# Variable containing the DB2 database name
export DBN=CIPROS

# List of variables containing old and new names for the tablespaces
# The first variable is the number of the changed tablespaces
export NTS=6
export OLDTSN[1]=CPRS_BASE
export NEWTSN[1]=CPRSBASE
export OLDTSN[2]=CPRS_READING
export NEWTSN[2]=CPRSREAD
export OLDTSN[3]=CPRS_READING_IND
export NEWTSN[3]=CPRSRIDX
export OLDTSN[4]=CPRS_LOGE
export NEWTSN[4]=CPRSLOGE
export OLDTSN[5]=CPRS_LAB
export NEWTSN[5]=CPRSLAB
export OLDTSN[6]=CPRS_PRDSCN
export NEWTSN[6]=CPRSPRDS

# List of variables containing old and new names for the tables
# The first variable is the number of the changed tables
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT

# List of variables containing old and new names
# (in DB2 syntax) for some data types
# The first variable is the number of the changed data types
export NDT=2
export OLDDT[1]="VARCHAR(1)"
export NEWDT[1]="CHAR"
export OLDDT[2]="VARCHAR(3)"
export NEWDT[2]="CHAR(3)"

# Function definition: Cut the " (double quote) symbols
#
```

```
function dq_cut
{
        tr -d '"'
}

# Function definition: Cut the "(",  ")" and " " (blank) symbols
#
function par_cut
{
        tr -d '(' | tr -d ')' | tr -d ' '
}

# Function definition: Modify the Oracle storage clause
#
function sto_cl
{
        sed  -e "s/\(^.*\) PCTFREE\(.*\)TABLESPACE \(.*\)$/\1IN $DBN\.\3/g"
}

# Function definition: Convert Number Datatypes
#
function dt_num
{
          sed   -e 's/ DECIMAL/ NUMBER/g' \
                -e 's/ NUMBER(\*)/ NUMBER/g' \
                -e 's/ NUMBER(\*[ ]*,[0 ]*)/ NUMBER/g' \
                -e 's/ NUMBER(\([0-9 ]*\)\,\([0 ]*\))/ INTEGER/g' \
                -e 's/ NUMBER(\([0-9 ]*\)\,\([0-9 ]*\))/ DECIMAL(\1\,\2)/g' \
                -e 's/ NUMBER(\([0-9 ]*\))/ INTEGER/g' \
                -e 's/ NUMBER/ FLOAT/g'
}

# Function definition: Convert Character Datatypes
#
function dt_char
{
        sed   -e 's/ VARCHAR2/ VARCHAR/g' \
              -e 's/ LONG VARCHAR/ LONG/g' \
              -e 's/ LONG/ CLOB(2G)/g' \
              -e 's/ CHAR(\(.*\))/ VARCHAR(\1)/g'
}

# Function definition: Convert Binary Datatypes
#
function dt_bin
{
        sed     -e 's/ LONG RAW/ BLOB(2G)/g' \
                -e 's/ RAW(\([0-9]*\))/ CHAR(\1) FOR BIT DATA/g'
}

# Function definition: Convert Date Datatypes
#
function dt_date
{
        sed   -e 's/ DATE,/ TIMESTAMP,/g' \
              -e 's/ DATE)/ TIMESTAMP)/g'
}

# Function definition: Modify the Oracle primary key clause
# for the primary key definitions (with CONSTRAINT clause)
#
function pk_cl
{
        sed "s/ALTER TABLE\(.*\)ADD[ ]*CONSTRAINT\(.*\) \
PRIMARY KEY\(.*\) USING.*/\1,\3/"
}

# Function definition: Modify the Oracle primary key clause
# for the primary key definitions (without CONSTRAINT clause)
#
function pk_cl_nc
{
        sed "s/ALTER TABLE\(.*\)ADD[ ]*PRIMARY[ ]*KEY\(.*\) \
USING.*/\1,\2/"
}

# Function definition: Add ';' to end of each create table
#
```

```
function stmt_end
{
        sed   -e 's/$/;/g'
}

# Function definition: Add a newline to end of each field declaration
#
function new_line
{
       sed 's/),/),\
            /g
            s/FLOAT,/FLOAT,\
            /g
            s/INTEGER,/INTEGER,\
            /g
            s/SMALLINT,/SMALLINT,\
            /g
            s/DECIMAL,/DECIMAL,\
            /g
            s/CHAR,/CHAR,\
            /g
            s/DATA,/DATA,\
            /g
            s/ROWID,/ROWID,\
            /g
            s/TIMESTAMP,/TIMESTAMP,\
            /g
            s/NOT NULL,/NOT NULL,\
            /g
            s/IN \(.*\)\.\(.*\);/\
             IN \1\.\2;/'
}


# Start of main program
#
# Select the CREATE TABLE statements in Oracle export file
# and apply the functions

grep -w "CREATE TABLE " $1 | dq_cut | sto_cl | \
dt_num | dt_bin | dt_char | dt_date | stmt_end  > $1.tmp.0

# Modify the Oracle tablespace names

i=0
while [[ $i -lt $NTS ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDTSN[$i]}
NEW=${NEWTSN[$i]}
        sed "s/$DBN.$OLD/$DBN.$NEW/" $1.tmp.$im1 > $1.tmp.$i
done

mv $1.tmp.$i $1.tmp.0

# Modify the data types declared at the beginning of the script

i=0
while [[ $i -lt $NDT ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDDT[$i]}
NEW=${NEWDT[$i]}
        sed "s/$OLD/$NEW/g" $1.tmp.$im1 > $1.tmp.$i
done

export CRFN=$1.tmp.$i

# Select all the primary keys and write into a temporary file

grep -w "ALTER TABLE" $1 | grep -w "PRIMARY KEY"  | dq_cut | \
pk_cl | pk_cl_nc | par_cut > $1.pk

awk -f pk.awk $1.pk > $1.tmp.0

# Modify the Oracle table names as declared at the beginning of the script
```

```
i=0
while [[ $i -lt $NT ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDTN[$i]}
NEW=${NEWTN[$i]}
        sed "s/TABLE $OLD/TABLE $NEW/" $1.tmp.$im1 > $1.tmp.$i
done

# Add the newline characters and display the result to standard output

cat $1.tmp.$i | new_line

# Remove the temporary files

rm -f $1.tmp*
```

A.1.1 sednn.sh script

```
#!/bin/ksh
sed -e "s/$1[ ]*VARCHAR(\([0-9]*\))/$1 VARCHAR(\1) NOT NULL/" \
    -e "s/$1[ ]*CHAR(\([0-9]*\))/$1 CHAR(\1) NOT NULL/" \
    -e "s/$1[ ]*CHAR,/$1 CHAR NOT NULL,/" \
    -e "s/$1[ ]*CHAR)/$1 CHAR NOT NULL)/" \
    -e "s/$1[ ]*CLOB(\([0-9]*\))/$1 CLOB(\1) NOT NULL/" \
    -e "s/$1[ ]*BLOB(\([0-9]*\))/$1 BLOB(\1) NOT NULL/" \
    -e "s/$1[ ]*DBCLOB(\([0-9]*\))/$1 DBCLOB(\1) NOT NULL/" \
    -e "s/$1[ ]*FLOAT(\([0-9]*\))/$1 FLOAT(\1) NOT NULL/" \
    -e "s/$1[ ]*FLOAT,/$1 FLOAT NOT NULL,/" \
    -e "s/$1[ ]*FLOAT)/$1 FLOAT NOT NULL)/" \
    -e "s/$1[ ]*REAL/$1 REAL NOT NULL/" \
    -e "s/$1[ ]*DOUBLE/$1 DOUBLE NOT NULL/" \
    -e "s/$1[ ]*INTEGER/$1 INTEGER NOT NULL/" \
    -e "s/$1[ ]*SMALLINT/$1 SMALLINT NOT NULL/" \
    -e "s/$1[ ]*TIMESTAMP,/$1 TIMESTAMP NOT NULL,/" \
    -e "s/$1[ ]*TIMESTAMP)/$1 TIMESTAMP NOT NULL)/" \
    -e "s/$1[ ]*DECIMAL(\([0-9, ]*\))/$1 DECIMAL(\1) NOT NULL/" \
    -e "s/NOT[ ]*NULL[ ]*NOT[ ]*NULL/NOT NULL/g"
```

A.1.2 pk.awk script

```
BEGIN { FS="," }
{
system("cp $CRFN $CRFN.sed."$2)
for (i=2;i<=NF;i++)
{
j=i+1
system("grep -w \"CREATE TABLE "$1" \" $CRFN.sed."$i" | sednn.sh "$i" >$CRFN.sed."$j)
}
system("mv $CRFN.sed."$j" $CRFN.sed")
system("cat $CRFN.sed")
system("rm -f $CRFN.sed*")
}
```

A.2 ddlind.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script ddlind.sh
#
# Starting from an Oracle export file,
# extracts the CREATE INDEX DDLs and ALTER TABLE DDLs and converts them
# into DB2/MVS syntax
#
# Syntax: ddlind.sh export_file
#
# The script writes the DDLs on standard output. If you want to save
# the DDLs, redirect the stdout to a file. Example:
#
```

```
# ddlind.sh export_file > output_ddl_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Some environment variables have to be defined before launching the script:
#
# Variables on DB2 definitions
export DBN=CIPROS
export STOGROUP=CIPROS01
export PRIQTY=4000
export SECQTY=400
export ERASE=NO

# List of variables containing old and new names for the tables
# The first variable is the number of the changed tables
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT

# List of variables containing old and new names for the indexes
# The first variable is the number of the changed indexes
export NI=5
export OLDIN[1]=CASE_FACILITY_ASGP1
export NEWIN[1]=CASE_FACIL_ASGP1
export OLDIN[2]=ENG_UNIT_STANDARDP1
export NEWIN[2]=ENG_UNIT_STNDRDP1
export OLDIN[3]=INSTRUMENT_SERVICEP1
export NEWIN[3]=INSTR_SERVICEP1
export OLDIN[4]=STREAM_METH_RESULTP1
export NEWIN[4]=STREAM_METH_RESP1
export OLDIN[5]=CASE_FACILITY_ASGI2
export NEWIN[5]=CASE_FACIL_ASGI2

# Function definition: Cut the " (double quote) symbols
#
function dq_cut
{
        tr -d '"'
}

# Function definition: Modify the Oracle storage clause
#
function sto_cl
{
        sed -e "s/\(^.*\) PCTFREE.*/\1 USING STOGROUP $1 PRIQTY $2 SECQTY $3 ERASE $4/"

}

# Function definition: Modify the Oracle unique clause
#
function uni_cl
{
        sed "s/ALTER TABLE\(.*\)ADD[ ]*CONSTRAINT\(.*\) \
UNIQUE\(.*\)USING.*/CREATE UNIQUE INDEX \2 ON \1 \3 USING \
STOGROUP $1 PRIQTY $2 SECQTY $3 ERASE $4/"
}

# Function definition: Modify the Oracle primary key clause
# for the primary indexes (with CONSTRAINT clause)
#
function pki_cl
{
         sed "s/ALTER TABLE\(.*\)ADD[ ]*CONSTRAINT\(.*\)\
 PRIMARY KEY\(.*\)USING.*/CREATE UNIQUE INDEX \2 ON \1 \3 \
USING STOGROUP $1 PRIQTY $2 SECQTY $3 ERASE $4/"
}

# Function definition: Modify the Oracle primary key clause
# for the primary key definitions (with CONSTRAINT clause)
```

```
#
function pk_cl
{
        sed "s/ALTER TABLE\(.*\)ADD[ ]*CONSTRAINT\(.*\) \
PRIMARY KEY\(.*\) USING.*/ALTER TABLE \1 ADD PRIMARY KEY \3/"
}

# Function definition: Modify the Oracle primary key clause
# for the primary key definitions (without CONSTRAINT clause)
#
function pk_cl_nc
{
        sed "s/ALTER TABLE\(.*\)ADD[ ]*PRIMARY[ ]*KEY\(.*\) \
USING.*/ALTER TABLE \1 ADD PRIMARY KEY \2/"
}

# Function definition: Add ';' to end of each create table
#
function stmt_end
{
        sed   -e 's/$/;/g'
}

# Function definition: Add a newline to end of each field declaration
#
function new_line
{
       sed 's/ ON /\
             ON /g
             s/(/\
             (/g
             s/USING/\
             USING/g
             s/PRIQTY/\
                   PRIQTY/g
             s/SECQTY/\
                   SECQTY/g
             s/ERASE/\
                   ERASE/g'
}


# Start of main program
#
# Select the ALTER TABLE ADD CONSTRAINT PRIMARY KEY statements in the export
# file and apply the functions for the creation of the primary unique indexes

grep -w "ALTER TABLE" $1 | grep "CONSTRAINT" | grep "PRIMARY KEY" | \
dq_cut | pki_cl $STOGROUP $PRIQTY $SECQTY $ERASE | stmt_end  >> $1.tmp.0

# Select the ALTER TABLE ADD CONSTRAINT PRIMARY KEY statements in the export
# file and apply the functions for the creation of the primary keys

grep -w "ALTER TABLE" $1 | grep "CONSTRAINT" | grep "PRIMARY KEY" | \
dq_cut | pk_cl $STOGROUP $PRIQTY $SECQTY $ERASE | stmt_end  >> $1.tmp.0

# Select the ALTER TABLE ADD PRIMARY KEY statements in the export
# file and apply the functions for the creation of the primary keys

grep -w "ALTER TABLE" $1 | grep "PRIMARY KEY" | grep -v "CONSTRAINT" | \
dq_cut | pk_cl_nc $STOGROUP $PRIQTY $SECQTY $ERASE | stmt_end  >> $1.tmp.0

# Select the CREATE INDEX statements in Oracle export file
# and apply the functions

grep -w "CREATE[A-Z ]*INDEX " $1 | dq_cut | \
sto_cl $STOGROUP $PRIQTY $SECQTY $ERASE | stmt_end  >> $1.tmp.0

# Select the ALTER TABLE ... UNIQUE statements in Oracle export file
# and apply the functions

grep -w "ALTER TABLE" $1 | grep "UNIQUE" | dq_cut | \
uni_cl $STOGROUP $PRIQTY $SECQTY $ERASE | stmt_end  >> $1.tmp.0


# Modify the Oracle table names as declared at the beginning of the script

i=0
```

```
while [[ $i -lt $NT ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDTN[$i]}
NEW=${NEWTN[$i]}
        sed "s/$OLD/$NEW/g" $1.tmp.$im1 > $1.tmp.$i
done

mv $1.tmp.$i $1.tmp.0

# Modify the DB2 index names as declared at the beginning of the script

i=0
while [[ $i -lt $NI ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDIN[$i]}
NEW=${NEWIN[$i]}
        sed "s/$OLD/$NEW/g" $1.tmp.$im1 > $1.tmp.$i
done

# Add the newline characters and display the result to standard output

cat $1.tmp.$i | new_line

# Remove the temporary files

rm -f $1.tmp*
```

A.3 ddlfk.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script ddlfk.sh
#
# Starting from an Oracle export file,
# extracts the ALTER TABLE ... FOREIGN KEY DDLs
# and converts them into DB2/MVS syntax
#
# Syntax: ddlfk.sh export_file
#
# The script writes the DDLs on standard output. If you want to save
# the DDLs, redirect the stdout to a file. Example:
#
# ddlfk.sh export_file > output_ddl_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Some environment variables have to be defined before launching the script:
#
# List of variables containing old and new names for the tables
# The first variable is the number of the changed tables
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT

# Function definition: Cut the " (double quote) symbols
#
function dq_cut
{
        tr -d '"'
}

# Function definition: Modify the Oracle storage clause
```

```
#
function fk_cl
{
        sed  -e "s/ADD[ ]*CONSTRAINT\(.*\)FOREIGN KEY/FOREIGN KEY/"
}

# Function definition: Add ';' to end of each create table
#
function stmt_end
{
        sed   -e 's/$/;/g'
}

# Function definition: Add a newline to end of each field declaration
#
function new_line
{
       sed 's/FOREIGN KEY/\
            FOREIGN KEY/
            s/REFERENCES/\
            REFERENCES/
            s/ON DELETE/\
            ON DELETE/'
}


# Start of main program
#
# Select the ALTER TABLE ... FOREIGN KEY statements in Oracle export file
# and apply the functions

grep -w "ALTER TABLE" $1 | grep "FOREIGN KEY" | \
dq_cut | fk_cl | stmt_end  > $1.tmp.0


# Modify the Oracle table names as declared at the beginning of the script

i=0
while [[ $i -lt $NT ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDTN[$i]}
NEW=${NEWTN[$i]}
        sed "s/$OLD/$NEW/g" $1.tmp.$im1 > $1.tmp.$i
done

# Add the newline characters and display the result to standard output

cat $1.tmp.$i | new_line

# Remove the temporary files

rm -f $1.tmp*
```

A.4 ddlgrnt.sh

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script ddlgrnt.sh
#
# Starting from an Oracle export file containing the table definitions,
# extracts the GRANT statements and converts them into DB2/MVS syntax
#
# Syntax: ddlgrnt.sh export_file
#
# The script writes the statements on standard output. If you want to save
# the statements, redirect the stdout to a file. Example:
#
# ddlgrnt.sh export_file > output_ddl_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
```

```
# Some environment variables have to be defined before launching the script:
#
# List of variables containing old and new names for the tables
# The first variable is the number of the changed tables
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT

# List of variables containing old and new names
# of granted users and roles
# The first variable is the number of the changed definition
export NG=2
export OLDG[1]="ROLE_CIPROS"
export NEWG[1]="PAOLOR2"
export OLDG[2]="ROLE_VIEW"
export NEWG[2]="PAOLOR3"

# Function definition: Cut the " (double quote) symbols
#
function dq_cut
{
        tr -d '"'
}

# Function definition: Add ';' to end of each grant statement
#
function stmt_end
{
        sed   -e 's/$/;/'
}

# Start of main program
#
# Select the GRANT statements in Oracle export file
# and apply the previous functions

grep -w "^GRANT" $1 | dq_cut | stmt_end  > $1.tmp.0

# Modify the granted users as declared at the beginning of the script

i=0
while [[ $i -lt $NG ]]
do
        im1=$i
        i=`expr $i + 1`
        OLD=${OLDG[$i]}
        NEW=${NEWG[$i]}
        sed "s/$OLD/$NEW/g" $1.tmp.$im1 > $1.tmp.$i
done

mv $1.tmp.$i $1.tmp.0

# Modify the table names as declared at the beginning of the script

i=0
while [[ $i -lt $NT ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDTN[$i]}
NEW=${NEWTN[$i]}
        sed "s/ $OLD/ $NEW/" $1.tmp.$im1 > $1.tmp.$i
done

# Display the result to standard output

cat $1.tmp.$i
```

```
# Remove the temporary files

rm -f $1.tmp*
```

A.5 ddlchk.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script ddlchk.sh
#
# Starting from an Oracle export file,
# extract the ALTER TABLE DDLs with ADD CHECK and convert them
# into DB2/MVS syntax
#
# Syntax: ddlchk.sh export_file
#
# The script writes the DDLs on standard output. If you want to save
# the DDLs, redirect the stdout to a file. Example:
#
# ddlchk.sh export_file > output_ddl_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# List of variables containing old and new names for the tables
# The first variable is the number of the changed tables
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT

# Function definition: Cut the " (double quote) symbols
#
function dq_cut
{
        tr -d '"'
}

# Function definition: Add ';' to end of each create table
#
function stmt_end
{
        sed   -e 's/$/;/g'
}

# Start of main program
#
# Select the ALTER TABLE ... ADD CHECK statements in Oracle export file
# and apply the functions

grep -w "ALTER TABLE" $1 | grep -E "ADD.*CHECK" | \
dq_cut | stmt_end  >> $1.tmp.0


# Modify the Oracle table names as declared at the beginning of the script

i=0
while [[ $i -lt $NT ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDTN[$i]}
NEW=${NEWTN[$i]}
        sed "s/$OLD/$NEW/g" $1.tmp.$im1 > $1.tmp.$i
done

# Write the file to standard output

cat $1.tmp.$i
```

```
# Remove the temporary files

rm -f $1.tmp*
```

A.6 ddlalias.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script ddlalias.sh
#
# Starting from a file containing the list of the tables,
# extracts from Oracle catalog all the defined PUBLIC synonyms
# for the CIPROS user and creates the CREATE ALIAS DB2 statements
#
# Syntax: ddlalias.sh list_table_file
#
# The script writes the output on standard output. If you want to save
# the output, redirect the stdout to a file. Example:
#
# ddlalias.sh list_table_file > output_ddl_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# The following environment variables contain the CIPROS user and
# the SYS user with the related Oracle password:
#
SYS=SYS
SYS_PWD=SYS
OWNER=CIPROS
#
# The following environment variable contains the DB2 schema
#
SCHEMA=PAOLOR2
#
# List of variables containing old and new names for the tables
# The first variable is the number of the changed tables
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT
#
# These environment variables are for internal use:
#
SQLFILE=selalias.sql
OUTFILE=selalias.out
FILENAME=$1
#
# Function definition: Create the sed script file
# to append the last line of the sql file to the previous one
#
function create_last_line_sed
{
echo "/.*/{" > last_line.sed
echo "N" >> last_line.sed
echo "s/_\\\n);/);/" >> last_line.sed
echo "}" >> last_line.sed
}

# Start of main program
#
# Create the select statement for ALIASES and write into the SQLFILE file
# The "@" symbol (instead of "'") and the "_" symbol (instead of ",")
# are used to avoid problems with sed command

NTABS=`wc -l < $FILENAME`
DIV2=`expr $NTABS % 2`
echo "select 'create alias ' || " > $SQLFILE
```

```
echo "sname || ' for $SCHEMA.' || " >> $SQLFILE
echo "tname || ';' from synonyms " >> $SQLFILE
if [[ $DIV2 -eq 0 ]]
then
echo "where creator='$OWNER' and syntype='PUBLIC'" >> $SQLFILE
else
echo "where creator='$OWNER' " >> $SQLFILE
echo " and syntype='PUBLIC'" >> $SQLFILE
fi
echo "and tname in (" >> $SQLFILE
awk '{printf("%s%s%s_\n"),"@",$1,"@"}' $FILENAME >> $SQLFILE
echo ");" >> $SQLFILE

# Change the "@" characters in "'"
sed s/@/\'/g $SQLFILE > $SQLFILE.tmp
mv $SQLFILE.tmp $SQLFILE

# Create and apply the sed command file to append the
# last line containing ");" to the previous one, then remove the sed file
create_last_line_sed
sed -f last_line.sed $SQLFILE > $SQLFILE.tmp
mv $SQLFILE.tmp $SQLFILE
rm -f last_line.sed

# Change the "_" characters at the end of each line in ","
sed s/_$/,/g $SQLFILE > $SQLFILE.tmp
mv $SQLFILE.tmp $SQLFILE

# Run the sql scripts
sqlplus -s $SYS/$SYS_PWD <<EOF  >/dev/null 2>&1
clear columns
clear breaks
set pagesize 50000
set newpage 1
set feedback off
spool $OUTFILE
@$SQLFILE
spool off
EOF

rm -f $SQLFILE

# Cut head from the file

tail +4 $OUTFILE > $OUTFILE.tmp1
mv $OUTFILE.tmp1 $OUTFILE

# Modify the Oracle table names as declared at the beginning of the script

mv $OUTFILE $OUTFILE.tmp.0

i=0
while [[ $i -lt $NT ]]
do
        im1=$i
        i=`expr $i + 1`
        OLD=${OLDTN[$i]}
        NEW=${NEWTN[$i]}
        sed "s/$OLD/$NEW/g" $OUTFILE.tmp.$im1 > $OUTFILE.tmp.$i
done

# Write the SET CURRENT SQLID statement to standard output

echo "SET CURRENT SQLID=CIPROS;"

# Write the file to standard output

cat $OUTFILE.tmp.$i

# Remove the temporary files

rm -f $OUTFILE*
```

A.7 ddlsyn.sh script

```
#!/usr/bin/ksh
```

```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script ddlsyn.sh
#
# Starting from a file containing the list of the tables,
# extracts from Oracle catalog all the defined PRIVATE synonyms
# for the CIPROS user and creates the CREATE SYNONYNM DB2 statements
#
# Syntax: ddlsyn.sh list_table_file
#
# The script writes the output on standard output. If you want to save
# the output, redirect the stdout to a file. Example:
#
# ddlsyn.sh list_table_file > output_ddl_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# The following environment variables contain the CIPROS user and password
#
OWNER=CIPROS
OWNER_PWD=CIPROS
#
# The following environment variable contains the DB2 schema
#
SCHEMA=CIPROS
#
# List of variables containing old and new names for the tables
# The first variable is the number of the changed tables
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT
#
# These environment variables are for internal use:
#
SQLFILE=selsyn.sql
OUTFILE=selsyn.out
FILENAME=$1
#
# Function definition: Create the sed script file
# to append the last line of the sql file to the previous one
#
function create_last_line_sed
{
echo "/.*/{" > last_line.sed
echo "N" >> last_line.sed
echo "s/_\\\n);/);/" >> last_line.sed
echo "}" >> last_line.sed
}

# Start of main program
#
# Create the select statement for SYNONYMS and write into the SQLFILE file
# The "@" symbol (instead of "'") and the "_" symbol (instead of ",")
# are used to avoid problems with sed command

NTABS=`wc -l < $FILENAME`
DIV2=`expr $NTABS % 2`
echo "select 'create synonym ' || " > $SQLFILE
echo "synonym_name || ' for $SCHEMA.' || " >> $SQLFILE
echo "table_name || ';' from user_synonyms " >> $SQLFILE
if [[ $DIV2 -eq 0 ]]
then
echo "where table_owner='$OWNER' " >> $SQLFILE
echo "and table_name in (" >> $SQLFILE
else
        echo "where table_owner='$OWNER' and table_name in (" >> $SQLFILE
fi
awk '{printf("%s%s%s_\n"),"@",$1,"@"}' $FILENAME >> $SQLFILE
echo ");" >> $SQLFILE
```

```
# Change the "@" characters in "'"
sed s/@/\'/g $SQLFILE > $SQLFILE.tmp
mv $SQLFILE.tmp $SQLFILE

# Create and apply the sed command file to append the
# last line containing ");" to the previous one, then remove the sed file
create_last_line_sed
sed -f last_line.sed $SQLFILE > $SQLFILE.tmp
mv $SQLFILE.tmp $SQLFILE
rm -f last_line.sed

# Change the "_" characters at the end of each line in ","
sed s/_$/,/g $SQLFILE > $SQLFILE.tmp
mv $SQLFILE.tmp $SQLFILE

# Run the sql scripts
sqlplus -s $OWNER/$OWNER_PWD <<EOF  >/dev/null 2>&1
clear columns
clear breaks
set pagesize 50000
set newpage 1
set feedback off
spool $OUTFILE
@$SQLFILE
spool off
EOF

rm -f $SQLFILE

# Cut head from the file

tail +4 $OUTFILE > $OUTFILE.tmp1
mv $OUTFILE.tmp1 $OUTFILE

# Modify the Oracle table names as declared at the beginning of the script

mv $OUTFILE $OUTFILE.tmp.0

i=0
while [[ $i -lt $NT ]]
do
        im1=$i
        i=`expr $i + 1`
        OLD=${OLDTN[$i]}
        NEW=${NEWTN[$i]}
        sed "s/$OLD/$NEW/g" $OUTFILE.tmp.$im1 > $OUTFILE.tmp.$i
done

# Write the file to standard output

cat $OUTFILE.tmp.$i

# Remove the temporary files

rm -f $OUTFILE*
```

A.8 ddlview.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script ddlview.sh
#
# Starting from an Oracle full export file, extracts the CREATE VIEW statements
#
# Syntax: ddlview.sh export_file
#
# The script writes the statements on standard output. If you want to save
# the statements, redirect the stdout to a file. Example:
#
# ddlview.sh export_file > output_ddl_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Some environment variables have to be defined before launching the script:
#
# List of variables containing old and new names for the tables
# The first variable is the number of the changed tables
```

```
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT

# Function definition: Cut the " (double quote) symbols
#
function dq_cut
{
       tr -d '"'
}

# Function definition: get the CREATE VIEW statements from the input file
#
function views
{
  read line
  while [ $? -eq 0 ]
  do
       echo $line | grep "^CREATE VIEW" 2>&1 >/dev/null
       if [ $? -eq 0 ]
       then
         echo $line | tr -d '"'
         read line
         while [ "$line" != "" ]
         do
               echo $line | sed -e 's/^.*select/select/' \
                                -e 's/^.*SELECT/SELECT/'
               read line
               done
         echo ";"
               fi
               read line
       done
}

# Start of main program
#
# Cat the Oracle export file and apply the previous functions

cat $1 | dq_cut | views > $1.tmp.0

# Modify the table names as declared at the beginning of the script

i=0
while [[ $i -lt $NT ]]
do
im1=$i
i=`expr $i + 1`
OLD=${OLDTN[$i]}
NEW=${NEWTN[$i]}
        sed "s/ $OLD/ $NEW/" $1.tmp.$im1 > $1.tmp.$i
done

# Display the result to standard output

cat $1.tmp.$i

# Remove the temporary files

rm -f $1.tmp*
```

A.9 download.sh script

```
#!/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script download.sh
```

```
#
# Starting from an flat file containing a list of all the table,
# extracts data from Oracle for each table and writes data into
# a file named table_name.DAT, formatted in columns
#
# Syntax: download.sh table_list_file
#
# This script uses the awk command with the following awk command files:
#
# desc.awkformats the query command files using RPAD and DECODE
#to obtain a column-formatted output
#
# count.awkcomputes the total length of a record
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Define the environment variables for the oracle user and password
CPRS_USR=cipros
CPRS_PWD=cipros
#
# Start of main program
#
# Loop on all the tables listed in the input file

for i in `cat $1`
do

# Define some environment variables for temporary files

export OUTFILE=$i.DAT
DSCFILE=$i.dsc
SQLFILE=$i.sql
VARFILE=$i.var
ALLFILE=$i.all
POSFILE=$i.pos
rm -f $OUTFILE
rm -f $DSCFILE
rm -f $SQLFILE

# Extract the table description from Oracle catalog

sqlplus -s $CPRS_USR/$CPRS_PWD <<EOF  >/dev/null 2>&1
clear columns
clear breaks
set pagesize 100
set newpage 1
set feedback off
spool $DSCFILE
desc $i
EOF

# Cut head and tail from the file containing the descriptions of the tables
# Change also the NOT NULL clause in a blank string
# and cut the blanks in the first column

tail +3 $DSCFILE | sed 's/NOT NULL/        /; s/^ //' > $DSCFILE.tmp1
NL=`wc -l < $DSCFILE.tmp1`
NLM1=`expr $NL - 1`
head -$NLM1 $DSCFILE.tmp1 > $DSCFILE.tmp2
cp $DSCFILE.tmp2 $VARFILE

# Change the data types, leaving in the file the respective lengths
# It is assumed that 41 bytes are enough to contain the significative
# part of the NUMBER fields

sed -e 's/ VARCHAR2(/ /' \
    -e 's/ NUMBER(/ /' \
    -e 's/ NUMBER/ 41/' \
    -e 's/ INTEGER(/ /' \
    -e 's/ INTEGER/ 41/' \
    -e 's/ CHAR(/ /' \
    -e 's/ CHAR/ 1/' \
    -e 's/ RAW(/ /' \
    -e 's/ VARCHAR(/ /' \
    -e 's/)//' \
    -e 's/\([0-9]*\)\,\([0-9 ]\)*/\1/' \
 $DSCFILE.tmp2 > $DSCFILE.tmp3

mv $DSCFILE.tmp3 $DSCFILE
```

```
rm -f $DSCFILE.tmp*

# Compute the record length of the table
# using the count.awk awk script

LS=`awk -f count.awk $DSCFILE`

# Prepare the heading of the query statement on the table
# by echoing the statements into the sql file

echo "clear columns" > $SQLFILE
echo "clear breaks" >> $SQLFILE
echo "set pagesize 50000" >> $SQLFILE
echo "set linesize $LS" >> $SQLFILE
echo "set feedback off" >> $SQLFILE
echo "set heading off" >> $SQLFILE
echo "set space 0" >> $SQLFILE
echo "set newpage 1" >> $SQLFILE
echo "spool $OUTFILE" >> $SQLFILE
echo "select '' " >> $SQLFILE

# Append to the query statement file the list of the table fields
# to obtain the column layout, using the desc.awk awk script

awk -f desc.awk $DSCFILE >> $SQLFILE

# Append to the query statement file the "from" clause
# and the closing instructions

echo "from $i;" >> $SQLFILE
echo "spool off" >> $SQLFILE
echo "quit" >> $SQLFILE

# Execute the query statement

sqlplus -s $CPRS_USR/$CPRS_PWD @$SQLFILE >/dev/null  2>&1

# Cut the first line from the output file

tail +2 $OUTFILE > $OUTFILE.tmp
mv $OUTFILE.tmp $OUTFILE

# Change  the DATE data type into its DB2 external length, 26 bytes

sed 's/ DATE/ 26/' $DSCFILE > $DSCFILE.tmp1
mv $DSCFILE.tmp1 $DSCFILE

# End of the loop

done
```

A.9.1 count.awk script

```
BEGIN { total=0 }
{
if ($2 == "DATE")
total +=26
else
total += $2
}
END { print total }
```

A.9.2 desc.awk script

```
BEGIN {}
{
if ($2 == "DATE")
print " || rpad(DECODE("$1",NULL,' ',TO_CHAR("$1",'YYYY-MM-DD-HH24.MI.SS') || '.000000'),26)
"
else
print " || rpad(DECODE("$1",NULL,' ',"$1"),"$2") "
}
```

A.10 nick.sh script

```
#!/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script nick.sh
#
# Starting from an flat file containing a list of all the table,
# creates the "db2 create nickname" statements
# for the DataJoiner configuration
#
# Syntax: nick.sh table_list_file
#
# The script writes on standard output the list of sql commands
# If you want to save the list to a file, you can redirect the stdout
# using the following syntax:
#
# nick.sh table_list_file > sql_file
#
# and then launch the db2 CLP as following:
#
# db2 -tvf sql_file
#
# This script uses the awk command with the following awk command files:
#
# nick.awk              parses and creates the sql command list starting
#from the table list file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Some environment variables as to be defined before launching the script:
#
# List of variables containing the server and db names (db2 on workstation
# where DataJoiner is installed, Oracle, DB2/MVS)
# DB2 server and database name
export DB2SVR="mvs63"
export DB2DB="cipros"
# Oracle server and database name
export OSVR="oradb"
export ODB="cipros"
# DB2 workstation database name
export DJDB="djdb"

# List of variables containing old and new names for the DB2 tables
# The first variable is the number of the changed tables
export NT=6
export OLDTN[1]=CASE_FACILITY_ASSIGNMENT
export NEWTN[1]=CASE_FACILITY_ASG
export OLDTN[2]=ENG_UNIT_CONVERSION
export NEWTN[2]=ENG_UNIT_CONV
export OLDTN[3]=FAC_INSTR_ASSIGNMENT
export NEWTN[3]=FAC_INSTR_ASG
export OLDTN[4]=INSTRUMENT_MDL_TYPE
export NEWTN[4]=INSTRUMENT_MDL
export OLDTN[5]=STREAM_INSTR_ASSIGNMENT
export NEWTN[5]=STREAM_INSTR_ASG
export OLDTN[6]=STREAM_SAMPLE_PT_ASS
export NEWTN[6]=STREAM_SAMPLE_PT

# List of variables containing old and new names
# for the Oracle nicknames (for Oracle table names, whose length
# is greater then 18 characters after adding an "O" before the name
# The first variable is the number of the changed tables
export NOT=2
export OLDOT[1]=OINSTRUMENT_SERVICE
export NEWOT[1]=OINSTRUMENT_SERV
export OLDOT[2]=OSTREAM_METH_RESULT
export NEWOT[2]=OSTREAM_METH_RES

# Function definition: Ad ';' to end of each create nickname statement
#
function stmt_end
{
        sed   -e 's/$/;/g'
}

# Start of main program
# Copy the input file in a temporary file
```

```
cp $1 $1.tmp.0

# Processing the nickname creation

awk -f nick.awk $1.tmp.0 > $1.tmp

mv $1.tmp $1.tmp.0

#Modify nicknames longer then 18 chars
#
i=0
while [[ $i -lt $NT ]]
do
        im1=$i
        i=`expr $i + 1`
        OLD=${OLDTN[$i]}
        NEW=${NEWTN[$i]}
        sed "s/nickname $OLD/nickname $NEW/g" $1.tmp.$im1 | \
        sed "s/nickname O$OLD/nickname O$NEW/g" > $1.tmp.$i
done

mv $1.tmp.$i $1.tmp.0

#Modify source table names (DB2/MVS) longer then 18 chars
#
i=0
while [[ $i -lt $NT ]]
do
        im1=$i
        i=`expr $i + 1`
        OLD=${OLDTN[$i]}
        NEW=${NEWTN[$i]}
        sed "s/$DB2SVR\.$DB2DB\.$OLD/$DB2SVR\.$DB2DB\.$NEW/g" $1.tmp.$im1 > $1.tmp.$i
done

mv $1.tmp.$i $1.tmp.0

#Modify Oracle nick names
#
i=0
while [[ $i -lt $NOT ]]
do
        im1=$i
        i=`expr $i + 1`
        OLD=${OLDOT[$i]}
        NEW=${NEWOT[$i]}
        sed "s/nickname $OLD/nickname $NEW/g" $1.tmp.$im1 > $1.tmp.$i
done

# Add the connect statement (to the DataJoiner database)
# to the output command file
# and add the ";" at the end of each statement

echo "connect to $DJDB;"; cat $1.tmp.$i | stmt_end

# Remove the temporary files

rm -f $1.tmp.*
```

A.10.1 nick.awk script

```
BEGIN { }
{
system("echo create nickname " $1 " for $DB2SVR.$DB2DB." $1 )
system("echo create nickname O" $1 " for $OSVR.$ODB." $1 )
}
END { }
```

A.11 gendcl.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script gendcl.sh
#
# Starting from a file containing the declaration of the
```

```
# host variables generated by DCLGEN DB2 utility,
# creates the following C-language statements, for each table
# (from rtb input structure into DB fields):
#
### For VARCHAR fields: ###
# TAB_STRUCT.FIELD.FIELD_len=strlen(rtb->field);
# strncpy(TABSTRUCT.FIELD.FIELD_data,rtb->field,TABSTRUCT.FIELD.FIELD_len);
### For DOUBLE fields: ###
# TABSTRUCT.FIELD=atof(rtb->field);
### For LONG INT and SHORT INT fields: ###
# TABSTRUCT.FIELD=atoi(rtb->field);
### For CHAR fields: ###
# strcpy(TABSTRUCT.FIELD,rtb->field);
#
# Syntax: gendcl.sh dclgen_input_file
#
# The script writes the statements on standard output. If you want to
# save them, redirect the stdout to a file. Example:
#
# gendcl.sh dclgen_input_file > output_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Function definition: parses the input file and creates the statements
#
function dclgen
{
read line
while [ $? -eq 0 ]
do
  echo "$line" | grep "C DECLARATION FOR TABLE"
  if [ $? -eq 0 ]
  then
    echo ""
    tn1=`echo "$line" | awk '{print $6}' | sed 's/ //g'`
    tn="DCL"$tn1
    read line
    read line
    endtab=0
    while [ $endtab -eq 0 ]
    do
      read line
      datatype=`echo "$line" | \
      sed 's/{//; s/long int/longint/; s/short int/shortint/' | \
      awk '{print $1}' | sed 's/ //g'`
      if [[ "$datatype" = "struct" ]]
        then
        read line
        read line
        read line
        fld=`echo "$line" | sed "s/ //g; s/\;//; s/\}//"`
lowfld=`echo $fld | \
  sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
echo "$tn.$fld.$fld""_len=strlen(rtb->$lowfld);"
echo "strncpy($tn.$fld.$fld""_data,rtb->$lowfld,$tn.$fld.$fld""_len);"
      elif [[ $datatype = "double" ]]
      then
fld=`echo "$line" | sed "s/ //g; s/\;//; s/double//; s/^{//"`
lowfld=`echo $fld | \
          sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
echo "$tn.$fld=atof(rtb->$lowfld);"
      elif [[ $datatype = "longint" ]]
      then
fld=`echo "$line" | sed "s/ //g; s/\;//; s/longint//; s/^{//"`
lowfld=`echo $fld | \
          sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
echo "$tn.$fld=atoi(rtb->$lowfld);"
      elif [[ $datatype = "shortint" ]]
      then
fld=`echo "$line" | sed "s/ //g; s/\;//; s/shortint//; s/^{//"`
lowfld=`echo $fld | \
          sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
echo "$tn.$fld=atoi(rtb->$lowfld);"
      elif [[ $datatype = "char" ]]
      then
fld=`echo "$line" | sed "s/ //g; s/\[.*\]\;//" | \
                            sed "s/\;//; s/char//; s/^{//"`
lowfld=`echo $fld | \
          sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
```

```
echo "strcpy($tn.$fld,rtb->$lowfld);"
      else
        echo ""
        echo ""
 endtab=1
      fi
    done
  fi
  read line
done
}

# Start of main program
#
# Applies the parsing function and displays the result to standard output
#
cat $1 | dclgen
```

A.12 genmemset.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script genmemset.sh
#
# Starting from a file containing the declaration of the
# host variables generated by DCLGEN DB2 utility,
# creates memset C-language statements, for each table, for
#VARCHAR fields
#
# Syntax: genmemset.sh dclgen_input_file
#
# The script writes the statements on standard output. If you want to
# save them, redirect the stdout to a file. Example:
#
# genmemset.sh dclgen_input_file > output_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Function definition: parses the input file and creates the statements
#
function dclgen_memset
{
read line
while [ $? -eq 0 ]
do
  echo "$line" | grep "C DECLARATION FOR TABLE"
  if [ $? -eq 0 ]
  then
    tn1=`echo "$line" | awk '{print $6}' | sed 's/ //g'`
    tn="DCL"$tn1
    echo ""
    read line
    read line
    endtab=0
    while [ $endtab -eq 0 ]
    do
      read line
      datatype=`echo "$line" | \
      sed 's/{//; s/long int/longint/; s/short int/shortint/' | \
      awk '{print $1}' | sed 's/ //g'`
      if [[ "$datatype" = "struct" ]]
        then
        read line
        read line
        read line
        fld=`echo "$line" | sed "s/ //g; s/\;//; s/\}//"`
        echo "memset($tn.$fld.$fld""_data,'\\\0',sizeof($tn.$fld.$fld""_data));"
      elif [[ $datatype = "double" ]]
      then
        i=OK
      elif [[ $datatype = "longint" ]]
      then
        i=OK
      elif [[ $datatype = "shortint" ]]
      then
        i=OK
```

```
      elif [[ $datatype = "char" ]]
      then
        fld=`echo "$line" | sed "s/ //g; s/\[.*\]\;//" | \
    sed "s/\;//; s/char//; s/^{//"`
        echo "memset($tn.$fld,'\\\0',sizeof($tn.$fld));"
      else
        echo ""
        echo ""
 endtab=1
      fi
    done
  fi
  read line
done
}

# Start of main program
#
# Applies the parsing function and displays the result to standard output
#
cat $1 | dclgen_memset
```

A.13 genstrcpy.sh script

```
#!/usr/bin/ksh
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Shell script genstrcpy.sh
#
# Starting from a file containing the declaration of the
# host variables generated by DCLGEN DB2 utility,
# creates strcpy and sprintf C-language statements, for each table
# (from the DB fields into the rtb fields), for fields:
#
### VARCHAR ###
### DOUBLE ###
### LONG INT ###
### SHORT INT ###
### CHAR ###
#
# Syntax: genstrcpypcy.sh dclgen_input_file
#
# The script writes the statements on standard output. If you want to
# save them, redirect the stdout to a file. Example:
#
# genstrcpy.sh dclgen_input_file > output_file
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Function definition: parses the input file and creates the statements
#
function dclgen_strcpy
{
read line
while [ $? -eq 0 ]
do
  echo "$line" | grep "C DECLARATION FOR TABLE"
  if [ $? -eq 0 ]
  then
    tn1=`echo "$line" | awk '{print $6}' | sed 's/ //g'`
    tn="DCL"$tn1
    echo ""
    read line
    read line
    endtab=0
    while [ $endtab -eq 0 ]
    do
      read line
      datatype=`echo "$line" | \
      sed 's/{//; s/long int/longint/; s/short int/shortint/' | \
      awk '{print $1}' | sed 's/ //g'`
      if [[ "$datatype" = "struct" ]]
        then
        read line
        read line
        read line
        fld=`echo "$line" | sed "s/ //g; s/\;//; s/\}//"`
```

```
lowfld=`echo $fld | \
  sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
echo "strcpy(rtb->$lowfld,$tn.$fld.$fld""_data);"
      elif [[ $datatype = "double" ]]
      then
        fld=`echo "$line" | sed "s/ //g; s/\;//; s/double//; s/^{//"`
        lowfld=`echo $fld | \
          sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
        echo "sprintf(rtb->$lowfld,\"%.8f\",$tn.$fld);"
      elif [[ $datatype = "longint" ]]
      then
        fld=`echo "$line" | sed "s/ //g; s/\;//; s/longint//; s/^{//"`
        lowfld=`echo $fld | \
          sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
        echo "sprintf(rtb->$lowfld,\"%i\",$tn.$fld);"
      elif [[ $datatype = "shortint" ]]
      then
        fld=`echo "$line" | sed "s/ //g; s/\;//; s/shortint//; s/^{//"`
        lowfld=`echo $fld | \
          sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
        echo "sprintf(rtb->$lowfld,\"%i\",$tn.$fld);"
      elif [[ $datatype = "char" ]]
      then
        fld=`echo "$line" | sed "s/ //g; s/\[.*\]\;//" | \
                            sed "s/\;//; s/char//; s/^{//"`
        lowfld=`echo $fld | \
          sed "y/QWERTYUIOPASDFGHJKLZXCVBNM/qwertyuiopasdfghjklzxcvbnm/"`
        echo "strcpy(rtb->$lowfld,$tn.$fld);"
      else
        echo ""
        echo ""
 endtab=1
     fi
    done
  fi
  read line
done
}

# Start of main program
#
# Applies the parsing function and displays the result to standard output
#
cat $1 | dclgen_strcpy
```

Appendix B. Sample DB2 for OS/390 jobs

This appendix contains the OS/390 JCL used for executing various DB2 batch jobs during the project. Here you will find jobs for DB2 definition, for data objects definition, for program preparation and execution, and for creation and use of user defined functions.

B.1 JCL for base function compile

```
//CPRSCBAS JOB (999,POK),'PAOLOR1',CLASS=A,MSGCLASS=T,                  00000107
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000207
/*JOBPARM L=999,SYSAFF=SC63                                             00000307
//*********************************************************************00060007
//*                                                                     00060007
//*********************************************************************00060007
//JOBLIB  DD  DSN=CEE.SCEERUN,DISP=SHR                                  00070007
//        DD  DSN=DSN610B.SDSNLOAD,DISP=SHR                             00080007
//*
//*            COMPILE THE C PROGRAM IF THE PRECOMPILE
//*            RETURN CODE IS 4 OR LESS
//*
//C       EXEC PGM=CBCDRVR,REGION=4096K,
//        PARM='/LSEARCH(''CIPROSN.C.INCLUDE'') OPTF(DD:OF)'            00150007
//STEPLIB  DD  DSN=CBC.SCBCCMP,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//SYSLIB   DD  DSN=CEE.SCEEH.H,DISP=SHR
//         DD  DSN=CEE.SCEEH.SYS.H,DISP=SHR
//         DD  DSN=DSN610B.SDSNC.H,DISP=SHR
//         DD  DSN=GDDM.SADMSAM,DISP=SHR
//C.OF         DD *                                                     00230007
  SOURCE LIST MARGINS(1,72) NESTINC(255) LONGNAME RENT                  00240007
//SYSLIN   DD  DSN=CIPROSN.C.BASEDLL(BRIDGE),DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSCPRT  DD  SYSOUT=*
//SYSTERM  DD  DUMMY
//SYSIN    DD  DSN=CIPROSN.C.SOURCE(BRIDGE),DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT2   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT3   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT4   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT5   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT6   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT7   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT8   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT9   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10  DD  SYSOUT=*
//SYSUT14  DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//*
```

B.2 JCL for SQL function precompile and compile

```
//CPRSSQLC JOB (999,POK),'PAOLOR1',CLASS=A,MSGCLASS=T,                  00000107
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000207
/*JOBPARM L=999,SYSAFF=SC63                                            00000307
//*                                                                     00060007
//JOBLIB  DD  DSN=CEE.SCEERUN,DISP=SHR                                  00070007
//        DD  DSN=DSN610B.SDSNLOAD,DISP=SHR                             00080007
//*                                                                     00090007
//*        STEP PC: SQL PREPROCESSOR                                    00100007
//*                                                                     00110007
//PC      EXEC PGM=DSNHPC,
//         PARM='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',         00140007
//         REGION=4096K
//STEPLIB  DD  DISP=SHR,DSN=DSN610B.SDSNEXIT.SC63
//         DD  DISP=SHR,DSN=DSN610B.SDSNLOAD
//DBRMLIB  DD DSN=CIPROSN.C.DBRMLIB(SQLLOG),                            00170007
//            DISP=SHR                                                  00180007
//SYSLIB   DD DSN=CIPROSN.C.INCLUDE,                                    00190007
//            DISP=SHR                                                  00200007
//SYSIN    DD DSN=CIPROSN.C.SOURCE(SQLLOG),                             00210007
//            DISP=SHR                                                  00220007
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSCIN   DD  DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=SYSALLDA,
//             SPACE=(800,(500,500))
//SYSUT1   DD  SPACE=(800,(500,500),,,ROUND),UNIT=SYSALLDA
//SYSUT2   DD  SPACE=(800,(500,500),,,ROUND),UNIT=SYSALLDA
//*
//*            COMPILE THE C PROGRAM IF THE PRECOMPILE
//*            RETURN CODE IS 4 OR LESS
//*
//C       EXEC PGM=CBCDRVR,COND=(4,LT,PC),REGION=4096K,
//        PARM='/LSEARCH(''CIPROSN.C.INCLUDE'') OPTF(DD:OF)'             00150007
//STEPLIB  DD  DSN=CBC.SCBCCMP,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//SYSLIB   DD  DSN=CEE.SCEEH.H,DISP=SHR
//         DD  DSN=CEE.SCEEH.SYS.H,DISP=SHR
//         DD  DSN=DSN610B.SDSNC.H,DISP=SHR
//         DD  DSN=GDDM.SADMSAM,DISP=SHR
//C.OF         DD *                                                     00230007
  SOURCE LIST MARGINS(1,72) NESTINC(255) LONGNAME RENT                  00240007
//SYSLIN   DD  DSN=CIPROSN.C.SQLDLL(SQLLOG),DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSCPRT  DD  SYSOUT=*
//SYSTERM  DD  DUMMY
//SYSIN    DD  DSN=&&DSNHOUT,DISP=(OLD,DELETE)
//SYSUT1   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT2   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT3   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT4   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT5   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT6   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT7   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT8   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT9   DD  UNIT=SYSDA,DISP=(NEW,DELETE),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10  DD  SYSOUT=*
//SYSUT14  DD  UNIT=SYSDA,DISP=(NEW,DELETE),
```

```
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//*
```

B.3 JCL for compile, prelink and link of main programs

```
//CPRSGLOB JOB (999,POK),'PAOLOR2',CLASS=A,
//   MSGCLASS=T,NOTIFY=&SYSUID,MSGLEVEL=(1,1),
//   REGION=0M
/*JOBPARM  T=1,L=50,SYSAFF=SC63
//*********************************************************************
//*
//*    CHANGE ALL >>>>>>>> RTDIN  <<<<<<<<< TO WHATEVER IS NEXT
//*
//*********************************************************************
//*-------------------------------------------------------------------*
//* COMPILE   C    DLL PROGRAM
//*-------------------------------------------------------------------*
//JOBLIB  DD  DSN=CEE.SCEERUN,DISP=SHR
//        DD  DSN=DSN610B.SDSNLOAD,DISP=SHR
//COMP01   EXEC  EDCC,
//    INFILE='',
//  CPARM='OPTFILE(DD:MYOPT)'
//COMPILE.SYSIN DD DSN=CIPROSN.C.SOURCE(RTDIN),DISP=SHR
//COMPILE.SYSLIN DD DSN=CIPROSN.C.OBJ(RTDIN),DISP=SHR
//COMPILE.MYOPT DD  *
  LSEARCH('CIPROSN.C.INCLUDE')
  LO DLL SO SSCOMM SOURCE LIST MARGINS(1,72) NESTINC(255) LONGNAME RENT
//*-------------------------------------------------------------------*
//* PRELINK/LINK   MYDLL1 AS A DLL
//*-------------------------------------------------------------------*
//          EXEC EDCPL
//PLKED.SYSIN  DD  DSN=CIPROSN.C.OBJ(RTDIN),DISP=SHR
//       DD  DSN=CIPROSN.C.SQLDLL(SQLERR),DISP=SHR
//       DD  DSN=CIPROSN.C.SQLDLL(SQLINS),DISP=SHR
//       DD  DSN=CIPROSN.C.SQLDLL(SQLLOG),DISP=SHR
//       DD  DSN=CIPROSN.C.SQLDLL(SQLSEL),DISP=SHR
//       DD  DSN=CIPROSN.C.SQLDLL(SQLUPD),DISP=SHR
//       DD  DSN=CIPROSN.C.SQLDLL(SQLUTIL),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(BRIDGE),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(CORERTD),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(DINEXEC),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(FINDTYP),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(INSLAB),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(LOGEV),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(MAPPING),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(READCFG),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(UTIL),DISP=SHR
//       DD  DSN=CIPROSN.C.BASEDLL(WRTLOG),DISP=SHR
//       DD  *
  IMPORT DATA 'libhttpdapi.so' all_api_callbacks
  IMPORT DATA 'libhttpdapi.so' asciitoebcdic
  IMPORT DATA 'libhttpdapi.so' badthread
  IMPORT DATA 'libhttpdapi.so' ebcdictoascii
  IMPORT CODE 'libhttpdapi.so' set_api_callbacks
  IMPORT CODE 'libhttpdapi.so' HTAPIdata_delete
  IMPORT CODE 'libhttpdapi.so' HTAPIdata_new
  IMPORT CODE 'libhttpdapi.so' HTAUTHEN
  IMPORT CODE 'libhttpdapi.so' HTCGI_new
  IMPORT CODE 'libhttpdapi.so' HTEXEC
  IMPORT CODE 'libhttpdapi.so' HTFilterstream_new
  IMPORT CODE 'libhttpdapi.so' HTFILE
  IMPORT CODE 'libhttpdapi.so' HTLOGE
  IMPORT CODE 'libhttpdapi.so' HTREAD
  IMPORT CODE 'libhttpdapi.so' HTREST
  IMPORT CODE 'libhttpdapi.so' HTSET
  IMPORT CODE 'libhttpdapi.so' HTWRITE
  IMPORT CODE 'libhttpdapi.so' HTXTRACT
  IMPORT DATA 'libhttpdapi.so' WWW_ENC_EBCDIC
  IMPORT DATA 'libhttpdapi.so' WWW_ENC_7BIT
  IMPORT DATA 'libhttpdapi.so' WWW_ENC_8BIT
//PLKED.SYSDEFSD DD DUMMY
//LKED.SYSLMOD   DD DSN=CIPROSN.C.LOADLIB(RTDIN),DISP=SHR
//LKED.RUNLIB    DD DSN=CIPROSN.C.LOADLIB,DISP=SHR
//LKED.SYSIN     DD *
```

```
    INCLUDE SYSLIB(DSNELI)
    INCLUDE RUNLIB(RTDIN)
/*
```

B.4 JCL for running the main programs RTDIN and LABIN

```
//CPRSRTDI JOB (999,POK),'PAOLOR2',CLASS=A,MSGCLASS=T,                  00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                             00000003
//*                                                                     00250000
//JOBLIB  DD  DSN=CEE.SCEERUN,DISP=SHR                                  00260000
//        DD  DSN=DSN610B.SDSNLOAD,DISP=SHR                             00270000
//*                                                                     00280000
//*        STEP 1 : RUN PROGRAM RTDIN                                   00630000
//*                                                                     00280000
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                     00640000
//MYVARS    DD  DSN=CIPROSN.C.ENVARS,DISP=SHR                           00660000
//SYSERR    DD  SYSOUT=*                                                00660000
//SYSPRINT  DD  SYSOUT=*                                                00670000
//SYSTSPRT  DD  SYSOUT=*                                                00670000
//CEEDUMP   DD  SYSOUT=*                                                00680000
//SYSUDUMP  DD  SYSOUT=*                                                00690000
//SYSOUT    DD  SYSOUT=*                                                00700000
//SYSTSIN   DD  *                                                       00740000
 DSN SYSTEM(DB2X)                                                       00750000
 RUN  PROGRAM(RTDIN) PLAN(CIPROS1)  -                                   00770000
      LIB('CIPROSN.C.LOADLIB') -                                        00780000
      PARM ('ENVAR("_CEE_ENVFILE=DD:MYVARS")/V060308D')                 00660000
 END                                                                    00810000
//*                                                                     00900000
```

B.5 JCL for creation of storage group, database, table spaces and tables

```
//CPRSDB JOB (999,POK),'CIPROSDB',CLASS=A,MSGCLASS=T,                   00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                             00000003
//*                                                                     00002100
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                        00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                             00002200
//*                                                                     00005300
//*        STEP  1: CREATE CIPROS STORAGE GROUPS, DATABASE, TABLESPACES 00005400
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                     00005500
//SYSTSPRT DD SYSOUT=*                                                  00005600
//SYSTSIN  DD *                                                         00005700
 DSN SYSTEM(DB2X)                                                       00005800
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -                                 00005900
      LIB('DB2V610X.RUNLIB.LOAD')                                       00006000
//SYSPRINT DD SYSOUT=*                                                  00006100
//SYSUDUMP DD SYSOUT=*                                                  00006200
//SYSIN    DD *                                                         00006300

   CREATE DATABASE CIPROS
      STOGROUP CIPROS01
      BUFFERPOOL BP0
      CCSID EBCDIC;

   CREATE TABLESPACE CPRSBASE
      IN CIPROS
      USING STOGROUP CIPROS01
                PRIQTY 150000
                SECQTY 20
                ERASE NO
      LOCKSIZE PAGE LOCKMAX SYSTEM
      BUFFERPOOL BP0
      CLOSE NO
      SEGSIZE 16
      CCSID EBCDIC;

   CREATE TABLESPACE CPRSLAB
      IN CIPROS
      USING STOGROUP CIPROS01
                PRIQTY 10000
                SECQTY 20
```

```
                ERASE NO
      LOCKSIZE PAGE LOCKMAX SYSTEM
      BUFFERPOOL BP0
      CLOSE NO
      SEGSIZE 16
      CCSID EBCDIC;

   CREATE TABLESPACE CPRSLOGE
      IN CIPROS
      USING STOGROUP CIPROS01
                PRIQTY 10000
                SECQTY 20
                ERASE NO
      LOCKSIZE PAGE LOCKMAX SYSTEM
      BUFFERPOOL BP0
      SEGSIZE 16
      CLOSE NO
      CCSID EBCDIC;

   CREATE TABLESPACE CPRSREAD
      IN CIPROS
      USING STOGROUP CIPROS01
                PRIQTY 5000
                SECQTY 20
                ERASE NO
      LOCKSIZE PAGE LOCKMAX SYSTEM
      BUFFERPOOL BP0
      CLOSE NO
      SEGSIZE 16
      CCSID EBCDIC;

   CREATE TABLESPACE CPRSPRDS
      IN CIPROS
      USING STOGROUP CIPROS01
                PRIQTY 10000
                SECQTY 20
                ERASE NO
      LOCKSIZE PAGE LOCKMAX SYSTEM
      BUFFERPOOL BP0
      CLOSE NO
      SEGSIZE 16
      CCSID EBCDIC;                                                    00017200
COMMIT;                                                                00017300
//*                                                                    00017500
//*        STEP  2: CREATE CIPROS TABLES, VIEWS                        00017600
//STEP002 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                    00017700
//SYSTSPRT DD  SYSOUT=*                                                00017800
//SYSTSIN  DD  *                                                       00017900
 DSN SYSTEM(DB2X)                                                      00018000
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -                                00018100
      LIB('DB2V610X.RUNLIB.LOAD')                                      00018200
//SYSPRINT DD  SYSOUT=*                                                00018300
//SYSUDUMP DD  SYSOUT=*                                                00018400
//SYSIN    DD  *                                                       00018500
CREATE TABLE ANALYSIS_METHOD (ANLY_METH_NAME VARCHAR(20) NOT NULL,
             DESCRIPTION VARCHAR(40))
             IN CIPROS.CPRSBASE;
CREATE TABLE ANALYSIS_SPEC (MTRL_NAME VARCHAR(15) NOT NULL,
             STD_RESLT_NAME VARCHAR(20) NOT NULL,
             DESCRIPTION VARCHAR(40),
             HI_VALUE VARCHAR(20),
             LO_VALUE VARCHAR(20))
IN CIPROS.CPRSBASE;
.
.
.
CREATE TABLE USRID_CONFIG (CODE INTEGER NOT NULL,
             USR_NAME VARCHAR(20) NOT NULL,
             VARIABLE_VALUE VARCHAR(40))
             IN CIPROS.CPRSBASE;
CREATE TABLE VALUE_TYPE (VALUE_TYPE VARCHAR(6) NOT NULL,
             DESCRIPTION VARCHAR(40))
             IN CIPROS.CPRSBASE;
COMMIT;                                                                00060700
```

B.6 JCL for creation of indexes for CIPROS tables

```
//CPRSIX JOB (999,POK),'CIPROSIX',CLASS=A,MSGCLASS=T,                  00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                  00000002
/*JOBPARM L=999,SYSAFF=SC63                                           00000003
//*                                                                   00002100
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                      00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                           00002200
//*                                                                   00005300
//*        STEP  1: CREATE CIPROS INDEXES                             00005400
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                   00005500
//SYSTSPRT DD SYSOUT=*                                                00005600
//SYSTSIN  DD *                                                       00005700
 DSN SYSTEM(DB2X)                                                     00005800
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -                               00005900
      LIB('DB2V610X.RUNLIB.LOAD')                                     00006000
//SYSPRINT DD SYSOUT=*                                                00006100
//SYSUDUMP DD SYSOUT=*                                                00006200
//SYSIN    DD *                                                       00006300
CREATE UNIQUE INDEX  ANALYSIS_METHODP1
             ON  ANALYSIS_METHOD
             (ANLY_METH_NAME)
             USING STOGROUP CIPROS01
                   PRIQTY 400
                   SECQTY 40
                   ERASE NO;
CREATE UNIQUE INDEX  ANALYSIS_SPECP1
             ON  ANALYSIS_SPEC
             (MTRL_NAME,STD_RESLT_NAME)
             USING STOGROUP CIPROS01
                   PRIQTY 400
                   SECQTY 40
                   ERASE NO;
.
.
.
CREATE UNIQUE INDEX  VALUE_TYPEP1
             ON  VALUE_TYPE
             (VALUE_TYPE)
             USING STOGROUP CIPROS01
                   PRIQTY 400
                   SECQTY 40
                   ERASE NO;
CREATE UNIQUE INDEX  MAP_FACILITYP1
             ON  MAP_FACILITY
             (FACILITY_ENV,ENV_NAME)
             USING STOGROUP CIPROS01
                   PRIQTY 400
                   SECQTY 40
                   ERASE NO;
COMMIT;                                                               00060700
ALTER TABLE  ANALYSIS_METHOD
      ADD PRIMARY KEY (ANLY_METH_NAME);
ALTER TABLE  ANALYSIS_SPEC  ADD PRIMARY KEY
             (MTRL_NAME,STD_RESLT_NAME);
.
.
.
ALTER TABLE  VALUE_TYPE  ADD PRIMARY KEY
             (VALUE_TYPE);
ALTER TABLE  MAP_FACILITY  ADD PRIMARY KEY
             (FACILITY_ENV,ENV_NAME);
COMMIT;                                                               00060700
CREATE INDEX ANALYSIS_SPECI2
             ON ANALYSIS_SPEC
             (STD_RESLT_NAME )
             USING STOGROUP CIPROS01
                   PRIQTY 400
                   SECQTY 40
                   ERASE NO;
.
.
.
CREATE UNIQUE INDEX  UNITC2
             ON  UNIT
             (UNT_NUM)
             USING STOGROUP CIPROS01
```

```
                    PRIQTY 400
                    SECQTY 40
                    ERASE NO;

COMMIT;                                                                 00060700
//*                                                                     00139300
```

B.7 JCL to alter tables for foreign keys

```
//CPRSFK JOB (999,POK),'CIPROSFK',CLASS=A,MSGCLASS=T,                   00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//*                                                                     00002100
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                        00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                             00002200
//*                                                                     00005300
//*        STEP  1: CREATE SAMPLE STORAGE GROUPS, TABLESPACES           00005400
//PH01S01 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                     00005500
//SYSTSPRT DD SYSOUT=*                                                  00005600
//SYSTSIN  DD *                                                         00005700
 DSN SYSTEM(DB2X)                                                       00005800
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -                                 00005900
      LIB('DB2V610X.RUNLIB.LOAD')                                       00006000
//SYSPRINT DD SYSOUT=*                                                  00006100
//SYSUDUMP DD SYSOUT=*                                                  00006200
//SYSIN    DD *                                                         00006300

ALTER TABLE ANALYSIS_SPEC
            FOREIGN KEY (MTRL_NAME)
            REFERENCES MATERIAL (MTRL_NAME);
.
.
.
ALTER TABLE USRID_CONFIG
            FOREIGN KEY (CODE)
            REFERENCES CIPROS_CONFIG (CODE);
                                                                        00018600
COMMIT;                                                                 00060700
//*                                                                     00139300
```

B.8 JCL for synonym creation

```
//CPRSSY JOB (999,POK),'CIPROSSY',CLASS=A,MSGCLASS=T,                   00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//*                                                                     00002100
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                        00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                             00002200
//*                                                                     00005300
//*        STEP  1: CREATE CIPROS SYNONYMS                              00005400
//PH01S01 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                     00005500
//SYSTSPRT DD SYSOUT=*                                                  00005600
//SYSTSIN  DD *                                                         00005700
 DSN SYSTEM(DB2X)                                                       00005800
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -                                 00005900
      LIB('DB2V610X.RUNLIB.LOAD')                                       00006000
//SYSPRINT DD SYSOUT=*                                                  00006100
//SYSUDUMP DD SYSOUT=*                                                  00006200
//SYSIN    DD *                                                         00006300

GRANT CREATEALIAS TO CIPROS;
SET CURRENT SQLID = 'CIPROS';
CREATE ALIAS ANALYSIS_METHOD FOR PAOLOR2.ANALYSIS_METHOD;
CREATE ALIAS ANALYSIS_SPEC FOR PAOLOR2.ANALYSIS_SPEC;
.
.
.
CREATE ALIAS MAP_FACILITY  FOR PAOLOR2.MAP_FACILITY;
                                                                        00018600
COMMIT;                                                                 00060700
//*                                                                     00139300
```

B.9 JCL for creation of CIPROS views

```
//CPRSVW JOB (999,POK),'CIPROSDB',CLASS=A,MSGCLASS=T,                  00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                  00000002
/*JOBPARM L=999,SYSAFF=SC63                                           00000003
//*                                                                    00002100
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                       00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                            00002200
//*                                                                    00005300
//*        STEP  1: CREATE CIPROS VIEWS                                00005400
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                    00005500
//SYSTSPRT DD SYSOUT=*                                                 00005600
//SYSTSIN  DD *                                                        00005700
 DSN SYSTEM(DB2X)                                                     00005800
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -                               00005900
      LIB('DB2V610X.RUNLIB.LOAD')                                     00006000
//SYSPRINT DD SYSOUT=*                                                 00006100
//SYSUDUMP DD SYSOUT=*                                                 00006200
//SYSIN    DD *                                                        00006300
CREATE VIEW CASES_GLOBAL
(CASE_NAME,
LOCATION,
CASE_OWNER,
CASE_TITLE,
DESCRIPTION,
PERIOD_NAME,
START_DATE,
END_DATE,
CASE_TIME_RES,
DATE_CHANGE,
DATA_STORED) AS
SELECT CASE_NAME,
LOCATION,
CASE_OWNER,
CASE_TITLE,
DESCRIPTION,
PERIOD_NAME,
START_DATE,
END_DATE,
CASE_TIME_RES,
TIMESTAMP_CHANGE,
DATA_STORED FROM CASES WHERE LOCATION = 'GLOBAL';

CREATE VIEW CORE_TABLES (TABLE_NAME,ACCESS) AS
SELECT
TTNAME,
CASE SELECTAUTH WHEN 'Y' THEN 'S'
                WHEN 'G' THEN 'S' ELSE '-' END ³³
CASE UPDATEAUTH WHEN 'Y' THEN 'U'
                WHEN 'G' THEN 'U' ELSE '-' END ³³
CASE INSERTAUTH WHEN 'Y' THEN 'I'
                WHEN 'G' THEN 'I' ELSE '-' END ³³
CASE DELETEAUTH WHEN 'Y' THEN 'D'
                WHEN 'G' THEN 'D' ELSE '-' END
FROM SYSIBM.SYSTABAUTH
WHERE TCREATOR='PAOLOR2'
AND GRANTEE=USER;
COMMIT;                                                              00060700

SET CURRENT SQLID = 'CIPROS';

CREATE ALIAS CIPROS.CORE_TABLES FOR PAOLOR2.CORE_TABLES;
CREATE ALIAS CIPROS.CASES_GLOBAL FOR PAOLOR2.CASES_GLOBAL;

COMMIT;                                                              00060700
```

B.10 JCL for deletion of CIPROS database and table spaces

```
//CPRSDLT JOB (999,POK),'CIPROSDB',CLASS=A,MSGCLASS=T,                00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                  00000002
/*JOBPARM L=999,SYSAFF=SC63                                           00000003
//*                                                                    00002100
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                       00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                            00002200
```

```
//*                                                                     00005300
//*        STEP  1: CREATE CIPROS STORAGE GROUPS, DATABASE, TABLESPACES 00005400
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                     00005500
//SYSTSPRT DD SYSOUT=*                                                  00005600
//SYSTSIN  DD *                                                         00005700
 DSN SYSTEM(DB2X)                                                       00005800
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61) -                                 00005900
      LIB('DB2V610X.RUNLIB.LOAD')                                       00006000
//SYSPRINT DD SYSOUT=*                                                  00006100
//SYSUDUMP DD SYSOUT=*                                                  00006200
//SYSIN    DD *                                                         00006300
  DROP TABLESPACE CIPROS.CPRSBASE;
  COMMIT;                                                               00060700
  DROP TABLESPACE CIPROS.CPRSLAB;
  COMMIT;                                                               00060700
  DROP TABLESPACE CIPROS.CPRSLOGE;
  COMMIT;                                                               00060700
  DROP TABLESPACE CIPROS.CPRSREAD;
  COMMIT;                                                               00060700
  DROP TABLESPACE CIPROS.CPRSPRDS;
  COMMIT;                                                               00060700
  DROP DATABASE CIPROS;
  COMMIT;                                                               00060700
  DROP STOGROUP CIPROS01;
  COMMIT;                                                               00060700
//*                                                                     00139300
```

B.11 JCL for REORG, RUNSTATS and COPY of CIPROS table spaces

```
//CPRSRRIC JOB (999,POK),'DB2V610X',CLASS=A,MSGCLASS=T,                 00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//JOBLIB  DD  DSN=DSN610B.SDSNEXIT.SC63,DISP=SHR                        00002200
//        DD  DSN=DSN610B.SDSNLOAD,DISP=SHR                             00002200
//*                                                                     00002300
//* JOB TO REORG, DO RUNSTATS AND TAKE IMAGE COPIES OF TABLESPACES      00002300
//* COMPRISING THE CIPROS DATABASE.                                     00002300
//*                                                                     00002300
//* WHILE DB2 VERSION 6 ALLOWS THE CONCATINATION OF REORGS, RUNSTATS    00002300
//* AND IMAGE COPIES, THESE ARE IN SEPERATE STEPS FOR CLARITY.          00002300
//*                                                                     00002300
//*  STEP000: STOP CIPROS DATABASE AND RESTART IT WITH ACCESS UTILITY   00114100
//*           THIS IS INSURANCE AGAINST OTHER ACCESS DURING REORGS      00114100
//*                                                                     00002300
//STEP001 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                  00114200
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)               00114400
//SYSIN    DD  *                                                        00114500
                                                                        00114600
 QUIESCE TABLESPACE CIPROS.CPRSBASE                                     00114700
         TABLESPACE CIPROS.CPRSLAB                                      00114700
         TABLESPACE CIPROS.CPRSLOGE                                     00114700
         TABLESPACE CIPROS.CPRSREAD                                     00114700
         TABLESPACE CIPROS.CPRSPRDS                                     00114700
                                                                        00115000
//*                                                                     00118600
//*        STEP002: REORGANIZE TABLESPACE CPRSBASE                      00118600
//*                                                                     00118600
//STEP002 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                  00118700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                                00118800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)               00118900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)               00119000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)               00119100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)               00119200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)               00119300
//DSNTRACE DD  SYSOUT=*                                                 00119400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)               00119500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)             00119600
//SYSIN    DD  *                                                        00119700
                                                                        00119800
 REORG TABLESPACE CIPROS.CPRSBASE SORTDEVT(SYSDA)                       00119900
//*                                                                     00118600
//*        STEP003: REORGANIZE TABLESPACE  CPRSLAB                      00118600
//*                                                                     00118600
//STEP003 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                  00118700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                                00118800
```

```
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00118900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119300
//DSNTRACE DD  SYSOUT=*                                                00119400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)              00119500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)            00119600
//SYSIN    DD  *                                                       00119700
                                                                       00119800
 REORG TABLESPACE CIPROS.CPRSLAB  SORTDEVT(SYSDA)                      00119900
//*                                                                    00118600
//*        STEP004: REORGANIZE TABLESPACE CPRSLOGE                     00118600
//*                                                                    00118600
//STEP004 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00118700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00118800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00118900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119300
//DSNTRACE DD  SYSOUT=*                                                00119400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)              00119500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)            00119600
//SYSIN    DD  *                                                       00119700
                                                                       00119800
 REORG TABLESPACE CIPROS.CPRSLOGE SORTDEVT(SYSDA)                      00119900
//*                                                                    00118600
//*        STEP005: REORGANIZE TABLESPACE CPRSREAD                     00118600
//*                                                                    00118600
//STEP005 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00118700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00118800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00118900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119300
//DSNTRACE DD  SYSOUT=*                                                00119400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)              00119500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)            00119600
//SYSIN    DD  *                                                       00119700
                                                                       00119800
 REORG TABLESPACE CIPROS.CPRSREAD SORTDEVT(SYSDA)                      00119900
//*                                                                    00118600
//*        STEP006: REORGANIZE TABLESPACE CPRSPRDS                     00118600
//*                                                                    00118600
//STEP006 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00118700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00118800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00118900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00119300
//DSNTRACE DD  SYSOUT=*                                                00119400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)              00119500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)            00119600
//SYSIN    DD  *                                                       00119700
                                                                       00119800
 REORG TABLESPACE CIPROS.CPRSPRDS SORTDEVT(SYSDA)                      00119900
//*                                                                    00118600
//*        STEP007: PRODUCE STATISTICS FOR CPRSBASE                    00137500
//*                                                                    00137600
//STEP007 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00137700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00137800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00137900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00138000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00138100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00138200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)              00138300
//DSNTRACE DD  SYSOUT=*                                                00138400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)              00138500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)            00138600
//SYSIN    DD  *                                                       00138700
                                                                       00120600
 RUNSTATS TABLESPACE CIPROS.CPRSBASE                                   00120700
         INDEX(ALL)                                                    00120800
                                                                       00139200
//*                                                                    00118600
```

```
//*        STEP008: PRODUCE STATISTICS FOR CPRSLAB                    00137500
//*                                                                   00137600
//STEP008 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)               00137700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                             00137800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00137900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138300
//DSNTRACE DD  SYSOUT=*                                              00138400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)            00138500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)          00138600
//SYSIN    DD  *                                                     00138700
                                                                     00138800
                                                                     00120600
 RUNSTATS TABLESPACE CIPROS.CPRSLAB                                  00120700
         INDEX(ALL)                                                  00120800
                                                                     00139200
//*                                                                  00118600
//*        STEP009: PRODUCE STATISTICS FOR CPRSLOGE                   00137500
//*                                                                   00137600
//STEP009 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)               00137700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                             00137800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00137900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138300
//DSNTRACE DD  SYSOUT=*                                              00138400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)            00138500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)          00138600
//SYSIN    DD  *                                                     00138700
                                                                     00138800
                                                                     00120600
 RUNSTATS TABLESPACE CIPROS.CPRSLOGE                                 00120700
         INDEX(ALL)                                                  00120800
                                                                     00139200
//*                                                                  00118600
//*        STEP010: PRODUCE STATISTICS FOR CPRSREAD                   00137500
//*                                                                   00137600
//STEP010 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)               00137700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                             00137800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00137900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138300
//DSNTRACE DD  SYSOUT=*                                              00138400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)            00138500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)          00138600
//SYSIN    DD  *                                                     00138700
                                                                     00138800
                                                                     00120600
 RUNSTATS TABLESPACE CIPROS.CPRSREAD                                 00120700
         INDEX(ALL)                                                  00120800
                                                                     00139200
//*                                                                  00118600
//*        STEP011: PRODUCE STATISTICS FOR CPRSPRDS                   00137500
//*                                                                   00137600
//STEP011 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)               00137700
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                             00137800
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00137900
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138000
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138100
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138200
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(20,20),,,ROUND)            00138300
//DSNTRACE DD  SYSOUT=*                                              00138400
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(50,50),,,ROUND)            00138500
//SYSREC   DD  UNIT=SYSALLDA,SPACE=(4000,(200,200),,,ROUND)          00138600
//SYSIN    DD  *                                                     00138700
                                                                     00138800
                                                                     00120600
 RUNSTATS TABLESPACE CIPROS.CPRSPRDS                                 00120700
         INDEX(ALL)                                                  00120800
                                                                     00139200
//*                                                                  00139300
//*        STEP012: TAKE IMAGE COPY OF TABLESPACE CPRSBASE            00115100
//*                                                                   00139300
```

```
//STEP012 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                  00115200
//DSNTRACE DD  SYSOUT=*                                                 00115300
//SYSCOPY  DD  DSN=CIPROS.COPY.BASE0001,                                00115400
//             UNIT=3390,DISP=(NEW,CATLG),SPACE=(4000,(200,20)),        00115500
//             VOL=SER=SBOX09                                           00115600
//SYSIN    DD  *                                                        00116600
                                                                        00116700
 COPY TABLESPACE CIPROS.CPRSBASE                                        00116800
//*                                                                     00139300
//*        STEP013: TAKE IMAGE COPY OF TABLESPACE CPRSLAB               00115100
//*                                                                     00139300
//STEP013 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                  00115200
//DSNTRACE DD  SYSOUT=*                                                 00115300
//SYSCOPY  DD  DSN=CIPROS.COPY.LAB0001,                                 00115400
//             UNIT=3390,DISP=(NEW,CATLG),SPACE=(4000,(200,20)),        00115500
//             VOL=SER=SBOX09                                           00115600
//SYSIN    DD  *                                                        00116600
                                                                        00116700
 COPY TABLESPACE CIPROS.CPRSLAB                                         00116800
//*                                                                     00139300
//*        STEP014: TAKE IMAGE COPY OF TABLESPACE CPRSLOGE              00115100
//*                                                                     00139300
//STEP014 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                  00115200
//DSNTRACE DD  SYSOUT=*                                                 00115300
//SYSCOPY  DD  DSN=CIPROS.COPY.LOGE0001,                                00115400
//             UNIT=3390,DISP=(NEW,CATLG),SPACE=(4000,(200,20)),        00115500
//             VOL=SER=SBOX09                                           00115600
//SYSIN    DD  *                                                        00116600
                                                                        00116700
 COPY TABLESPACE CIPROS.CPRSLOGE                                        00116800
//*                                                                     00139300
//*        STEP015: TAKE IMAGE COPY OF TABLESPACE CPRSREAD              00115100
//*                                                                     00139300
//STEP015 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                  00115200
//DSNTRACE DD  SYSOUT=*                                                 00115300
//SYSCOPY  DD  DSN=CIPROS.COPY.READ0001,                                00115400
//             UNIT=3390,DISP=(NEW,CATLG),SPACE=(4000,(200,20)),        00115500
//             VOL=SER=SBOX09                                           00115600
//SYSIN    DD  *                                                        00116600
                                                                        00116700
 COPY TABLESPACE CIPROS.CPRSREAD                                        00116800
//*                                                                     00139300
//*        STEP016: TAKE IMAGE COPY OF TABLESPACE CPRSPRDS              00115100
//*                                                                     00139300
//STEP016 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                  00115200
//DSNTRACE DD  SYSOUT=*                                                 00115300
//SYSCOPY  DD  DSN=CIPROS.COPY.PRDS0001,                                00115400
//             UNIT=3390,DISP=(NEW,CATLG),SPACE=(4000,(200,20)),        00115500
//             VOL=SER=SBOX09                                           00115600
//SYSIN    DD  *                                                        00116600
                                                                        00116700
 COPY TABLESPACE CIPROS.CPRSPRDS                                        00116800
//*                                                                     00005500
```

B.12 JCL for RECOVER of a CIPROS table space

```
//CPRSRCVY JOB (999,POK),'CIPROSDB',CLASS=A,MSGCLASS=T,                 00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//*                                                                     00002100
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                        00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                             00002200
//*                                                                     00005300
//STEP001 EXEC PGM=DSNUTILB,PARM='DB2X,DSNTEX'                          00005500
//SYSPRINT DD SYSOUT=*                                                  00006100
//UTPRINT  DD SYSOUT=*                                                  00006100
//SYSUDUMP DD SYSOUT=*                                                  00006200
//SYSIN    DD *                                                         00006300
 RECOVER TABLESPACE CIPROS.CPRSBASE                                     00005900
```

B.13 JCL for rebuilding a CIPROS index

```
//CPRSRBLD JOB (999,POK),'CIPROSDB',CLASS=A,MSGCLASS=T,                 00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//*                                                                     00002100
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                        00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                             00002200
//*                                                                     00005300
//STEP001 EXEC PGM=DSNUTILB,PARM='DB2X,DSNTEX'                          00005500
//SYSPRINT DD SYSOUT=*                                                  00006100
//UTPRINT  DD SYSOUT=*                                                  00006100
//SYSUDUMP DD SYSOUT=*                                                  00006200
//SYSIN    DD *                                                         00006300
 REBUILD INDEX ALL TABLESPACE CIPROS.CPRSBASE                           00005900
```

B.14 JCL to produce C language table structures (DCLGEN)

```
//CPRSDGEN JOB (999,POK),'DB2V610X',CLASS=A,MSGCLASS=T,                 00000000
//         NOTIFY=PAOLOR2,TIME=1440,REGION=0M                           00000100
//*                                                                     00000208
//* ********************************************************************00000308
//* THIS JOB CREATES OR REPLACES DCLGENS FOR THE CIPROS EXAMPLE TABLES. 00000508
//*       EVERY TABLE IS INCLUDED IN THIS JOB.  FOR INDIVIDUAL TABLE    00000608
//*       CHANGES, A ONE STEP JOB WITH THE PROPER PARAMETERS COULD BE   00000708
//*       SUBMITTED OR YOU COULD USE DB2I OPTION 2.                     00000808
//*                                                                     00000908
//* NOTE: IF RUNNING A BATCH DCLGEN FOR THE FIRST TIME AND THE DESIRED  00001008
//*       PDF MEMBER DOES NOT EXIST SET THE PARAMETER FOR ACTION TO:    00001108
//*                ACTION (ADD)                                         00001208
//*       THE UTILITY WILL CREATE THE MEMBER NAMED IN THE LIBRARY PARM  00001308
//*       IF RUNNING A BATCH DCLGEN SUBSEQUENTLY FOR THE SAME TABLE     00001408
//*       SET THE PARAMETER FOR ACTION TO:                              00001508
//*                ACTION (REPLACE)                                     00001608
//*       THE UTILITY WILL REPLACE THE MEMBER NAMED IN THE LIBRARY PARM 00001708
//*                                                                     00001808
//* JOB CARDS, JOBPARM CARD, JOBLIB, STEPLIB, DATASET NAMES AND OTHER   00001908
//*       INFORMATION WILL HAVE TO BE CHANGED TO REFLECT CONDITIONS,    00002008
//*       NAMING STANDARDS AND PREFERENCES AT YOUR SITE.                00002108
//* ********************************************************************00002208
//*                                                                     00002308
/*JOBPARM L=999,SYSAFF=SC63                                            00002408
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                        00002508
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                             00002608
//        DD  DSN=DB2V610X.RUNLIB.LOAD,DISP=SHR                         00002708
//STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                     00002808
//SYSTSPRT DD  SYSOUT=*                                                 00002908
//SYSPRINT DD  SYSOUT=*                                                 00003008
//SYSUDUMP DD  SYSOUT=*                                                 00003108
//SYSTSIN  DD  *                                                        00003208
 DSN SYSTEM(DB2X)                                                       00003308
 DCLGEN TABLE(ANALYSIS_METHOD) -                                        00003408
        LIBRARY('CIPROSN.C.DCLGEN(ANMETHOD)') -                         00003508
        LANGUAGE(C) -                                                   00003608
        ACTION (REPLACE) -                                              00003708
        APOST                                                           00003808
.
.
.
//STEP064 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                     06180007
//SYSTSPRT DD  SYSOUT=*                                                 06190001
//SYSPRINT DD  SYSOUT=*                                                 06200001
//SYSUDUMP DD  SYSOUT=*                                                 06210001
//SYSTSIN  DD  *                                                        06220001
 DSN SYSTEM(DB2X)                                                       06230001
 DCLGEN TABLE(VALUE_TYPE) -                                             06240003
        LIBRARY('CIPROSN.C.DCLGEN(VALUETYP)') -                         06250001
        LANGUAGE(C) -                                                   06260001
        ACTION (REPLACE) -                                              06261005
        APOST                                                           06270001
```

B.15 JCL for first job to LOAD CIPROS tables

```
//CPRSLOD1 JOB (999,POK),'PAOLOR2 ',CLASS=A,MSGCLASS=T,                 00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                       00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                            00002200
//*****************************************************************00105600
//**   TABLES LOADED WITH THIS JCL ARE THOSE CIPROS TABLES THAT HAVE    00105600
//**   NO REFERENTIAL INTEGRITY CONSTRAINTS OR ARE THE ONES THAT MUST   00105600
//**   BE LOADED FIRST BECAUSE OF REFERENTIAL INTEGRITY CONSTRAINTS.    00105600
//**                                                                    00105600
//**   THE EXCEPTION TO THIS ARE THE ENGINEERING TABLES. TABLES         00105600
//**   ENG_UNIT_TYPE, ENGINEERING_UNIT, ENG_UNIT_CONV AND               00105600
//**   ENG_UNIT_STANDARD ARE IN THE ORDER THEY NEED TO BE LOADED DUE TO 00105600
//**   REFERENTIAL INTEGRITY CONSTRAINTS.                               00105600
//*****************************************************************00105600
//*                                                                    00102200
//STEP001 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00102300
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102400
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00102500
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102600
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//DSNTRACE DD  SYSOUT=*                                                00103100
//CPRSRECS DD  DSN=CIPROS.ANMETHOD.DATAOUT,                            00103200
//             DISP=SHR                                                00103300
//SYSERR   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104400
//SYSDISC  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104500
//SYSMAP   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104600
//SYSIN    DD  *                                                       00104700
 LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES                          00104900
      INTO TABLE PAOLOR2.ANALYSIS_METHOD                              00105000
           (ANLY_METH_NAME POSITION(1) VARCHAR,
            DESCRIPTION    POSITION(23) VARCHAR)
      ENFORCE CONSTRAINTS                                             00105600
//*
.
.
.
//*                                                                    00102200
//STEP007 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00102300
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102400
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00102500
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102600
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//DSNTRACE DD  SYSOUT=*                                                00103100
//CPRSRECS DD  DSN=CIPROS.EUNITTYP.DATAOUT,                            00103200
//             DISP=SHR                                                00103300
//SYSERR   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104400
//SYSDISC  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104500
//SYSMAP   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104600
//SYSIN    DD  *                                                       00104700
 LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES                          00104900
      INTO TABLE PAOLOR2.ENG_UNIT_TYPE                                00105000
           (ENG_UNT_TYPE POSITION(1) VARCHAR,
            DESCRIPTION POSITION(18) VARCHAR)
      ENFORCE CONSTRAINTS                                             00105600
//*                                                                    00102200
//STEP008 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00102300
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102400
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00102500
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102600
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//DSNTRACE DD  SYSOUT=*                                                00103100
//CPRSRECS DD  DSN=CIPROS.ENGUNIT.DATAOUT,                             00103200
//             DISP=SHR                                                00103300
//SYSERR   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104400
//SYSDISC  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104500
```

```
//SYSMAP   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)            00104600
//SYSIN    DD  *                                                       00104700
 LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES                           00104900
      INTO TABLE PAOLOR2.ENGINEERING_UNIT                              00105000
           (ENG_UNT POSITION(1) VARCHAR,
            ENG_UNT_TYPE POSITION(13) VARCHAR
             NULLIF (ENG_UNT_TYPE = '              '),
            DESCRIPTION POSITION(30) VARCHAR)
     ENFORCE CONSTRAINTS                                               00105600
//*                                                                    00102200
//STEP009 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00102300
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102400
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00102500
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102600
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//DSNTRACE DD  SYSOUT=*                                                00103100
//CPRSRECS DD  DSN=CIPROS.EUNITSTD.DATAOUT,                            00103200
//             DISP=SHR                                                00103300
//SYSERR   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104400
//SYSDISC  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104500
//SYSMAP   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104600
//SYSIN    DD  *                                                       00104700
 LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES                           00104900
      INTO TABLE PAOLOR2.ENG_UNIT_STANDARD                             00105000
           (INSTR_TYPE POSITION(1) VARCHAR,
            VALUE_TYPE POSITION(18) VARCHAR,
            STANDARD_TYPE POSITION(26) VARCHAR,
            ENG_UNT POSITION(34) VARCHAR)
     ENFORCE CONSTRAINTS                                               00105600
//*                                                                    00102200
//STEP010 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00102300
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102400
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00102500
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102600
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//DSNTRACE DD  SYSOUT=*                                                00103100
//CPRSRECS DD  DSN=CIPROS.EUNITCON.DATAOUT,                            00103200
//             DISP=SHR                                                00103300
//SYSERR   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104400
//SYSDISC  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104500
//SYSMAP   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104600
//SYSIN    DD  *                                                       00104700
 LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES                           00104900
      INTO TABLE PAOLOR2.ENG_UNIT_CONV                                 00105000
           (MULTIPLIER POSITION(1) FLOAT EXTERNAL(41),
            BIAS POSITION(42) FLOAT EXTERNAL(41),
            SRC_ENG_UNT POSITION(83) VARCHAR,
            TARG_ENG_UNT POSITION(95) VARCHAR)
     ENFORCE CONSTRAINTS                                               00105600
//*
.
.
.
//*                                                                    00102200
//STEP029 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                 00102300
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102400
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00102500
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102600
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//DSNTRACE DD  SYSOUT=*                                                00103100
//CPRSRECS DD  DSN=CIPROS.VALUETYP.DATAOUT,                            00103200
//             DISP=SHR                                                00103300
//SYSERR   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104400
//SYSDISC  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104500
//SYSMAP   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104600
//SYSIN    DD  *                                                       00104700
 LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES                           00104900
      INTO TABLE PAOLOR2.VALUE_TYPE                                    00105000
           (VALUE_TYPE POSITION(1) VARCHAR,
```

```
           DESCRIPTION POSITION(9) VARCHAR)
      ENFORCE CONSTRAINTS                                              00105600
//*                                                                     00139300
```

B.16 JCL for second job to LOAD CIPROS tables

```
//CPRSLOD2 JOB (999,POK),'PAOLOR2 ',CLASS=A,MSGCLASS=T,                00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                  00000002
/*JOBPARM L=999,SYSAFF=SC63                                           00000003
//JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR                       00002200
//        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR                            00002200
//*                                                                    00102200
///*                                                                    00102200
//STEP000 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                00102300
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102400
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00102500
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102600
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//DSNTRACE DD  SYSOUT=*                                                00103100
//CPRSRECS DD  DSN=CIPROS.MATERIAL.DATAOUT,                            00103200
//             DISP=SHR                                                00103300
//SYSERR   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104400
//SYSDISC  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104500
//SYSMAP   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104600
//SYSIN    DD  *                                                       00104700
 LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES                           00104900
      INTO TABLE PAOLOR2.MATERIAL                                      00105000
           (MTRL_NAME POSITION(1) VARCHAR,
            MTRL_NUM POSITION(18) INTEGER EXTERNAL(4),
            MTRL_TYPE POSITION(22) VARCHAR,
            DESCRIPTION POSITION(39) VARCHAR,
            GROUP_FLAG POSITION(81) CHAR)
      ENFORCE CONSTRAINTS
.
.
.
//*                                                                    00102200
//STEP011 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)                00102300
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102400
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR                               00102500
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102600
//SORTWK01 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK02 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//SORTWK03 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102700
//SORTWK04 DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00102800
//DSNTRACE DD  SYSOUT=*                                                00103100
//CPRSRECS DD  DSN=CIPROS.TAG.DATAOUT,                                 00103200
//             DISP=SHR                                                00103300
//SYSERR   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104400
//SYSDISC  DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104500
//SYSMAP   DD  UNIT=SYSALLDA,SPACE=(4000,(10,5),RLSE,,ROUND)           00104600
//SYSIN    DD  *                                                       00104700
 LOAD DATA INDDN(CPRSRECS) LOG NO RESUME YES                           00104900
      INTO TABLE PAOLOR2.TAG
            (TAG_NAME POSITION(1) VARCHAR,
             INSTR_NAME POSITION(23) VARCHAR,
             READING_TYPE POSITION(45) VARCHAR,
             VALUE_TYPE POSITION(53) VARCHAR,
             TIME_RES POSITION(61) VARCHAR,
             STATUS POSITION(73) VARCHAR,
             HI_VALUE POSITION(85) FLOAT EXTERNAL(41),
             LO_VALUE POSITION(126) FLOAT EXTERNAL(41),
             DATA_TYPE POSITION(167) VARCHAR,
             ENG_UNT POSITION(175) VARCHAR,
             DESCRIPTION POSITION(187) VARCHAR,
             CREATION_DATE POSITION(229) TIMESTAMP EXTERNAL(26)
              NULLIF (CREATION_DATE = '                          '),
             DISCONN_DATE POSITION(255) TIMESTAMP EXTERNAL(26)
              NULLIF (DISCONN_DATE = '                         '))
      ENFORCE CONSTRAINTS                                              00105600
//*
```

B.17 JCL for binding with use of packages

```
//PAOLOC JOB (999,POK),'PAOLOR2',CLASS=A,MSGCLASS=T,                    00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//JOBLIB  DD  DSN=CEE.SCEERUN,DISP=SHR                                  00260000
//        DD  DSN=DSN610.QPP.SDSNLOAD,DISP=SHR                          00270000
//STEP001  EXEC PGM=IKJEFT01,DYNAMNBR=20                               00640000
//SYSTSPRT  DD  SYSOUT=*                                               00660000
//SYSPRINT  DD  SYSOUT=*                                               00670000
//CEEDUMP   DD  SYSOUT=*                                               00680000
//SYSUDUMP  DD  SYSOUT=*                                               00690000
//SYSOUT    DD  SYSOUT=*                                               00700000
//REPORT    DD  SYSOUT=*                                               00710000
//SYSTSIN  DD *
   DSN SYSTEM(DB2X) RETRY (1) TEST (0)
   BIND PLAN(CIPROS1)  -
      PKLIST(CIPROS1.*) -
      ACTION(REPLACE)   -
      ACQUIRE(USE)      -
      RELEASE(COMMIT)   -
      ISOLATION(CS)     -
      VALIDATE(BIND)    -
      RETAIN            -
      EXPLAIN(YES)
   END
/*
```

B.18 JCL stream including REXX program

```
//ITSIS20  JOB 9101,'PBRUNI            ',NOTIFY=ITSIS2,
//         MSGLEVEL=(1,1),MSGCLASS=8,CLASS=K,REGION=6M
//*====================================================================
//*--- TEST XXLOAD - Prepare variable length data for DB2 LOAD -------
//*====================================================================
//COPY1 EXEC PGM=IEBGENER
//SYSUT1 DD *,DLM=$$
/* REXX - Prepare variable length data for DB2 LOAD -----------------*/
/*                                                                   */
/* Description: Prepare input for DB2 LOAD adding length field to    */
/*         variable length data                                      */
/*                                                                   */
/* Input:  CTLIN    - Description of Input data                      */
/*         DATAIN   - Input data to be converted                     */
/* Output: SYSPRINT - Messages                                       */
/*         DATAOUT  - Output data for DB2 LOAD                       */
/*         CTLOUT   - Description of Output data ... da fare         */
/*                                                                   */
/*------------------------------------------------------- C.Venturini */

  ExitRC = 0                  /* Init Return Code = OK       */
  IoMeC = Sysvar('SYSICMD')   /* Nome di questa Exec         */
  EnvXe = Sysvar('SYSENV')    /* Environment: FORE o BACK    */

  If EnvXe = "FORE" Then
    Do
      Say "This eXec must be run in BATCH only"
      EXIT 16
    End

  /* Try writing to DDName DATAOUT ------------------------------*/
  "EXECIO * DISKW DATAOUT (FINIS STEM BegLine.)"
  If RC > 4 Then
    Do
      Say "Error writing DDname DATAOUT"
      Say "Return Code from EXECIO:" RC
      AllocError = 1
    End

  /* Try writing to DDName CTLOUT -------------------------------*/
  "EXECIO * DISKW CTLOUT  (FINIS STEM BegLine.)"
  If RC > 4 Then
    Do
      Say "Error writing DDname CTLOUT"
```

```
    Say "Return Code from EXECIO:" RC
    AllocError = 1
  End

/* Read data from DDName CTLIN --------------------------------*/
"EXECIO * DISKR CTLIN   (FINIS STEM CtIn.)"
If RC > 4 Then
  Do
    Say "Error reading DDname CTLIN"
    Say "Return Code da EXECIO:" RC
    AllocError = 1
  End

/* Read data from DDName DATAIN -------------------------------*/
"EXECIO * DISKR DATAIN  (FINIS STEM DatIn.)"
If RC > 4 Then
  Do
    Say "Error reading DDname DATAIN"
    Say "Return Code da EXECIO:" RC
    AllocError = 1
  End

If AllocError = 1 Then
  Do
    Say "Error accessing files ... quitting ..."
    EXIT 16
  End

oCtl. = "" ; oCtlIx = 0 ; oCtl.0 = 0     /* Clear output array */

oCtlIx = oCtlIx+1
oCtl.oCtlIx = "* Control file describing converted data (DATAOUT)"
oCtlIx = oCtlIx+1
oCtl.oCtlIx = "* (same format as CTLIN)"

/* Clear arrays to be loaded from CTLIN -----------------------*/
cBeg. = "" ; cLen. = "" ; cTyp. = "" ; j = 0
oBeg = 1                              /* init output position */

/* Load CTLIN into arrays doing some validity check -----------*/
Do i = 1 to CtIn.0
  If Left(CtIn.i,1) = "*" Then Iterate    /* Skip comment line */
  j = j+1
  Parse VAR CtIn.i w1 w2 w3 .
  If ((w3 <> "F") & (w3 <> "V")) Then
    Do
      Say "Wrong data in CTLIN ..."
      Say "Type must be 'F' or 'V' ... "
      Exit 8
    End
  cBeg.j = Strip(w1)
  cLen.j = Strip(w2)
  cTyp.j = Strip(w3)
  oLen = cLen.j
  oTyp = cTyp.j
  oCtlIx = oCtlIx+1
  oCtl.oCtlIx = oBeg oLen oTyp
  If w3 = "V" Then
    Do
      oBeg = oBeg+oLen+2
    End
  Else
    Do
      oBeg = oBeg+oLen
    End
End
NumFields = j

oCtl.0 = oCtlIx
"EXECIO * DISKW CTLOUT  (FINIS STEM oCtl.)"
If RC > 4 Then
  Do
    Say "Error writing DDname CTLOUT"
    Say "Return Code from EXECIO:" RC
    EXIT 8
  End

oData. = "" ; oDataIx = 0 ; oData.0 = 0  /* Clear output array */
```

```
  /* Convert DATAIN into DATAOUT --------------------------------*/
  Do i = 1 to DatIn.0
    wOut = ""                                /* clear output row */
    Do j = 1 to NumFields
      wField = Substr(DatIn.i,cBeg.j,cLen.j)
      If cTyp.j = "V" Then
        Do
          wF = Strip(wField)                 /* remove blanks  */
          wL = Length(wF)                    /* get length     */
          wLX1 = Right('000'D2X(wL),4)       /* convert to hex */
          wLX2 = X2C(wLX1)                   /* ... binary     */
          wOut = wOut³³wLX2³³wField    /* append length & data */
        End
      Else
        Do
          wOut = wOut³³wField        /* append data to out row */
        End
    End
    oDataIx = oDataIx+1
    oData.oDataIx = wOut
  End

  oData.0 = oDataIx
  "EXECIO * DISKW DATAOUT (FINIS STEM oData.)"
  If RC > 4 Then
    Do
      Say "Error writing DDname DATAOUT"
      Say "Return Code from EXECIO:" RC
      EXIT 8
    End

  EXIT 0

$$
//SYSUT2 DD DSN=&&TEMP(XXLOAD),DISP=(NEW,PASS),
//  DCB=SYS1.TSO.CLIST,
//  SPACE=(TRK,(5,5,5)),UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//*
//CLEANUP  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE ITSIS2.PROVA.PBRUNI.DATAOUT
  DELETE ITSIS2.PROVA.PBRUNI.DATAOUT  NOSCRATCH
  DELETE ITSIS2.PROVA.PBRUNI.CTLOUT
  DELETE ITSIS2.PROVA.PBRUNI.CTLOUT   NOSCRATCH
  SET MAXCC=0
//*
//CALLEXEC EXEC PGM=IKJEFT01,DYNAMNBR=99,PARM=XXLOAD
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CTLIN    DD *
* Control file describing input file to be converted (DATAIN)
* Each line of this control file contains 3 fields
* separated by blank(s):
* 1) Position of input field
* 2) Length   of input field
* 3) Type     of input field  (V=Variable_length F=Fixed_length)
*    comments are allowed after 3rd field
1   10   V    1st input field from position  1 is Variable length
15  9    F    2nd  ...    ..   ..    ...    15 is Fixed    length
30  8    V    3rd  ...    ..   ..    ...    30 is Variable length
45  8    F    4th  ...    ..   ..    ...    45 is Fixed    length
//DATAIN   DD *
VAR1          FIX1          VAR2          FIX2    UNUSED DATA
1234567890    123456789     12345678      12345678***********
ABCDEFGHIJ    KLMNOPQRS     TUVWXYZA      HGFEDCBA***********
Sample        datain        for           testing ***********
may           contain       hex
±<*%ö**********
//*-+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
//DATAOUT  DD DSN=ITSIS2.PROVA.PBRUNI.DATAOUT,
//         DCB=(LRECL=255,BLKSIZE=0,RECFM=VB),
//         UNIT=SYSDA,SPACE=(TRK,(15,15)),DISP=(NEW,CATLG)
//CTLOUT   DD DSN=ITSIS2.PROVA.PBRUNI.CTLOUT,
```

```
//         DCB=(LRECL=80,BLKSIZE=0,RECFM=FB),
//         UNIT=SYSDA,SPACE=(TRK,(1,1)),DISP=(NEW,CATLG)
//SYSPROC  DD DISP=SHR,DSN=SYS1.TSO.CLIST
//SYSEXEC  DD DISP=SHR,DSN=&&TEMP
//SYSTSIN  DD DUMMY
```

B.19 JCL for the conversion of data using REXX program

```
//CPRSALLL JOB (999,POK),'PAOLOR2',CLASS=A,MSGCLASS=T,                  00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                             00000003
//* ********************************************************************
//* THIS JOB CREATES THE LOAD FILES FOR ALL THE PROJECT TABLES.
//* THE REXX PROGRAM ALLOWS FOR COMMENTS BEYOND THE POSITIONAL INPUT
//* AS EXPLAINED BELOW.  IN THIS AREA WE HAVE CHOSEN TO PLACE THE
//* PARTIAL DDL USED TO DESCRIBE THE TABLE TO WHICH THE DATA IS RELATED.
//* THESE COMMENTS CAN BE USED TO CHECK THE ACCURACY OF THE POSITIONAL
//* AND DATA TYPE VALUES FOR INPUT TO THE REXX PROGRAM.
//* THE FIRST STEP LOADS THE REXX PROGRAM AND THE SUBSEQUENT STEPS ARE
//* ARRANGED IN PAIRS. THE FIRST OF THE PAIR DELETES THE FILES THE NEXT
//* WILL CREATE FOR THE ADJUSTED DATA. THERE IS ONE PAIR FOR EACH TABLE.
//*
//* NOTE: LENGTHS OF FIELDS WITH A DDL COMMENT INDICATING AN INTEGER
//*       VALUE WILL VARY. THIS WILL DEPEND UPON THE ORACLE "NUMBER"
//*       DEFINITION THAT WAS USED TO GENERATE THE INPUT DATA AND THE
//*       INTEGER DDL DEFINITION. IF THE ORACLE NUMBER WAS 3 THE FIELD
//*       WILL BE 3 BYTES LONG.  IF IT WAS 4 THE INPUT WILL CONTAIN 4,
//*       AND SO ON.  THE POSITION USED TO LOCATE THE DATA IN THE FILE
//*       WILL, OF COURSE, VARY AS WELL.  SO, FIELDS INDICATED TO BE
//*       INTEGER BY THE DDL WILL HAVETO BE EXAMINED AND THE
//*       POSITIONAL LOCATOR FOR THE REXX PROGRAM ADJUSTED TO MATCH.
//*       DO NOT ASSUME ALL INTEGER FIELDS WILL BE THE SAME SIZE.
//*
//*       NEXT TO THE DDL INTEGER DEFINITION WE HAVE ADDED THE SIZE OF
//*       THE FIELD BASED ON THE INPUT FILE GENERATED FROM THE ORACLE
//*       DATA.
//*
//*       THIS HAS DOWNSTREAM CONSEQUENCES FOR THE DB2 UTILITY LOAD FOR
//*       WHICH THIS JOB PREPARES THE DATA.  THE POSITION DESCRIPTION OF
//*       INTEGER DATA FOR ANY PARTICULAR INPUT FILE WILL HAVE TO
//*       REFLECT THE ABOVE, AS WELL.
//*
//* REMEMBER: THE FILE BLOCK AND RECORD SIZES FOR OUTPUT ARE VARIABLE
//*           BLOCKED AT 1.  THEREFORE, ADD 4 TO THE TOTAL FOR THE
//*           RECORD TO ACCOUNT FOR THE BLOCK SIZE AND RECORD SIZE BYTES
//*           OR YOUR RECORDS WILL LOOSE THEIR LAST 4 BYTES OF DATA.
//*           THESE LENGTH BYTES AT THE BEGINING OF THE RECORDS ARE NOT
//*           DISPLAYED.  THE OPERATING SYSTEM WILL ASSUME YOU HAVE
//*           ACCOUNTED FOR THESE BYTES IN YOUR ALLOCATION.
//*           IN THIS JOB, ONLY FILE DATAOUT IS AFFECTED BY THIS
//*           CONSIDERATION.
//*
//* ********************************************************************
//*====================================================================
//*--- TEST XXLOAD - PREPARE VARIABLE LENGTH DATA FOR DB2 LOAD -------
//*====================================================================
//LOAD001 EXEC PGM=IEBGENER
//SYSUT1 DD DSN=PAOLOR2.C.SOURCE(REXXPGM),DISP=SHR
//SYSUT2 DD DSN=&&TEMP(XXLOAD),DISP=(NEW,PASS),
//          DCB=(LRECL=80,BLKSIZE=6160,RECFM=FB),
//          SPACE=(TRK,(5,5,5)),UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//*
//STEP001  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE CIPROS.ANMETHOD.DATAOUT
  DELETE CIPROS.ANMETHOD.DATAOUT NOSCRATCH
  DELETE CIPROS.ANMETHOD.CTLOUT
  DELETE CIPROS.ANMETHOD.CTLOUT  NOSCRATCH
  SET MAXCC=0
//*
//STEP002 EXEC PGM=IKJEFT01,DYNAMNBR=99,PARM=XXLOAD
//SYSTSPRT DD SYSOUT=*
```

```
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*** CONTROL FILE DESCRIBING INPUT FILE TO BE CONVERTED (DATAIN)
//** EACH LINE OF THIS CONTROL FILE CONTAINS 3 FIELDS
//** SEPARATED BY BLANK(S):
//** 1) POSITION OF INPUT FIELD
//** 2) LENGTH   OF INPUT FIELD
//** 3) TYPE     OF INPUT FIELD  (V=VARIABLE_LENGTH F=FIXED_LENGTH)
//**    COMMENTS ARE ALLOWED AFTER 3RD FIELD
//*1   10   V    1ST INPUT FIELD FROM POSITION  1 IS VARIABLE LENGTH
//*15  9    F    2ND  ...    ..   ..    ...    15 IS FIXED    LENGTH
//*30  8    V    3RD  ...    ..   ..    ...    30 IS VARIABLE LENGTH
//*45  8    F    4TH  ...    ..   ..    ...    45 IS FIXED    LENGTH
//*-+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
//CTLIN    DD *
1   20   V         ANALYSIS_METHOD (ANLY_METH_NAME VARCHAR(20)
21  40   V         DESCRIPTION VARCHAR(40)
//DATAIN   DD DSN=PAOLOR3.CIPROS.DATA(ANMETHOD),DISP=SHR
//DATAOUT  DD DSN=CIPROS.ANMETHOD.DATAOUT,
//         DCB=(LRECL=68,BLKSIZE=0,RECFM=VB),
//         UNIT=3390,VOL=SER=SBOX09,
//         SPACE=(TRK,(15,15)),DISP=(NEW,CATLG,CATLG)
//CTLOUT   DD DSN=CIPROS.ANMETHOD.CTLOUT,
//         DCB=(LRECL=80,BLKSIZE=0,RECFM=FB),
//         UNIT=SYSDA,SPACE=(TRK,(1,1)),DISP=(NEW,CATLG,CATLG)
//SYSPROC  DD DISP=SHR,DSN=SYS1.TSOCLIST
//SYSEXEC  DD DISP=(OLD,PASS),DSN=&&TEMP
//SYSTSIN  DD DUMMY
//*
.
.
.
//*
//STEP127  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE CIPROS.VALUETYP.DATAOUT
  DELETE CIPROS.VALUETYP.DATAOUT NOSCRATCH
  DELETE CIPROS.VALUETYP.CTLOUT
  DELETE CIPROS.VALUETYP.CTLOUT  NOSCRATCH
  SET MAXCC=0
//*
//STEP128 EXEC PGM=IKJEFT01,DYNAMNBR=99,PARM=XXLOAD
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*** CONTROL FILE DESCRIBING INPUT FILE TO BE CONVERTED (DATAIN)
//** EACH LINE OF THIS CONTROL FILE CONTAINS 3 FIELDS
//** SEPARATED BY BLANK(S):
//** 1) POSITION OF INPUT FIELD
//** 2) LENGTH   OF INPUT FIELD
//** 3) TYPE     OF INPUT FIELD  (V=VARIABLE_LENGTH F=FIXED_LENGTH)
//**    COMMENTS ARE ALLOWED AFTER 3RD FIELD
//*1   10   V    1ST INPUT FIELD FROM POSITION  1 IS VARIABLE LENGTH
//*15  9    F    2ND  ...    ..   ..    ...    15 IS FIXED    LENGTH
//*30  8    V    3RD  ...    ..   ..    ...    30 IS VARIABLE LENGTH
//*45  8    F    4TH  ...    ..   ..    ...    45 IS FIXED    LENGTH
//*-+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
//CTLIN    DD *
1   6    V         VALUE_TYPE (VALUE_TYPE VARCHAR(6)
7   40   V         DESCRIPTION VARCHAR(40))
//DATAIN   DD DSN=PAOLOR3.CIPROS.DATA(VALUETYP),DISP=SHR
//DATAOUT  DD DSN=CIPROS.VALUETYP.DATAOUT,
//         DCB=(LRECL=54,BLKSIZE=0,RECFM=VB),
//         UNIT=3390,VOL=SER=SBOX09,
//         SPACE=(TRK,(15,15)),DISP=(NEW,CATLG,CATLG)
//CTLOUT   DD DSN=CIPROS.VALUETYP.CTLOUT,
//         DCB=(LRECL=80,BLKSIZE=0,RECFM=FB),
//         UNIT=SYSDA,SPACE=(TRK,(1,1)),DISP=(NEW,CATLG,CATLG)
//SYSPROC  DD DISP=SHR,DSN=SYS1.TSOCLIST
//SYSEXEC  DD DISP=(OLD,PASS),DSN=&&TEMP
//SYSTSIN  DD DUMMY
```

B.20 DSNTIJUZ - DB2 installation job stream

```
//DB2ZE JOB (999,POK),'DB2V610Z',CLASS=A,MSGCLASS=T,                    00000001
// NOTIFY=HAIMO,TIME=1440,REGION=0M                                     00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//*******************************************************************/00010000
//* JOB NAME = DSNTIJUZ                                              */00020000
//*                                                                  */00030000
//* DESCRIPTIVE NAME = INSTALLATION JOB STREAM                       */00040000
//*                                                                  */00050000
//*    LICENSED MATERIALS - PROPERTY OF IBM                          */00060000
//*    5645-DB2                                                      */00070000
//*    (C) COPYRIGHT 1982, 1998 IBM CORP.  ALL RIGHTS RESERVED.      */00080000
//*                                                                  */00090000
//*    STATUS = VERSION 6                                            */00100000
//*                                                                  */00110000
//* FUNCTION = DSNZPARM AND DSNHDECP UPDATES                         */00120000
//*                                                                  */00130000
//* PSEUDOCODE =                                                     */00140000
//*   DSNTIZA  STEP  ASSEMBLE DSN6.... MACROS, CREATE DSNZPARM       */00150000
//*   DSNTIZL  STEP  LINK EDIT DSNZPARM                              */00160000
//*   DSNTLOG  STEP  UPDATE PASSWORDS                                */00170000
//*   DSNTIZP  STEP  ASSEMBLE DSNHDECP DATA-ONLY LOAD MODULE         */00180000
//*   DSNTIZQ  STEP  LINK EDIT DSNHDECP LOAD MODULE                  */00190000
//*   DSNTIMQ  STEP  SMP/E PROCESSING FOR DSNHDECP                   */00200000
//*                                                                  */00210000
//* NOTES = STEP DSNTIMQ MUST BE CUSTOMIZED FOR SMP.  SEE THE NOTES  */00220000
//*         NOTES PRECEDING STEP DSNTIMQ BEFORE RUNNING THIS JOB.    */00230000
//*                                                                  */00240000
//*******************************************************************/00250000
//*                                                                    00260000
//DSNTIZA EXEC PGM=ASMA90,PARM='OBJECT,NODECK'                         00270000
//SYSLIB   DD  DISP=SHR,                                               00280000
//         DSN=DSN610.QPP.SDSNMACS                                     00290000
//         DD  DISP=SHR,                                               00300000
//         DSN=SYS1.MACLIB                                             00310000
//SYSLIN   DD  DSN=&&LOADSET(DSNTILMZ),DISP=(NEW,PASS),                00320000
//             UNIT=SYSALLDA,                                          00330000
//             SPACE=(800,(50,50,2)),DCB=(BLKSIZE=800)                 00340000
//SYSPRINT DD  SYSOUT=*                                                00350000
//SYSUDUMP DD  SYSOUT=*                                                00360000
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(800,(50,50),,,ROUND)               00370000
//SYSUT2   DD  UNIT=SYSALLDA,SPACE=(800,(50,50),,,ROUND)               00380000
//SYSUT3   DD  UNIT=SYSALLDA,SPACE=(800,(50,50),,,ROUND)               00390000
//SYSIN    DD  *                                                       00400000
    DSN6ENV    MVS=XA                                                 00410000
    DSN6SPRM   RESTART,                                              X00410001
               ALL,                                                  X00410002
               ABEXP=YES,                                            X00410003
               ABIND=YES,                                            X00410004
               AUTH=YES,                                             X00410005
               AUTHCACH=1024,                                        X00410006
               BINDNV=BINDADD,                                       X00410007
               BMPTOUT=4,                                            X00410008
               CACHEDYN=NO,                                          X00410009
               CACHEPAC=32768,                                       X00410010
               CACHERAC=32768,                                       X00410011
               CATALOG=DB2V610Z,                                     X00410012
               CDSSRDEF=1,                                           X00410013
               CHGDC=NO,                                             X00410014
               CONTSTOR=NO,                                          X00410015
               DECDIV3=NO,                                           X00410016
               DEFLTID=IBMUSER,                                      X00410017
               DESCSTAT=NO,                                          X00410018
               DLITOUT=6,                                            X00410019
               DSMAX=3000,                                           X00410020
               EDMPOOL=14812,                                        X00410021
               EDMDSPAC=0,                                           X00410022
               EDPROP=NO,                                            X00410023
               HOPAUTH=BOTH,                                         X00410024
               IRLMAUT=YES,                                          X00410025
               IRLMPRC=IRLZPROC,                                     X00410026
               IRLMSID=IRLZ,                                         X00410027
               IRLMRWT=60,                                           X00410028
               IRLMSWT=300,                                          X00410029
               LEMAX=20,                                             X00410030
               MAXRBLK=4000,                                         X00410031
```

```
                MAXKEEPD=5000,                                  X00410032
                NUMLKTS=1000,                                   X00410033
                NUMLKUS=10000,                                  X00410034
                OPTHINTS=NO,                                    X00410035
                RECALL=YES,                                     X00410036
                RECALLD=120,                                    X00410037
                RELCURHL=YES,                                   X00410038
                RETLWAIT=0,                                     X00410039
                RETVLCFK=NO,                                    X00410040
                RGFCOLID=DSNRGCOL,                              X00410041
                RGFDBNAM=DSNRGFDB,                              X00410042
                RGFDEDPL=NO,                                    X00410043
                RGFDEFLT=ACCEPT,                                X00410044
                RGFESCP=,                                       X00410045
                RGFFULLQ=YES,                                   X00410046
                RGFINSTL=NO,                                    X00410047
                RGFNMORT=DSN_REGISTER_OBJT,                     X00410048
                RGFNMPRT=DSN_REGISTER_APPL,                     X00410049
                RRULOCK=NO,                                     X00410050
                SEQCACH=BYPASS,                                 X00410051
                SEQPRES=NO,                                     X00410052
                SITETYP=LOCALSITE,                              X00410053
                SRTPOOL=1000,                                   X00410054
                SYSADM=HAIMO,                                   X00410055
                SYSADM2=PAOLO,                                  X00410056
                SYSOPR1=SYSOPR,                                 X00410057
                SYSOPR2=SYSOPR,                                 X00410058
                TRKRSITE=NO,                                    X00410059
                UTIMOUT=6                                        00410060
DSN6ARVP        ALCUNIT=BLK,                                    X00410061
                ARCWRTC=(1,3,4),                                X00410062
                ARCWTOR=YES,                                    X00410063
                ARCPFX1=DB2V610Z.ARCHLOG1,                      X00410064
                ARCPFX2=DB2V610Z.ARCHLOG2,                      X00410065
                ARCRETN=9999,                                   X00410066
                BLKSIZE=28672,                                  X00410067
                CATALOG=YES,                                    X00410068
                COMPACT=NO,                                     X00410069
                PRIQTY=1234,                                    X00410070
                PROTECT=NO,                                     X00410071
                QUIESCE=5,                                      X00410072
                SECQTY=154,                                     X00410073
                TSTAMP=NO,                                      X00410074
                UNIT=SYSALLDA,                                  X00410075
                UNIT2=SYSALLDA                                   00410076
DSN6LOGP        DEALLCT=(0),                                    X00410077
                INBUFF=60,                                      X00410078
                MAXARCH=1000,                                   X00410079
                MAXRTU=2,                                       X00410080
                OUTBUFF=4000,                                   X00410081
                TWOACTV=YES,                                    X00410082
                TWOARCH=YES,                                    X00410083
                WRTHRSH=20,                                     X00410084
                ARC2FRST=NO                                      00410085
DSN6SYSP        AUDITST=NO,                                     X00410086
                BACKODUR=5,                                     X00410087
                CONDBAT=64,                                     X00410088
                CTHREAD=70,                                     X00410089
                DBPROTCL=DRDA,                                  X00410090
                DLDFREQ=5,                                      X00410091
                DSSTIME=5,                                      X00410092
                EXTRAREQ=100,                                   X00410093
                EXTRASRV=100,                                   X00410094
                IDBACK=20,                                      X00410095
                IDFORE=40,                                      X00410096
                IDXBPOOL=BP0,                                   X00410097
                LBACKOUT=AUTO,                                  X00410098
                LOBVALA=2048,                                   X00410099
                LOBVALS=2048,                                   X00410100
                LOGAPSTG=0,                                     X00410101
                LOGLOAD=50000,                                  X00410102
                MAXDBAT=64,                                     X00410103
                MON=NO,                                         X00410104
                MONSIZE=8192,                                   X00410105
                PCLOSEN=5,                                      X00410106
                PCLOSET=10,                                     X00410107
                RLF=NO,                                         X00410108
                RLFTBL=01,                                      X00410109
```

```
                RLFERR=NOLIMIT,                                          X00410110
                RLFAUTH=SYSIBM,                                          X00410111
                ROUTCDE=(1),                                             X00410112
                EXTSEC=NO,                                               X00410113
                SMFACCT=(1),                                             X00410114
                SMFSTAT=YES,                                             X00410115
                STATIME=30,                                              X00410116
                STORMXAB=0,                                              X00410117
                STORPROC=DB2ZSPAS,                                       X00410118
                STORTIME=180,                                            X00410119
                TBSBPOOL=BP0,                                            X00410120
                TRACSTR=NO,                                              X00410121
                TRACTBL=16,                                              X00410122
                URCHKTH=0,                                               X00410123
                WLMENV=                                                   00410124
    DSN6FAC     DDF=NO,                                                  X00410125
                CMTSTAT=ACTIVE,                                          X00410126
                IDTHTOIN=0,                                              X00410127
                RESYNC=2,                                                X00410128
                RLFERRD=NOLIMIT,                                         X00410129
                TCPALVER=NO,                                             X00410130
                MAXTYPE1=0                                                00410131
    DSN6GRP     DSHARE=NO,                                               X00410132
                GRPNAME=DSNCAT,                                          X00410133
                MEMBNAME=DSN1,                                           X00410134
                COORDNTR=NO,                                             X00410135
                ASSIST=NO                                                 00410136
    END                                                                   01780000
//******************************************************************** 01790000
//* LINK EDIT THE NEW DSNZPARM MEMBER.  PUT LOAD MODULE IN SDSNEXIT.  * 01800000
//******************************************************************** 01810000
//DSNTIZL EXEC PGM=IEWL,PARM='LIST,XREF,LET,RENT',                       01820000
//             COND=(4,LT)                                               01830000
//ADSNLOAD DD  DISP=SHR,                                                 01840000
//         DSN=DSN610.QPP.SDSNLOAD                                       01850000
//         DD  DISP=SHR,                                                 01860000
//         DSN=DSN610.QPP.ADSNLOAD                                       01870000
//SYSPUNCH DD  DSN=&&LOADSET(DSNTILMZ),DISP=(OLD,DELETE)                 01880000
//SYSLMOD  DD  DISP=SHR,                                                 01890000
//         DSN=DSN610.QPP.SDSNEXIT                                       01900000
//SYSPRINT DD  SYSOUT=*                                                  01910000
//SYSUDUMP DD  SYSOUT=*                                                  01920000
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(1024,(50,50))                        01930000
//SYSLIN   DD  *                                                         01940000
   INCLUDE SYSPUNCH(DSNTILMZ)                                            01950000
   INCLUDE ADSNLOAD(DSNZPARM)                                            01960000
   ORDER DSNAA                                                           01970000
   INCLUDE ADSNLOAD(DSNAA)                                               01980000
   INCLUDE ADSNLOAD(DSNFSYSP)                                            01990000
   INCLUDE ADSNLOAD(DSNJARVP)                                            02000000
   INCLUDE ADSNLOAD(DSNJLOGP)                                            02010000
   INCLUDE ADSNLOAD(DSNTSPRM)                                            02020000
   INCLUDE ADSNLOAD(DSNVDIR1)                                            02030000
   INCLUDE ADSNLOAD(DSNZMSTR)                                            02040000
   INCLUDE ADSNLOAD(DSN3DIR1)                                            02050000
   INCLUDE ADSNLOAD(DSN7GRP)                                             02060000
   ENTRY   DSNZMSTR                                                      02070000
   NAME    DSNZDB2Z(R)                                                   02080000
//*                                                                      02090000
//*        CHANGE LOG INVENTORY:                                         02100000
//*        UPDATE BSDS                                                   02110000
//*                                                                      02120000
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)                                  02130000
//STEPLIB  DD  DISP=SHR,DSN=DSN610.QPP.SDSNLOAD                          02140000
//SYSUT1   DD  DISP=OLD,DSN=DB2V610Z.BSDS01                              02150000
//SYSUT2   DD  DISP=OLD,DSN=DB2V610Z.BSDS02                              02160000
//SYSPRINT DD  SYSOUT=*                                                  02170000
//SYSUDUMP DD  SYSOUT=*                                                  02180000
//SYSIN    DD  *                                                         02190000
 DDF    LOCATION=DB2Z,LUNAME=SCPDB2Z,                                   02190001
        NOPASSWD,RESPORT=33341,PORT=33340                               02190002
//*                                                                      02220000
//******************************************************************** 02230000
//*   ASSEMBLE AND LINK EDIT DATA-ONLY LOAD MODULE DSNHDECP.             02240000
//*   THE FOLLOWING STEPS ARE NEEDED ONLY IF THE                         02250000
//*   VALUES ARE CHANGED FROM THOSE WHICH ARE SHIPPED.                   02260000
//******************************************************************** 02270000
//DSNTIZP EXEC PGM=ASMA90,PARM='OBJECT,NODECK',COND=(4,LT)               02280000
```

```
//SYSLIB   DD  DISP=SHR,                                                02290000
//         DSN=DSN610.QPP.SDSNMACS                                      02300000
//SYSLIN   DD  DSN=&&LOADSET(DSNHDECA),DISP=(NEW,PASS),UNIT=SYSALLDA,   02310000
//             SPACE=(80,(50,50,2)),DCB=(BLKSIZE=80)                    02320000
//SYSPRINT DD  SYSOUT=*                                                02330000
//SYSUDUMP DD  SYSOUT=*                                                02340000
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(800,(50,50),,,ROUND)               02350000
//SYSUT2   DD  UNIT=SYSALLDA,SPACE=(800,(50,50),,,ROUND)               02360000
//SYSUT3   DD  UNIT=SYSALLDA,SPACE=(800,(50,50),,,ROUND)               02370000
//SYSIN    DD  *                                                       02380000
   DSNHDECM    CHARSET=ALPHANUM,                                      X02380001
               ASCCSID=0,                                             X02380002
               AMCCSID=65534,                                         X02380003
               AGCCSID=65534,                                         X02380004
               SCCSID=0,                                              X02380005
               MCCSID=65534,                                          X02380006
               GCCSID=65534,                                          X02380007
               ENSCHEME=EBCDIC,                                       X02380008
               DATE=ISO,                                              X02380009
               DATELEN=0,                                             X02380010
               DECARTH=DEC15,                                         X02380011
               DECIMAL=PERIOD,                                        X02380012
               DEFLANG=IBMCOB,                                        X02380013
               DELIM=DEFAULT,                                         X02380014
               MIXED=NO,                                              X02380015
               SQLDELI=DEFAULT,                                       X02380016
               DSQLDELI=APOST,                                        X02380017
               SSID=DB2Z,                                             X02380018
               STDSQL=NO,                                             X02380019
               TIME=ISO,                                              X02380020
               TIMELEN=0,                                             X02380021
               DYNRULS=YES,                                           X02380022
               LOCALE=,                                               X02380023
               COMPAT=OFF                                              02380024
    END                                                                02630000
//*                                                                    02640000
//******************************************************************** 02650000
//*   LINK EDIT DSNHDECP.                                            * 02660000
//*   DSNHDECP IS A DATA-ONLY LOAD MODULE CONTAINING DEFAULT VALUES   * 02670000
//*   REQUIRED BY DB2 AND APPLICATION PROGRAMS.                      * 02680000
//*   THIS STEP IS CREATED ONLY WHEN THE DEFAULTS SUPPLIED IN         * 02690000
//*   DSNHDECP ARE NOT SUITABLE.                                     * 02700000
//******************************************************************** 02710000
//DSNTIZQ EXEC PGM=IEWL,PARM='LIST,XREF,LET,RENT',                     02720000
//             COND=(4,LT)                                             02730000
//ADSNLOAD DD  DISP=SHR,                                               02740000
//         DSN=DSN610.QPP.SDSNEXIT                                     02750000
//         DD  DISP=SHR,                                               02760000
//         DSN=DSN610.QPP.ADSNLOAD                                     02770000
//SYSPUNCH DD  DSN=&&LOADSET(DSNHDECA),DISP=(OLD,DELETE)               02780000
//SYSLMOD  DD  DISP=SHR,                                               02790000
//         DSN=DSN610.QPP.SDSNEXIT                                     02800000
//SYSPRINT DD  SYSOUT=*                                                02810000
//SYSUDUMP DD  SYSOUT=*                                                02820000
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(1024,(50,50))                      02830000
//SYSLIN   DD  *                                                       02840000
   INCLUDE SYSPUNCH(DSNHDECA)                                        02850000
   ORDER   DSNAA                                                     02860000
   INCLUDE ADSNLOAD(DSNAA)                                           02870000
   INCLUDE ADSNLOAD(DSNARIB)                                         02880000
   INCLUDE ADSNLOAD(DSNHDECP)                                        02890000
   ENTRY   DSNHDECP                                                  02900000
   MODE    AMODE(24),RMODE(24)                                       02910000
   NAME    DSNHDECP(R)                                               02920000
//*                                                                    02930000
//******************************************************************** 02940000
//* DO SMP/E PROCESSING TO TRACK DSNHDECP CHANGES.                   * 02950000
//* THIS STEP IS ONLY USED WHEN THE DEFAULT DSNHDECP IS NOT SUITABLE. * 02960000
//*                                                                  * 02970000
//* NOTE: THIS STEP MUST BE CUSTOMIZED AS FOLLOWS FOR SMP:           * 02980000
//* 1. LOCATE AND CHANGE THE FOLLOWING STRINGS TO THE VALUES YOU     * 02990000
//*    SPECIFIED FOR THEM IN JOB DSNTIJAE:                           * 03000000
//*    A.'?SMPPRE?' TO THE PREFIX OF YOUR SMP LIBRARY NAME.          * 03010000
//*    B.'?SMPMLQ?' TO THE MIDDLE LEVEL QUALIFIER OF YOUR SMP CSI    * 03020000
//* 2. UPDATE SYSOUT CLASSES AS DESIRED (DEFAULT IS '*')             * 03030000
//******************************************************************** 03040000
//DSNTIMQ EXEC PGM=GIMSMP,PARM='CSI=?SMPPRE?.?SMPMLQ?.CSI',            03050000
//             REGION=4096K,COND=(2,LT)                                03060000
```

```
//SYSPRINT DD  SYSOUT=*                                                 03070000
//SYSUDUMP DD  SYSOUT=*                                                 03080000
//SMPCNTL DD *                                                          03090000
    SET BDY(DSNTARG).                                                   03100000
    JCLIN.                                                              03110000
//SMPJCLIN DD DISP=SHR,                                                 03120000
//            DSN=DB2V610Z.NEW.SDSNSAMP(DSNTIJUZ)                       03130000
//*                                                                     03140000
```

B.21 DSNTEJ2U - DB2 sample JCL to create user defined functions

```
//PAOLOR2 JOB (999,POK),'PAOLOR2',CLASS=A,MSGCLASS=T,                   00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                   00000002
/*JOBPARM L=999,SYSAFF=SC63                                            00000003
//*********************************************************************00000100
//*  NAME = DSNTEJ2U                                                  *00000200
//*                                                                    *00000300
//*  DESCRIPTIVE NAME = DB2 SAMPLE APPLICATION                         *00000400
//*                     PHASE 2                                        *00000500
//*                     USER DEFINED FUNCTIONS (C/C++)                 *00000600
//*                                                                    *00000700
//*  NOTE: THE C++ FUNCTIONS ARE REMOVED FROM THIS JCL                 *00000600
//*  NOTE: SUBSTITUTE YOUR INSTALLATION DATASET NAMES WHERE APPROPRIATE*00000600
//*  NOTE: WHERE THE ! IS FOUND BELOW.                                 *00000600
//*                                                                    *00000700
//*    LICENSED MATERIALS - PROPERTY OF IBM                            *00000800
//*    5645-DB2                                                        *00000900
//*    (C) COPYRIGHT 1998 IBM CORP.  ALL RIGHTS RESERVED.              *00001000
//*                                                                    *00001100
//*    STATUS = VERSION 6                                              *00001200
//*                                                                    *00001300
//*  FUNCTION = THIS JCL PREPARES THE FOLLOWING DB2 USER-DEFINED       *00001400
//*             FUNCTIONS (UDF'S) AND A DRIVER PROGRAM TO INVOKE THEM. *00001500
//*                                                                    *00001600
//*     NOTES = ENSURE THAT LINE NUMBER SEQUENCING IS SET 'ON' IF      *00001700
//*             THIS JOB IS SUBMITTED FROM AN ISPF EDIT SESSION        *00001800
//*                                                                    *00001900
//*             THIS JOB IS RUN AFTER PHASE 1.                         *00002000
//*                                                                    *00002100
//* CHANGE ACTIVITY =                                                  *00002200
//*********************************************************************00002300
//*                                                                     00002400
//JOBLIB  DD  DSN=CEE.V!R!M!.SCEERUN,DISP=SHR                           00002500
//        DD  DSN=DSN!!0.SDSNEXIT,DISP=SHR                              00002600
//        DD  DSN=DSN!!0.SDSNLOAD,DISP=SHR                              00002700
//        DD  DSN=DSN!!0.RUNLIB.LOAD,DISP=SHR                           00002800
//*                                                                     00002900
//*        STEP  1: DROP ANY EXISTING DB2 SAMPLE UDF'S                  00003000
//*                                                                     00003100
//PH02US01 EXEC PGM=IKJEFT01,DYNAMNBR=20                               00003200
//SYSTSPRT DD SYSOUT=*                                                  00003300
//SYSTSIN  DD *                                                         00003400
 DSN SYSTEM(DSN)                                                       00003500
 RUN PROGRAM(DSNTIAD)          -                                       00003600
     PLAN(DSNTIA!!)            -                                       00003700
     LIB('DSN!!0.RUNLIB.LOAD') -                                       00003800
     PARM('RC0')                                                       00003900
//SYSPRINT DD SYSOUT=*                                                  00004000
//SYSUDUMP DD SYSOUT=*                                                  00004100
//SYSIN    DD *                                                         00004200
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUCDDVV   RESTRICT;                  00004300
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUCDVVV   RESTRICT;                  00004400
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUADV     RESTRICT;                  00004500
                                                                       00004600
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUCTTVV   RESTRICT;                  00004700
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUCTVVV   RESTRICT;                  00004800
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUATV     RESTRICT;                  00004900
                                                                       00005000
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUCYFV    RESTRICT;                  00005100
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUCYFVV   RESTRICT;                  00005200
                                                                       00005300
 DROP SPECIFIC FUNCTION  DSN8.DSN8EUDND     RESTRICT;                  00005400
 DROP SPECIFIC FUNCTION  DSN8.DSN8EUDNV     RESTRICT;                  00005500
                                                                       00005600
 DROP SPECIFIC FUNCTION  DSN8.DSN8EUMND     RESTRICT;                  00005700
 DROP SPECIFIC FUNCTION  DSN8.DSN8EUMNV     RESTRICT;                  00005800
```

```
                                                                        00005900
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTINV    RESTRICT;                   00006000
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTINVV   RESTRICT;                   00006100
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTINVVV  RESTRICT;                   00006200
                                                                        00006300
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTISV    RESTRICT;                   00006400
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTISVV   RESTRICT;                   00006500
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTISVVV  RESTRICT;                   00006600
                                                                        00006700
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTILV    RESTRICT;                   00006800
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTILVV   RESTRICT;                   00006900
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUTILVVV  RESTRICT;                   00007000
                                                                        00007100
 DROP SPECIFIC FUNCTION  DSN8.DSN8DUWFV     RESTRICT;                   00007200
//*                                                                     00007300
//*        STEP  2: DEFINE SAMPLE UDF'S TO DB2                          00007400
//*                                                                     00007500
//PH02US02 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                    00007600
//SYSTSPRT DD SYSOUT=*                                                  00007700
//SYSTSIN  DD *                                                         00007800
 DSN SYSTEM(DSN)                                                        00007900
 RUN PROGRAM(DSNTIAD)          -                                        00008000
     PLAN(DSNTIA!!)            -                                        00008100
     LIB('DSN!!0.RUNLIB.LOAD')                                          00008200
//SYSPRINT DD SYSOUT=*                                                  00008300
//SYSUDUMP DD SYSOUT=*                                                  00008400
//SYSIN    DD *                                                         00008500
                                                                        00008600
 CREATE FUNCTION                                                        00008700
   DSN8.ALTDATE(                                                        00008800
     VARCHAR(13) )                                                      00008900
   RETURNS                                                              00009000
     VARCHAR(17)                                                        00009100
   SPECIFIC DSN8.DSN8DUADV                                              00009200
   LANGUAGE C                                                           00009300
   DETERMINISTIC                                                        00009400
   NO SQL                                                               00009500
   EXTERNAL NAME DSN8DUAD                                               00009600
   PARAMETER STYLE DB2SQL                                               00009700
   NULL CALL                                                            00009800
   NO EXTERNAL ACTION                                                   00009900
   NO SCRATCHPAD                                                        00010000
   NO FINAL CALL                                                        00010100
   ALLOW PARALLEL                                                       00010200
   NO COLLID                                                            00010300
   ASUTIME LIMIT 5                                                      00010400
   STAY RESIDENT NO                                                     00010500
   PROGRAM TYPE SUB                                                     00010600
   WLM ENVIRONMENT WLMENV                                               00010700
   EXTERNAL SECURITY DB2                                                00010800
   NO DBINFO;                                                           00010900
                                                                        00011000
 CREATE FUNCTION                                                        00011100
   DSN8.ALTDATE(                                                        00011200
     VARCHAR(17),                                                       00011300
     VARCHAR(13),                                                       00011400
     VARCHAR(13) )                                                      00011500
   RETURNS                                                              00011600
     VARCHAR(17)                                                        00011700
   SPECIFIC DSN8.DSN8DUCDVVV                                            00011800
   LANGUAGE C                                                           00011900
   DETERMINISTIC                                                        00012000
   NO SQL                                                               00012100
   EXTERNAL NAME DSN8DUCD                                               00012200
   PARAMETER STYLE DB2SQL                                               00012300
   NULL CALL                                                            00012400
   NO EXTERNAL ACTION                                                   00012500
   NO SCRATCHPAD                                                        00012600
   NO FINAL CALL                                                        00012700
   ALLOW PARALLEL                                                       00012800
   NO COLLID                                                            00012900
   ASUTIME LIMIT 5                                                      00013000
   STAY RESIDENT NO                                                     00013100
   PROGRAM TYPE SUB                                                     00013200
   WLM ENVIRONMENT WLMENV                                               00013300
   EXTERNAL SECURITY DB2                                                00013400
   NO DBINFO;                                                           00013500
                                                                        00013600
```

```
CREATE FUNCTION                                                 00013700
  DSN8.ALTDATE(                                                 00013800
    DATE,                                                       00013900
    VARCHAR(13),                                                00014000
    VARCHAR(13) )                                               00014100
  RETURNS                                                       00014200
    VARCHAR(17)                                                 00014300
  SPECIFIC DSN8.DSN8DUCDDVV                                     00014400
  SOURCE SPECIFIC DSN8.DSN8DUCDVVV;                             00014500
                                                                00014600
CREATE FUNCTION                                                 00014700
  DSN8.ALTTIME(                                                 00014800
    VARCHAR(14) )                                               00014900
  RETURNS                                                       00015000
    VARCHAR(11)                                                 00015100
  SPECIFIC DSN8.DSN8DUATV                                       00015200
  LANGUAGE C                                                    00015300
  DETERMINISTIC                                                 00015400
  NO SQL                                                        00015500
  EXTERNAL NAME DSN8DUAT                                        00015600
  PARAMETER STYLE DB2SQL                                        00015700
  NULL CALL                                                     00015800
  NO EXTERNAL ACTION                                            00015900
  NO SCRATCHPAD                                                 00016000
  NO FINAL CALL                                                 00016100
  ALLOW PARALLEL                                                00016200
  NO COLLID                                                     00016300
  ASUTIME LIMIT 5                                               00016400
  STAY RESIDENT NO                                              00016500
  PROGRAM TYPE SUB                                              00016600
  WLM ENVIRONMENT WLMENV                                        00016700
  EXTERNAL SECURITY DB2                                         00016800
  NO DBINFO;                                                    00016900
                                                                00017000
CREATE FUNCTION                                                 00017100
  DSN8.ALTTIME(                                                 00017200
    VARCHAR(11),                                                00017300
    VARCHAR(14),                                                00017400
    VARCHAR(14) )                                               00017500
  RETURNS                                                       00017600
    VARCHAR(11)                                                 00017700
  SPECIFIC DSN8.DSN8DUCTVVV                                     00017800
  LANGUAGE C                                                    00017900
  DETERMINISTIC                                                 00018000
  NO SQL                                                        00018100
  EXTERNAL NAME DSN8DUCT                                        00018200
  PARAMETER STYLE DB2SQL                                        00018300
  NULL CALL                                                     00018400
  NO EXTERNAL ACTION                                            00018500
  NO SCRATCHPAD                                                 00018600
  NO FINAL CALL                                                 00018700
  ALLOW PARALLEL                                                00018800
  NO COLLID                                                     00018900
  ASUTIME LIMIT 5                                               00019000
  STAY RESIDENT NO                                              00019100
  PROGRAM TYPE SUB                                              00019200
  WLM ENVIRONMENT WLMENV                                        00019300
  EXTERNAL SECURITY DB2                                         00019400
  NO DBINFO;                                                    00019500
                                                                00019600
CREATE FUNCTION                                                 00019700
  DSN8.ALTTIME(                                                 00019800
    TIME,                                                       00019900
    VARCHAR(14),                                                00020000
    VARCHAR(14) )                                               00020100
  RETURNS                                                       00020200
    VARCHAR(11)                                                 00020300
  SPECIFIC DSN8.DSN8DUCTTVV                                     00020400
  SOURCE SPECIFIC DSN8.DSN8DUCTVVV;                             00020500
                                                                00020600
CREATE FUNCTION                                                 00020700
  DSN8.CURRENCY(                                                00020800
    FLOAT,                                                      00020900
    VARCHAR(2) )                                                00021000
  RETURNS                                                       00021100
    VARCHAR(19)                                                 00021200
  SPECIFIC DSN8.DSN8DUCYFV                                      00021300
  LANGUAGE C                                                    00021400
```

```
  DETERMINISTIC                                                 00021500
  NO SQL                                                        00021600
  EXTERNAL NAME DSN8DUCY                                        00021700
  PARAMETER STYLE DB2SQL                                        00021800
  NULL CALL                                                     00021900
  NO EXTERNAL ACTION                                            00022000
  NO SCRATCHPAD                                                 00022100
  NO FINAL CALL                                                 00022200
  ALLOW PARALLEL                                                00022300
  NO COLLID                                                     00022400
  ASUTIME LIMIT 5                                               00022500
  STAY RESIDENT NO                                              00022600
  PROGRAM TYPE MAIN                                             00022700
  WLM ENVIRONMENT WLMENV                                        00022800
  EXTERNAL SECURITY DB2                                         00022900
  NO DBINFO;                                                    00023000
                                                                00023100
CREATE FUNCTION                                                 00023200
  DSN8.CURRENCY(                                                00023300
    FLOAT,                                                      00023400
    VARCHAR(2),                                                 00023500
    VARCHAR(5) )                                                00023600
  RETURNS                                                       00023700
    VARCHAR(19)                                                 00023800
  SPECIFIC DSN8.DSN8DUCYFVV                                     00023900
  LANGUAGE C                                                    00024000
  DETERMINISTIC                                                 00024100
  NO SQL                                                        00024200
  EXTERNAL NAME DSN8DUCY                                        00024300
  PARAMETER STYLE DB2SQL                                        00024400
  NULL CALL                                                     00024500
  NO EXTERNAL ACTION                                            00024600
  NO SCRATCHPAD                                                 00024700
  NO FINAL CALL                                                 00024800
  ALLOW PARALLEL                                                00024900
  NO COLLID                                                     00025000
  ASUTIME LIMIT 5                                               00025100
  STAY RESIDENT NO                                              00025200
  PROGRAM TYPE MAIN                                             00025300
  WLM ENVIRONMENT WLMENV                                        00025400
  EXTERNAL SECURITY DB2                                         00025500
  NO DBINFO;                                                    00025600
                                                                00025700
CREATE FUNCTION                                                 00025800
  DSN8.DAYNAME(                                                 00025900
    VARCHAR(10) )                                               00026000
  RETURNS                                                       00026100
    VARCHAR(9)                                                  00026200
  SPECIFIC DSN8.DSN8EUDNV                                       00026300
  LANGUAGE C                                                    00026400
  DETERMINISTIC                                                 00026500
  NO SQL                                                        00026600
  EXTERNAL NAME DSN8EUDN                                        00026700
  PARAMETER STYLE DB2SQL                                        00026800
  NULL CALL                                                     00026900
  NO EXTERNAL ACTION                                            00027000
  NO SCRATCHPAD                                                 00027100
  NO FINAL CALL                                                 00027200
  ALLOW PARALLEL                                                00027300
  NO COLLID                                                     00027400
  ASUTIME LIMIT 5                                               00027500
  STAY RESIDENT NO                                              00027600
  PROGRAM TYPE SUB                                              00027700
  WLM ENVIRONMENT WLMENV                                        00027800
  EXTERNAL SECURITY DB2                                         00027900
  NO DBINFO;                                                    00028000
                                                                00028100
CREATE FUNCTION                                                 00028200
  DSN8.DAYNAME(                                                 00028300
    DATE )                                                      00028400
  RETURNS                                                       00028500
    VARCHAR(9)                                                  00028600
  SPECIFIC DSN8.DSN8EUDND                                       00028700
  SOURCE SPECIFIC DSN8.DSN8EUDNV;                               00028800
                                                                00028900
CREATE FUNCTION                                                 00029000
  DSN8.MONTHNAME(                                               00029100
    VARCHAR(10) )                                               00029200
```

```
  RETURNS                                                          00029300
    VARCHAR(9)                                                     00029400
  SPECIFIC DSN8.DSN8EUMNV                                          00029500
  LANGUAGE C                                                       00029600
  DETERMINISTIC                                                    00029700
  NO SQL                                                           00029800
  EXTERNAL NAME DSN8EUMN                                           00029900
  PARAMETER STYLE DB2SQL                                           00030000
  NULL CALL                                                        00030100
  NO EXTERNAL ACTION                                               00030200
  NO SCRATCHPAD                                                    00030300
  NO FINAL CALL                                                    00030400
  ALLOW PARALLEL                                                   00030500
  NO COLLID                                                        00030600
  ASUTIME LIMIT 5                                                  00030700
  STAY RESIDENT NO                                                 00030800
  PROGRAM TYPE SUB                                                 00030900
  WLM ENVIRONMENT WLMENV                                           00031000
  EXTERNAL SECURITY DB2                                            00031100
  NO DBINFO;                                                       00031200
                                                                   00031300
CREATE FUNCTION                                                    00031400
  DSN8.MONTHNAME(                                                  00031500
    DATE )                                                         00031600
  RETURNS                                                          00031700
    VARCHAR(9)                                                     00031800
  SPECIFIC DSN8.DSN8EUMND                                          00031900
  SOURCE SPECIFIC DSN8.DSN8EUMNV;                                  00032000
                                                                   00032100
CREATE FUNCTION                                                    00032200
  DSN8.TABLE_NAME(                                                 00032300
    VARCHAR(18) )                                                  00032400
  RETURNS                                                          00032500
    VARCHAR(18)                                                    00032600
  SPECIFIC DSN8.DSN8DUTINV                                         00032700
  LANGUAGE C                                                       00032800
  DETERMINISTIC                                                    00032900
  READS SQL DATA                                                   00033000
  EXTERNAL NAME DSN8DUTI                                           00033100
  PARAMETER STYLE DB2SQL                                           00033200
  NULL CALL                                                        00033300
  NO EXTERNAL ACTION                                               00033400
  NO SCRATCHPAD                                                    00033500
  NO FINAL CALL                                                    00033600
  ALLOW PARALLEL                                                   00033700
  COLLID DSN8DU!!                                                  00033800
  ASUTIME LIMIT 5                                                  00033900
  STAY RESIDENT NO                                                 00034000
  PROGRAM TYPE MAIN                                                00034100
  WLM ENVIRONMENT WLMENV                                           00034200
  EXTERNAL SECURITY DB2                                            00034300
  NO DBINFO;                                                       00034400
                                                                   00034500
CREATE FUNCTION                                                    00034600
  DSN8.TABLE_NAME(                                                 00034700
    VARCHAR(18),                                                   00034800
    VARCHAR(8) )                                                   00034900
  RETURNS                                                          00035000
    VARCHAR(18)                                                    00035100
  SPECIFIC DSN8.DSN8DUTINVV                                        00035200
  LANGUAGE C                                                       00035300
  DETERMINISTIC                                                    00035400
  READS SQL DATA                                                   00035500
  EXTERNAL NAME DSN8DUTI                                           00035600
  PARAMETER STYLE DB2SQL                                           00035700
  NULL CALL                                                        00035800
  NO EXTERNAL ACTION                                               00035900
  NO SCRATCHPAD                                                    00036000
  NO FINAL CALL                                                    00036100
  ALLOW PARALLEL                                                   00036200
  COLLID DSN8DU!!                                                  00036300
  ASUTIME LIMIT 5                                                  00036400
  STAY RESIDENT NO                                                 00036500
  PROGRAM TYPE MAIN                                                00036600
  WLM ENVIRONMENT WLMENV                                           00036700
  EXTERNAL SECURITY DB2                                            00036800
  NO DBINFO;                                                       00036900
                                                                   00037000
```

```
CREATE FUNCTION                                                  00037100
  DSN8.TABLE_NAME(                                               00037200
    VARCHAR(18),                                                 00037300
    VARCHAR(8),                                                  00037400
    VARCHAR(16) )                                                00037500
  RETURNS                                                        00037600
    VARCHAR(18)                                                  00037700
  SPECIFIC DSN8.DSN8DUTINVVV                                     00037800
  LANGUAGE C                                                     00037900
  DETERMINISTIC                                                  00038000
  READS SQL DATA                                                 00038100
  EXTERNAL NAME DSN8DUTI                                         00038200
  PARAMETER STYLE DB2SQL                                         00038300
  NULL CALL                                                      00038400
  NO EXTERNAL ACTION                                             00038500
  NO SCRATCHPAD                                                  00038600
  NO FINAL CALL                                                  00038700
  ALLOW PARALLEL                                                 00038800
  COLLID DSN8DU!!                                                00038900
  ASUTIME LIMIT 5                                                00039000
  STAY RESIDENT NO                                               00039100
  PROGRAM TYPE MAIN                                              00039200
  WLM ENVIRONMENT WLMENV                                         00039300
  EXTERNAL SECURITY DB2                                          00039400
  NO DBINFO;                                                     00039500
                                                                 00039600
CREATE FUNCTION                                                  00039700
  DSN8.TABLE_SCHEMA(                                             00039800
    VARCHAR(18) )                                                00039900
  RETURNS                                                        00040000
    VARCHAR(8)                                                   00040100
  SPECIFIC DSN8.DSN8DUTISV                                       00040200
  LANGUAGE C                                                     00040300
  DETERMINISTIC                                                  00040400
  READS SQL DATA                                                 00040500
  EXTERNAL NAME DSN8DUTI                                         00040600
  PARAMETER STYLE DB2SQL                                         00040700
  NULL CALL                                                      00040800
  NO EXTERNAL ACTION                                             00040900
  NO SCRATCHPAD                                                  00041000
  NO FINAL CALL                                                  00041100
  ALLOW PARALLEL                                                 00041200
  COLLID DSN8DU!!                                                00041300
  ASUTIME LIMIT 5                                                00041400
  STAY RESIDENT NO                                               00041500
  PROGRAM TYPE MAIN                                              00041600
  WLM ENVIRONMENT WLMENV                                         00041700
  EXTERNAL SECURITY DB2                                          00041800
  NO DBINFO;                                                     00041900
                                                                 00042000
CREATE FUNCTION                                                  00042100
  DSN8.TABLE_SCHEMA(                                             00042200
    VARCHAR(18),                                                 00042300
    VARCHAR(8) )                                                 00042400
  RETURNS                                                        00042500
    VARCHAR(8)                                                   00042600
  SPECIFIC DSN8.DSN8DUTISVV                                      00042700
  LANGUAGE C                                                     00042800
  DETERMINISTIC                                                  00042900
  READS SQL DATA                                                 00043000
  EXTERNAL NAME DSN8DUTI                                         00043100
  PARAMETER STYLE DB2SQL                                         00043200
  NULL CALL                                                      00043300
  NO EXTERNAL ACTION                                             00043400
  NO SCRATCHPAD                                                  00043500
  NO FINAL CALL                                                  00043600
  ALLOW PARALLEL                                                 00043700
  COLLID DSN8DU!!                                                00043800
  ASUTIME LIMIT 5                                                00043900
  STAY RESIDENT NO                                               00044000
  PROGRAM TYPE MAIN                                              00044100
  WLM ENVIRONMENT WLMENV                                         00044200
  EXTERNAL SECURITY DB2                                          00044300
  NO DBINFO;                                                     00044400
                                                                 00044500
CREATE FUNCTION                                                  00044600
  DSN8.TABLE_SCHEMA(                                             00044700
    VARCHAR(18),                                                 00044800
```

```
    VARCHAR(8),                                                  00044900
    VARCHAR(16) )                                                00045000
  RETURNS                                                        00045100
    VARCHAR(8)                                                   00045200
  SPECIFIC DSN8.DSN8DUTISVVV                                     00045300
  LANGUAGE C                                                     00045400
  DETERMINISTIC                                                  00045500
  READS SQL DATA                                                 00045600
  EXTERNAL NAME DSN8DUTI                                         00045700
  PARAMETER STYLE DB2SQL                                         00045800
  NULL CALL                                                      00045900
  NO EXTERNAL ACTION                                             00046000
  NO SCRATCHPAD                                                  00046100
  NO FINAL CALL                                                  00046200
  ALLOW PARALLEL                                                 00046300
  COLLID DSN8DU!!                                                00046400
  ASUTIME LIMIT 5                                                00046500
  STAY RESIDENT NO                                               00046600
  PROGRAM TYPE MAIN                                              00046700
  WLM ENVIRONMENT WLMENV                                         00046800
  EXTERNAL SECURITY DB2                                          00046900
  NO DBINFO;                                                     00047000
                                                                 00047100
CREATE FUNCTION                                                  00047200
  DSN8.TABLE_LOCATION(                                           00047300
    VARCHAR(18) )                                                00047400
  RETURNS                                                        00047500
    VARCHAR(16)                                                  00047600
  SPECIFIC DSN8.DSN8DUTILV                                       00047700
  LANGUAGE C                                                     00047800
  DETERMINISTIC                                                  00047900
  READS SQL DATA                                                 00048000
  EXTERNAL NAME DSN8DUTI                                         00048100
  PARAMETER STYLE DB2SQL                                         00048200
  NULL CALL                                                      00048300
  NO EXTERNAL ACTION                                             00048400
  NO SCRATCHPAD                                                  00048500
  NO FINAL CALL                                                  00048600
  ALLOW PARALLEL                                                 00048700
  COLLID DSN8DU!!                                                00048800
  ASUTIME LIMIT 5                                                00048900
  STAY RESIDENT NO                                               00049000
  PROGRAM TYPE MAIN                                              00049100
  WLM ENVIRONMENT WLMENV                                         00049200
  EXTERNAL SECURITY DB2                                          00049300
  NO DBINFO;                                                     00049400
                                                                 00049500
CREATE FUNCTION                                                  00049600
  DSN8.TABLE_LOCATION(                                           00049700
    VARCHAR(18),                                                 00049800
    VARCHAR(8) )                                                 00049900
  RETURNS                                                        00050000
    VARCHAR(16)                                                  00050100
  SPECIFIC DSN8.DSN8DUTILVV                                      00050200
  LANGUAGE C                                                     00050300
  DETERMINISTIC                                                  00050400
  READS SQL DATA                                                 00050500
  EXTERNAL NAME DSN8DUTI                                         00050600
  PARAMETER STYLE DB2SQL                                         00050700
  NULL CALL                                                      00050800
  NO EXTERNAL ACTION                                             00050900
  NO SCRATCHPAD                                                  00051000
  NO FINAL CALL                                                  00051100
  ALLOW PARALLEL                                                 00051200
  COLLID DSN8DU!!                                                00051300
  ASUTIME LIMIT 5                                                00051400
  STAY RESIDENT NO                                               00051500
  PROGRAM TYPE MAIN                                              00051600
  WLM ENVIRONMENT WLMENV                                         00051700
  EXTERNAL SECURITY DB2                                          00051800
  NO DBINFO;                                                     00051900
                                                                 00052000
CREATE FUNCTION                                                  00052100
  DSN8.TABLE_LOCATION(                                           00052200
    VARCHAR(18),                                                 00052300
    VARCHAR(8),                                                  00052400
    VARCHAR(16) )                                                00052500
  RETURNS                                                        00052600
```

```
    VARCHAR(16)                                                    00052700
  SPECIFIC DSN8.DSN8DUTILVVV                                       00052800
  LANGUAGE C                                                       00052900
  DETERMINISTIC                                                    00053000
  READS SQL DATA                                                   00053100
  EXTERNAL NAME DSN8DUTI                                           00053200
  PARAMETER STYLE DB2SQL                                           00053300
  NULL CALL                                                        00053400
  NO EXTERNAL ACTION                                               00053500
  NO SCRATCHPAD                                                    00053600
  NO FINAL CALL                                                    00053700
  ALLOW PARALLEL                                                   00053800
  COLLID DSN8DU!!                                                  00053900
  ASUTIME LIMIT 5                                                  00054000
  STAY RESIDENT NO                                                 00054100
  PROGRAM TYPE MAIN                                                00054200
  WLM ENVIRONMENT WLMENV                                           00054300
  EXTERNAL SECURITY DB2                                            00054400
  NO DBINFO;                                                       00054500
                                                                   00054600
CREATE FUNCTION                                                    00054700
  DSN8.WEATHER(                                                    00054800
    VARCHAR(44) )                                                  00054900
  RETURNS                                                          00055000
    TABLE(                                                         00055100
      CITY          VARCHAR(30),                                   00055200
      TEMP_IN_F     INTEGER,                                       00055300
      HUMIDITY      INTEGER,                                       00055400
      WIND          VARCHAR(5),                                    00055500
      WIND_VELOCITY INTEGER,                                       00055600
      BAROMETER     FLOAT,                                         00055700
      FORECAST      VARCHAR(25) )                                  00055800
  SPECIFIC DSN8.DSN8DUWFV                                          00055900
  LANGUAGE C                                                       00056000
  DETERMINISTIC                                                    00056100
  NO SQL                                                           00056200
  EXTERNAL NAME DSN8DUWF                                           00056300
  PARAMETER STYLE DB2SQL                                           00056400
  NULL CALL                                                        00056500
  NO EXTERNAL ACTION                                               00056600
  SCRATCHPAD                                                       00056700
  FINAL CALL                                                       00056800
  DISALLOW PARALLEL                                                00056900
  NO COLLID                                                        00057000
  ASUTIME LIMIT 5                                                  00057100
  STAY RESIDENT NO                                                 00057200
  PROGRAM TYPE SUB                                                 00057300
  WLM ENVIRONMENT WLMENV                                           00057400
  EXTERNAL SECURITY DB2                                            00057500
  NO DBINFO;                                                       00057600
                                                                   00057700
GRANT EXECUTE ON SPECIFIC FUNCTION DSN8.DSN8DUADV,                 00057800
                                   DSN8.DSN8DUCDVVV,               00057900
                                   DSN8.DSN8DUCDDVV,               00058000
                                   DSN8.DSN8DUATV,                 00058100
                                   DSN8.DSN8DUCTVVV,               00058200
                                   DSN8.DSN8DUCTTVV,               00058300
                                   DSN8.DSN8DUCYFV,                00058400
                                   DSN8.DSN8DUCYFVV,               00058500
                                   DSN8.DSN8EUDNV,                 00058600
                                   DSN8.DSN8EUDND,                 00058700
                                   DSN8.DSN8EUMNV,                 00058800
                                   DSN8.DSN8EUMND,                 00058900
                                   DSN8.DSN8DUTINV,                00059000
                                   DSN8.DSN8DUTINVV,               00059100
                                   DSN8.DSN8DUTINVVV,              00059200
                                   DSN8.DSN8DUTISV,                00059300
                                   DSN8.DSN8DUTISVV,               00059400
                                   DSN8.DSN8DUTISVVV,              00059500
                                   DSN8.DSN8DUTILV,                00059600
                                   DSN8.DSN8DUTILVV,               00059700
                                   DSN8.DSN8DUTILVVV,              00059800
                                   DSN8.DSN8DUWFV                  00059900
             TO PUBLIC;                                            00060000
                                                                   00060100
//*                                                                00060200
//*        STEP  3: PREPARE EXTERNAL FOR CURRENT DATE ALTDATE UDF  00060300
//*                                                                00060400
```

```
//PH02US03 EXEC DSNHC,MEM=DSN8DUAD,COND=(4,LT),                          00060500
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',       00060600
//         PARM.C='SOURCE RENT XREF MARGINS(1,72)',                      00060700
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                  00060800
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8DUAD),                     00060900
//             DISP=SHR                                                  00061000
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                                00061100
//             DISP=SHR                                                  00061200
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8DUAD),                         00061300
//             DISP=SHR                                                  00061400
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8DUAD),                      00061500
//             DISP=SHR                                                  00061600
//LKED.SYSIN   DD *                                                      00061700
 INCLUDE SYSLIB(DSNRLI)                                                 00061800
 NAME DSN8DUAD(R)                                                       00061900
//*                                                                      00062000
//*        STEP  4: PREPARE EXTERNAL FOR GIVEN DATE ALTDATE UDF          00062100
//*                                                                      00062200
//PH02US04 EXEC DSNHC,MEM=DSN8DUCD,COND=(4,LT),                          00062300
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',       00062400
//         PARM.C='SOURCE RENT XREF MARGINS(1,72)',                      00062500
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                  00062600
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8DUCD),                     00062700
//             DISP=SHR                                                  00062800
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                                00062900
//             DISP=SHR                                                  00063000
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8DUCD),                         00063100
//             DISP=SHR                                                  00063200
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8DUCD),                      00063300
//             DISP=SHR                                                  00063400
//LKED.SYSIN   DD *                                                      00063500
 INCLUDE SYSLIB(DSNRLI)                                                 00063600
 NAME DSN8DUCD(R)                                                       00063700
//*                                                                      00063800
//*        STEP  5: PREPARE EXTERNAL FOR CURRENT TIME ALTTIME UDF        00063900
//*                                                                      00064000
//PH02US05 EXEC DSNHC,MEM=DSN8DUAT,COND=(4,LT),                          00064100
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',       00064200
//         PARM.C='SOURCE RENT XREF MARGINS(1,72)',                      00064300
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                  00064400
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8DUAT),                     00064500
//             DISP=SHR                                                  00064600
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                                00064700
//             DISP=SHR                                                  00064800
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8DUAT),                         00064900
//             DISP=SHR                                                  00065000
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8DUAT),                      00065100
//             DISP=SHR                                                  00065200
//LKED.SYSIN   DD *                                                      00065300
 INCLUDE SYSLIB(DSNRLI)                                                 00065400
 NAME DSN8DUAT(R)                                                       00065500
//*                                                                      00065600
//*        STEP  6: PREPARE EXTERNAL FOR GIVEN TIME ALTTIME UDF          00065700
//*                                                                      00065800
//PH02US06 EXEC DSNHC,MEM=DSN8DUCT,COND=(4,LT),                          00065900
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',       00066000
//         PARM.C='SOURCE RENT XREF MARGINS(1,72)',                      00066100
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                  00066200
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8DUCT),                     00066300
//             DISP=SHR                                                  00066400
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                                00066500
//             DISP=SHR                                                  00066600
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8DUCT),                         00066700
//             DISP=SHR                                                  00066800
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8DUCT),                      00066900
//             DISP=SHR                                                  00067000
//LKED.SYSIN   DD *                                                      00067100
 INCLUDE SYSLIB(DSNRLI)                                                 00067200
 NAME DSN8DUCT(R)                                                       00067300
//*                                                                      00067400
//*        STEP  7: PREPARE EXTERNAL FOR CURRENCY UDF                    00067500
//*                                                                      00067600
//PH02US07 EXEC DSNHC,MEM=DSN8DUCY,COND=(4,LT),                          00067700
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',       00067800
//         PARM.C='SOURCE RENT XREF MARGINS(1,72)',                      00067900
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                  00068000
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8DUCY),                     00068100
//             DISP=SHR                                                  00068200
```

```
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                                  00068300
//             DISP=SHR                                                    00068400
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8DUCY),                           00068500
//             DISP=SHR                                                    00068600
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8DUCY),                        00068700
//             DISP=SHR                                                    00068800
//LKED.SYSIN   DD *                                                        00068900
 INCLUDE SYSLIB(DSNRLI)                                                   00069000
 NAME DSN8DUCY(R)                                                         00069100
//*                                                                        00069200
//*        STEP  8: PREPARE EXTERNAL FOR DAYNAME UDF                       00069300
//*                                                                        00069400
//PH02US08 EXEC DSNHCPP,MEM=DSN8EUDN,COND=(4,LT),                          00069500
//         PARM.PC='HOST(CPP),SOURCE,XREF,MARGINS(1,80),STDSQL(NO)',       00069600
//         PARM.CP='/CXX SOURCE XREF NOMAR OPTFILE(DD:CCOPTS)',            00069700
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                    00069800
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8EUDN),                       00069900
//             DISP=SHR                                                    00070000
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                                  00070100
//             DISP=SHR                                                    00070200
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8EUDN),                           00070300
//             DISP=SHR                                                    00070400
//CP.CCOPTS    DD DSN=SYS1.PROCLIB(DSNHCPPS),DISP=SHR                      00070500
//CP.USERLIB   DD DSN=DSN!!0.SRCLIB.DATA,                                  00070600
//             DISP=SHR                                                    00070700
//PLKED.SYSDEFSD DD  DUMMY                                                 00070800
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8EUDN),                        00070900
//             DISP=SHR                                                    00071000
//LKED.SYSIN   DD *                                                        00071100
 INCLUDE SYSLIB(DSNRLI)                                                   00071200
 NAME DSN8EUDN(R)                                                         00071300
//*                                                                        00071400
//*        STEP  9: PREPARE EXTERNAL FOR MONTHNAME UDF                     00071500
//*                                                                        00071600
//PH02US09 EXEC DSNHCPP,MEM=DSN8EUMN,COND=(4,LT),                          00071700
//         PARM.PC='HOST(CPP),SOURCE,XREF,MARGINS(1,80),STDSQL(NO)',       00071800
//         PARM.CP='/CXX SOURCE XREF NOMAR OPTFILE(DD:CCOPTS)',            00071900
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                    00072000
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8EUMN),                       00072100
//             DISP=SHR                                                    00072200
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                                  00072300
//             DISP=SHR                                                    00072400
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8EUMN),                           00072500
//             DISP=SHR                                                    00072600
//CP.CCOPTS    DD DSN=SYS1.PROCLIB(DSNHCPPS),DISP=SHR                      00072700
//CP.USERLIB   DD DSN=DSN!!0.SRCLIB.DATA,                                  00072800
//             DISP=SHR                                                    00072900
//PLKED.SYSDEFSD DD  DUMMY                                                 00073000
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8EUMN),                        00073100
//             DISP=SHR                                                    00073200
//LKED.SYSIN   DD *                                                        00073300
 INCLUDE SYSLIB(DSNRLI)                                                   00073400
 NAME DSN8EUMN(R)                                                         00073500
//*                                                                        00073600
//*        STEP 10: PREPARE EXTERNAL FOR TABLE_NAME, TABLE_SCHEMA,         00073700
//*                 AND TABLE_LOCATION UDF'S                               00073800
//*                                                                        00073900
//PH02US10 EXEC DSNHC,MEM=DSN8DUTI,COND=(4,LT),                            00074000
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',         00074100
//         PARM.C='SOURCE RENT XREF MARGINS(1,72)',                        00074200
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                    00074300
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8DUTI),                       00074400
//             DISP=SHR                                                    00074500
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                                  00074600
//             DISP=SHR                                                    00074700
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8DUTI),                           00074800
//             DISP=SHR                                                    00074900
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8DUTI),                        00075000
//             DISP=SHR                                                    00075100
//LKED.IGNORE  DD *                                                        00075200
//LKED.SYSIN   DD *                                                        00075300
 INCLUDE SYSLIB(DSNRLI)                                                   00075400
 NAME DSN8DUTI(R)                                                         00075500
//*                                                                        00075600
//*        STEP 11: BIND PACKAGE FOR TABLE_NAME, TABLE_SCHEMA, AND         00075700
//*                 TABLE_LOCATION UDF'S                                   00075800
//*                                                                        00075900
//PH02US11 EXEC PGM=IKJEFT01,COND=(4,LT)                                   00076000
```

```
//DBRMLIB  DD  DSN=DSN!!0.DBRMLIB.DATA,DISP=SHR                          00076100
//SYSTSPRT DD  SYSOUT=*                                                 00076200
//SYSPRINT DD  SYSOUT=*                                                 00076300
//CEEDUMP  DD  SYSOUT=*                                                 00076400
//SYSUDUMP DD  SYSOUT=*                                                 00076500
//SYSOUT   DD  SYSOUT=*                                                 00076600
//REPORT   DD  SYSOUT=*                                                 00076700
//SYSIN    DD  *                                                        00076800
//SYSTSIN  DD  *                                                        00076900
 DSN SYSTEM(DSN)                                                        00077000
 BIND PACKAGE (DSN8DU!!) MEMBER(DSN8DUTI) ACT(REP) ISO(CS)              00077100
 END                                                                    00077200
//*                                                                     00077300
//*        STEP 12: EXERCISE THE SAMPLE UDF'S                           00077400
//*                                                                     00077500
//PH02US12 EXEC PGM=IKJEFT01,COND=(4,LT),DYNAMNBR=20                    00077600
//SYSTSPRT DD  SYSOUT=*                                                 00077700
//SYSPRINT DD  SYSOUT=*                                                 00077800
//SYSUDUMP DD  SYSOUT=*                                                 00077900
//SYSTSIN  DD  *                                                        00078000
 DSN SYSTEM(DSN)                                                        00078100
 RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP!!) -                                 00078200
      LIB('DSN!!0.RUNLIB.LOAD') PARMS('/ALIGN(MID)')                    00078300
 END                                                                    00078400
//SYSIN    DD  DSN=DSN!!0.SDSNSAMP(DSNTESU),                            00078500
//             DISP=SHR                                                 00078600
//*                                                                     00078700
//*        STEP 13: PREPARE EXTERNAL FOR WEATHER UDF TABLE FUNCTION     00078800
//*                                                                     00078900
//PH02US13 EXEC DSNHC,MEM=DSN8DUWF,COND=(4,LT),                         00079000
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',      00079100
//         PARM.C='SOURCE RENT XREF MARGINS(1,72)',                     00079200
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                 00079300
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8DUWF),                    00079400
//             DISP=SHR                                                 00079500
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                               00079600
//             DISP=SHR                                                 00079700
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8DUWF),                        00079800
//             DISP=SHR                                                 00079900
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8DUWF),                     00080000
//             DISP=SHR                                                 00080100
//LKED.IGNORE  DD *                                                     00080200
//LKED.SYSIN   DD *                                                     00080300
 INCLUDE SYSLIB(DSNRLI)                                                 00080400
 NAME DSN8DUWF(R)                                                       00080500
//*                                                                     00080600
//*        STEP 14: PREPARE CLIENT FOR WEATHER UDF TABLE FUNCTION       00080700
//*                                                                     00080800
//PH02US14 EXEC DSNHC,MEM=DSN8DUWC,COND=(4,LT),                         00080900
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',      00081000
//         PARM.C='SOURCE RENT XREF MARGINS(1,72)',                     00081100
//         PARM.LKED='MAP,RENT,REUS,AMODE=31,RMODE=ANY'                 00081200
//PC.DBRMLIB   DD DSN=DSN!!0.DBRMLIB.DATA(DSN8DUWC),                    00081300
//             DISP=SHR                                                 00081400
//PC.SYSLIB    DD DSN=DSN!!0.SRCLIB.DATA,                               00081500
//             DISP=SHR                                                 00081600
//PC.SYSIN     DD DSN=DSN!!0.SDSNSAMP(DSN8DUWC),                        00081700
//             DISP=SHR                                                 00081800
//LKED.SYSLMOD DD DSN=DSN!!0.RUNLIB.LOAD(DSN8DUWC),                     00081900
//             DISP=SHR                                                 00082000
//LKED.IGNORE  DD *                                                     00082100
//LKED.SYSIN   DD *                                                     00082200
 INCLUDE SYSLIB(DSNELI)                                                 00082300
 INCLUDE SYSLIB(DSNTIAR)                                                00082400
 NAME DSN8DUWC(R)                                                       00082500
//*                                                                     00082600
//*        STEP 15: BIND PACKAGE & PLAN FOR WEATHER TBL FUNC CLIENT     00082700
//*                                                                     00082800
//PH02US15 EXEC PGM=IKJEFT01,COND=(4,LT)                                00082900
//DBRMLIB  DD  DSN=DSN!!0.DBRMLIB.DATA,DISP=SHR                          00083000
//SYSTSPRT DD  SYSOUT=*                                                 00083100
//SYSPRINT DD  SYSOUT=*                                                 00083200
//CEEDUMP  DD  SYSOUT=*                                                 00083300
//SYSUDUMP DD  SYSOUT=*                                                 00083400
//SYSOUT   DD  SYSOUT=*                                                 00083500
//REPORT   DD  SYSOUT=*                                                 00083600
//SYSIN    DD  *                                                        00083700
//SYSTSIN  DD  *                                                        00083800
```

```
DSN SYSTEM(DSN)                                                          00083900
BIND PACKAGE (DSN8DU!!) MEMBER(DSN8DUWC) ACT(REP) ISO(CS)                00084000
BIND PLAN (DSN8UW!!) PKLIST(DSN8DU!!.*) ACT(REP) ISO(CS) SQLRULES(DB2)   00084100
RUN PROGRAM(DSNTIAD) PLAN(DSNTIA!!) -                                    00084200
    LIB('DSN!!0.RUNLIB.LOAD')                                            00084300
END                                                                      00084400
```

B.22 DSNTEJ2D - Sample C program execution using sample tables

```
//DB2XF JOB (999,POK),'DB2V610X',CLASS=A,MSGCLASS=T,                     00000001
// NOTIFY=PAOLOR2,TIME=1440,REGION=0M                                    00000002
/*JOBPARM L=999,SYSAFF=SC63                                             00000003
//*********************************************************************00010000
//*  NAME = DSNTEJ2D                                                   *00020000
//*                                                                     *00030000
//*  DESCRIPTIVE NAME = DB2 SAMPLE APPLICATION                          *00040000
//*                     PHASE 2                                         *00050000
//*                     C                                               *00060000
//*                                                                     *00070000
//*    LICENSED MATERIALS - PROPERTY OF IBM                             *00080000
//*    5645-DB2                                                         *00090000
//*    (C) COPYRIGHT 1982, 1998 IBM CORP.  ALL RIGHTS RESERVED.         *00100000
//*                                                                     *00110000
//*    STATUS = VERSION 6                                               *00120000
//*                                                                     *00130000
//*  FUNCTION = THIS JCL PERFORMS THE PHASE 2 C LANGUAGE SETUP FOR      *00140000
//*             THE SAMPLE APPLICATIONS.  IT PREPARES AND EXECUTES      *00150000
//*             C BATCH PROGRAMS.                                       *00160000
//*                                                                     *00170000
//*     NOTES = ENSURE THAT LINE NUMBER SEQUENCING IS SET 'ON' IF       *00180000
//*             THIS JOB IS SUBMITTED FROM AN ISPF EDIT SESSION         *00190000
//*                                                                     *00200000
//*             THIS JOB IS RUN AFTER PHASE 1.                          *00210000
//*                                                                     *00220000
//* CHANGE ACTIVITY =                                                   *00230000
//*********************************************************************00240000
//*                                                                      00250000
//JOBLIB  DD  DSN=CEE.SCEERUN,DISP=SHR                                   00260000
//        DD  DSN=DSN610.QPP.SDSNLOAD,DISP=SHR                           00270000
//*                                                                      00280000
//*        STEP 1 : PREPARE ERROR MESSAGE ROUTINE                        00290000
//PH02DS01 EXEC DSNHC,MEM=DSN8MDG,                                       00300000
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',       00310000
//         PARM.C='SOURCE XREF MARGINS(1,72)',                           00320000
//         PARM.LKED='NCAL,MAP,AMODE=31,RMODE=ANY'                       00330000
//PC.DBRMLIB   DD DSN=DB2V610X.DBRMLIB.DATA(DSN8MDG),                    00340000
//             DISP=SHR                                                  00350000
//PC.SYSLIB    DD DSN=DB2V610X.SRCLIB.DATA,                              00360000
//             DISP=SHR                                                  00370000
//PC.SYSIN     DD DSN=DSN610.QPP.SDSNSAMP(DSN8MDG),                      00380000
//             DISP=SHR                                                  00390000
//LKED.SYSLMOD DD DSN=DB2V610X.RUNLIB.LOAD(DSN8MDG),                     00400000
//             DISP=SHR                                                  00410000
//*                                                                      00420000
//*        STEP 2 : PREPARE C PHONE PROGRAM                              00430000
//PH02DS02 EXEC DSNHC,MEM=DSN8BD3,                                       00440000
//         COND=(4,LT),                                                  00450000
//         PARM.PC='HOST(C),SOURCE,XREF,MARGINS(1,72),STDSQL(NO)',       00460000
//         PARM.C='SOURCE LIST MARGINS(1,72)',                           00470000
//         PARM.LKED='AMODE=31,RMODE=ANY,MAP'                            00480000
//PC.DBRMLIB   DD DSN=DB2V610X.DBRMLIB.DATA(DSN8BD3),                    00490000
//             DISP=SHR                                                  00500000
//PC.SYSLIB    DD DSN=DB2V610X.SRCLIB.DATA,                              00510000
//             DISP=SHR                                                  00520000
//PC.SYSIN     DD DSN=DSN610.QPP.SDSNSAMP(DSN8BD3),                      00530000
//             DISP=SHR                                                  00540000
//LKED.SYSLMOD DD DSN=DB2V610X.RUNLIB.LOAD(DSN8BD3),                     00550000
//             DISP=SHR                                                  00560000
//LKED.RUNLIB  DD DSN=DB2V610X.RUNLIB.LOAD,                              00570000
//             DISP=SHR                                                  00580000
//LKED.SYSIN   DD *                                                      00590000
     INCLUDE SYSLIB(DSNELI)                                            00600000
     INCLUDE RUNLIB(DSN8MDG)                                           00610000
//*                                                                      00620000
//*        STEP 3 : BIND AND RUN PROGRAMS                                00630000
```

```
//PH02DS03 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)                    00640000
//DBRMLIB   DD  DISP=SHR,DSN=DB2V610X.DBRMLIB.DATA                      00650000
//SYSTSPRT  DD  SYSOUT=*                                                00660000
//SYSPRINT  DD  SYSOUT=*                                                00670000
//CEEDUMP   DD  SYSOUT=*                                                00680000
//SYSUDUMP  DD  SYSOUT=*                                                00690000
//SYSOUT    DD  SYSOUT=*                                                00700000
//REPORT    DD  SYSOUT=*                                                00710000
//SYSIN     DD  *                                                       00720000
 GRANT BIND, EXECUTE ON PLAN DSN8BD61 TO PUBLIC;                        00730000
//SYSTSIN   DD  *                                                       00740000
 DSN SYSTEM(DB2X)                                                       00750000
 BIND PLAN(DSN8BD61) MEMBER(DSN8BD3) ACT(REP) ISOLATION(CS)             00760000
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA61)  -                                00770000
      LIB('DB2V610X.RUNLIB.LOAD')                                       00780000
 RUN  PROGRAM(DSN8BD3) PLAN(DSN8BD61) -                                 00790000
      LIB('DB2V610X.RUNLIB.LOAD')                                       00800000
 END                                                                    00810000
//CARDIN   DD  *                                                        00820000
L*                                                                      00830000
LJO%                                                                    00840000
L%SON                                                                   00850000
LSMITH                                                                  00860000
LBROWN          ALAN                                                    00870000
LBROWN          DAVID                                                   00880000
U                          0002304265                                   00890000
//*                                                                     00900000
```

Appendix C. Sample data preparation program

This program provides a quick and dirty solution to the need of calculating the length of variable length fields before passing it as input to the DB2 LOAD utility.

```
The following is a REXX program for adding the two binary bytes of length data for use by the
DB2 load utility./* REXX - Prepare variable length data for DB2 LOAD -----------------*/
/*                                                                   */
/* Description: Prepare input for DB2 LOAD adding length field to    */
/*         variable length data                                      */
/*                                                                   */
/* Input:  CTLIN    - Description of Input data                      */
/*         DATAIN   - Input data to be converted                     */
/* Output: SYSPRINT - Messages                                       */
/*         DATAOUT  - Output data for DB2 LOAD                       */
/*         CTLOUT   - Description of Output data ... da fare         */
/*                                                                   */
/*------------------------------------------------------- C.Venturini */

  ExitRC = 0                  /* Init Return Code = OK       */
  IoMeC = Sysvar('SYSICMD')   /* Nome di questa Exec         */
  EnvXe = Sysvar('SYSENV')    /* Environment: FORE o BACK    */

  If EnvXe = "FORE" Then
    Do
      Say "This eXec must be run in BATCH only"
      EXIT 16
    End

  /* Try writing to DDName DATAOUT ------------------------------*/
  "EXECIO * DISKW DATAOUT (FINIS STEM BegLine.)"
  If RC > 4 Then
    Do
      Say "Error writing DDname DATAOUT"
      Say "Return Code from EXECIO:" RC
      AllocError = 1
    End

  /* Try writing to DDName CTLOUT -------------------------------*/
  "EXECIO * DISKW CTLOUT  (FINIS STEM BegLine.)"
  If RC > 4 Then
    Do
      Say "Error writing DDname CTLOUT"
      Say "Return Code from EXECIO:" RC
      AllocError = 1
    End

  /* Read data from DDName CTLIN --------------------------------*/
  "EXECIO * DISKR CTLIN   (FINIS STEM CtIn.)"
  If RC > 4 Then
    Do
      Say "Error reading DDname CTLIN"
      Say "Return Code da EXECIO:" RC
      AllocError = 1
    End

  /* Read data from DDName DATAIN -------------------------------*/
  "EXECIO * DISKR DATAIN  (FINIS STEM DatIn.)"
  If RC > 4 Then
    Do
      Say "Error reading DDname DATAIN"
      Say "Return Code da EXECIO:" RC
      AllocError = 1
    End

  If AllocError = 1 Then
    Do
      Say "Error accessing files ... quitting ..."
      EXIT 16
    End

  oCtl. = "" ; oCtlIx = 0 ; oCtl.0 = 0     /* Clear output array */
```

```
oCtlIx = oCtlIx+1
oCtl.oCtlIx = "* Control file describing converted data (DATAOUT)"
oCtlIx = oCtlIx+1
oCtl.oCtlIx = "* (same format as CTLIN)"

/* Clear arrays to be loaded from CTLIN ----------------------*/
cBeg. = "" ; cLen. = "" ; cTyp. = "" ; j = 0
oBeg = 1                              /* init output position */

/* Load CTLIN into arrays doing some validity check -----------*/
Do i = 1 to CtIn.0
  If Left(CtIn.i,1) = "*" Then Iterate    /* Skip comment line */
  j = j+1
  Parse VAR CtIn.i w1 w2 w3 .
  If ((w3 <> "F") & (w3 <> "V")) Then
    Do
      Say "Wrong data in CTLIN ..."
      Say "Type must be 'F' or 'V' ... "
      Exit 8
    End
  cBeg.j = Strip(w1)
  cLen.j = Strip(w2)
  cTyp.j = Strip(w3)
  oLen = cLen.j
  oTyp = cTyp.j
  oCtlIx = oCtlIx+1
  oCtl.oCtlIx = oBeg oLen oTyp
  If w3 = "V" Then
    Do
      oBeg = oBeg+oLen+2
    End
  Else
    Do
      oBeg = oBeg+oLen
    End
End
NumFields = j

oCtl.0 = oCtlIx
"EXECIO * DISKW CTLOUT  (FINIS STEM oCtl.)"
If RC > 4 Then
  Do
    Say "Error writing DDname CTLOUT"
    Say "Return Code from EXECIO:" RC
    EXIT 8
  End

oData. = "" ; oDataIx = 0 ; oData.0 = 0  /* Clear output array */

/* Convert DATAIN into DATAOUT --------------------------------*/
Do i = 1 to DatIn.0
  wOut = ""                             /* clear output row */
  Do j = 1 to NumFields
    wField = Substr(DatIn.i,cBeg.j,cLen.j)
    If cTyp.j = "V" Then
      Do
        wF = Strip(wField)                /* remove blanks  */
        wL = Length(wF)                   /* get length     */
        If wL = 0 Then wL = 1             /*gag*/
        wLX1 = Right('000'D2X(wL),4)      /* convert to hex */
        wLX2 = X2C(wLX1)                  /* ... binary     */
        wOut = wOut³³wLX2³³wField    /* append length & data */
      End
    Else
      Do
        If wField = "" Then wField = copies(' ',clen.j) /*gag*/
        wOut = wOut³³wField        /* append data to out row */
      End
  End
  oDataIx = oDataIx+1
  oData.oDataIx = wOut
End

oData.0 = oDataIx
"EXECIO * DISKW DATAOUT (FINIS STEM oData.)"
If RC > 4 Then
  Do
    Say "Error writing DDname DATAOUT"
```

```
    Say "Return Code from EXECIO:" RC
    EXIT 8
  End

EXIT 0
```

Appendix D. OS/390 TSO tools and tips

In this appendix we have included some very basic tools and tips on navigation within TSO for those that approach TSO for the first time.

D.1 TSO and ISPF

The umbrella application which you will use inside of TSO is the Interactive Structured Programming Facility (ISPF). ISPF is menu driven, and you can navigate to any point inside it using the menus. Once you have become familiar with its various options you can use the fastpath facility. A fastpath command is signaled by using "=" in front of the path to the option. Fastpath navigation may be done from the command line or from any line with leader dots. The option is expressed as a number or string of numbers that designate the menu and menu option that it represents. The numbers are separated by dots. As shown in Figure 117, the path to the data set option is menu 3 option 2. This is typed as 3.2. If you were in the edit option (option 2) and you wanted to go to the data set utility 3.2 you would type that, with an "=" on the command line.

```
Menu  RefList  RefMode  Utilities  LMF  Workstation  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                               Edit Entry Panel
Command ===> =3.2

ISPF Library:
   Project . . . PAOLOR2
   Group . . . . CIPROS   . . .          . . .          . . .
   Type  . . . . JCL
   Member  . . .                (Blank or pattern for member selection list)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .          (If not cataloged)

Workstation File:
   File Name . . . . .
                                        Options
Initial Macro  . . . .                  /  Confirm Cancel/Move/Replace
Profile Name . . . . .                     Mixed Mode
Format Name  . . . . .                     Edit on Workstation
Data Set Password  . .                     Preserve VB record length
```

Figure 117. Using a fastpath command

The command could be typed in at the member line, as well, since it has "leader dots".

TSO

Logging on to TSO will be the first step. You will usually see some kind of panel that lists your options and will allow you to enter that option on a command line.

At the ITSO the system we used was SC63TS. You would enter this on the command line and press enter. Figure 118 is an example screen.

```
EMSP03        ITSO Application Selection       Help: 293-1660 Term  : TCP38025
                                               Date: 07/12/99 Time  : 20:39:14
                                               User: TCP38025 Group : DLPUBGRP
Esc PA3  Cmd PF10 Prefix $$       Print        Broadcast:    Printer:
Name-----Status-M/B-JmpK | Name-----Status-M/B-JmpK | Name-----Status-M/B-JmpK
ARGENTIN 11:19      PF22 | HONGKONG 11:18      PF22 | RALYDPD6 11:16      PF22
ASEAHONE 11:18      PF22 | IBMNET   11:18      PF22 | RETAIN   11:19      PF22
AUSTRALI 11:18      PF22 | IMS510   11:18      PF22 | RETAIN1  11:19      PF22
AUSTRIA  11:18      PF22 | IRELAND  11:18      PF22 | SCSCPAAV 11:18      PF22
BRAZIL   11:18      PF22 | IS2      11:19      PF22 | SCSCPAAY 11:18      PF22
CANADA   11:19      PF22 | MMS      11:19      PF22 | SCSCPAAZ 11:18      PF22
CCDNWP   11:18      PF22 | NETVMVS  11:18      PF22 | SCSCPAA1 20:00      PF22
DENMARK  11:18      PF22 | NETVNET  11:18      PA2  | SCSCPAA2 11:18      PF22
EHONE2   11:18      PF22 | NETVPOK  11:18      PF22 | SCSCPAA3 11:18      PF22
EHONE4   11:18      PF22 | NETVVMXA 11:18      PF22 | SCSCPAA4 11:18      PF22
EMEA     11:18      PF22 | NORWAY   11:18      PF22 | SCSCPAA5 11:18      PF22
FINLAND  11:18      PF22 | PORTUGAL 11:18      PF22 | SCSCPAA6 11:18      PF22
FUJISAWA 11:19      PF22 | RALAPPS  10:00      PF22 | SCSCPAA9 11:18      PF22
HONEFB   11:19      PF22 | RALTSO   09:37      PF22 | SCSCPCA1 11:18      PF22
- Enter application name or a command. (LOGOFF terminates all sessions..) ----


==> sc63ts_____________   ________   ________   ________              Page 001
F1=Help  PF2=Lang  PF3=Disc  PF4=Keys  PF7=Backw  PF8=Forw
```

Figure 118. Application selection panel

On the next panel in Figure 119, enter your userid and press enter.

```
 IKJ56700A ENTER USERID -
paolor1
```

Figure 119. Enter userid panel

Then type in your password. Your password will not appear. Then press enter, as shown in Figure 120.

```
    Enter LOGON parameters below:                RACF LOGON parameters:

    Userid    ===> PAOLOR1

    Password  ===>                               New Password ===>

    Procedure ===> IKJACCNT                      Group Ident  ===>

    Acct Nmbr ===> ACCNT#

    Size      ===> 6072

    Perform   ===>

    Command   ===>

    Enter an 'S' before each option desired below:
            -Nomail         -Nonotice        -Reconnect        -OIDcard

 PF1/PF13 ==> Help    PF3/PF15 ==> Logoff    PA1 ==> Attention    PA2 ==> Reshow
 You may request specific help information by entering a '?' in any entry field
```

Figure 120. Password panel

If you are successful you will see a screen like the one in Figure 121. Here you type in the name of the CLIST that will execute your ISPF session and press enter. For ITSO it is ISPPDF

```
 ICH70001I PAOLOR1  LAST ACCESS AT 19:49:03 ON MONDAY, JULY 12, 1999
 PAOLOR1 LOGON IN PROGRESS AT 21:04:05 ON JULY 12, 1999
  ============================================================
  =                                                          =
  =   This is the ITSO POK test sysplex.                     =
  =   Use the CLIST ISPPDF to access ISPF.                   =
  =   Contact Bob Haimowitz for assistance (T/L 541-3529).   =
  =   You have no one to blame but yourself.                 =
  =                                                          =
  =   SC63 and SC64 are running OS/390 V2R7 with JES2.       =
  =   SC65 is running OS/390 V2R8 with JES3 primary,         =
  =        JES2 secondary.                                   =
  =                                                13 Jun 99 =
  ============================================================
 READY
isppdf
```

Figure 121. TSO READY prompt

You do not need to enter your site's CLIST name at the ready prompt if automatically provided at logon time at your installation by your system programmer.

At this point, a master menu will be displayed with a number of options or the CLIST may take you directly to the ISPF Primary Option Menu. This menu will vary from site to site with more or less options displayed on it. At the ITSO it appears like the menu in Figure 122.

```
 Menu  Utilities  Compilers  Options  Status  Help
 ------------------------------------------------------------------------------
                           ISPF Primary Option Menu
 Option ===>

 0  Settings      Terminal and user parameters           User ID . : PAOLOR1
 1  View          Display source data or listings        Time. . . : 21:11
 2  Edit          Create or change source data           Terminal. : 3278A
 3  Utilities     Perform utility functions              Screen. . : 1
 4  Foreground    Interactive language processing        Language. : ENGLISH
 5  Batch         Submit job for language processing     Appl ID . : PDF
 6  Command       Enter TSO or Workstation commands      TSO logon : IKJACCT
 7  Dialog Test   Perform dialog testing                 TSO prefix: PAOLOR1
 8  LM Facility   Library administrator functions        System ID : SC63
 9  IBM Products  IBM program development products       MVS acct. : ACCNT#
 10 SCLM          SW Configuration Library Manager       Release . : ISPF 4.5
 11 Workplace     ISPF Object/Action Workplace


 Enter X to Terminate using log/list defaults



  F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

Figure 122. ISPF Primary Option Menu

At this point you are ready to use ISPF.

Tip:
ISPF uses PF keys to do a number of functions, such as paging up and down. To find out what they are at your site type the word "KEYS" on the command line and a panel listing them will be displayed. To return to the previous screen type "end" on the command line or press PF3.

Logoff is achieved by using PF3 to back out of the session you are in till you reach the ready prompt. At the ready prompt type in logoff. You will be returned to the original menu screen shown in Figure 123.

```
 PAOLOR1.SC63.SPFLOG1.LIST has been deleted.
 READY
logoff
```

Figure 123. Logoff command from the TSO READY prompt

ISPF/PDF

This is a powerful and comprehensive set of tools. The *OS/390 V2R7.0 TSO/E Command Reference,* SC28-1969-02, and *OS/390 V2R7.0 TSO/E User's Guide,* SC28-1968-01, cover these and other topics in detail. See Table 40.

Table 40. AIX to OS/390 system utility mapping

Tool	AIX	OS/390
Editor	vi	ISPF Editor
List	ls	ISPF Data Set Utility Dslist
New directory/PDS	mkdir	ISPF Data Set Utility Data Set
New file/member	touch, vi	For a PDS open the member as its name in edit. For a new file, create it in the data set utility and open the file as its name in edit.
Move	mv	ISPF Data Set Utility Move/Copy
Copy	cp	ISPF Data Set Utility Move/Copy
Delete	rmdir, del	ISPF Data Set Utility Data Set

Data set utility:

The data set utilities will help you create and manage data sets in OS/390. In 6.4, “Creating a PDS on OS/390” on page 117 we discuss option 3.2 and how to use it.

Edit

Option 2 off of the Primary Menu is another place you will spend a lot of time. This is the editor. With it you can edit all the various kinds of data sets common to OS/390 except for VSAM. VSAM has its own set of utilities.

The option 2 main panel looks like this the panel in Figure 124.

```
Menu  RefList  RefMode  Utilities  LMF  Workstation  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                              Edit Entry Panel
Command ===>

ISPF Library:
   Project . . . CIPROS
   Group . . . . JCL      . . .          . . .          . . .
   Type  . . . . LOAD
   Member  . . .                 (Blank or pattern for member selection list)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                       Options
Initial Macro  . . . .                 /  Confirm Cancel/Move/Replace
Profile Name . . . . .                    Mixed Mode
Format Name  . . . . .                    Edit on Workstation
Data Set Password  . .                    Preserve VB record length

 F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

Figure 124. Edit entry panel

From this panel you can enter the name of a data set to be modified. Mostly, you will be using PDS and the member parameter because that is where you will type in the name of a program, for example, and press enter.

If you intend to edit a new one just typing in the name will open a member to be used. If you should save that member before typing anything, nothing is created.

If you leave the member line blank and press enter the PDS index listing is displayed. Various information about each member is given, such as, the last time it was changed. By running the cursor down the left side of the panel to the member you want, and pressing enter, you will be taken into the editor.

Either way you select a member, the edit panel looks like the panel in Figure 125.

```
 File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
 EDIT       CIPROS.JCL.LOAD(ANECOPY) - 01.00               Columns 00001 00072
 Command ===>                                                 Scroll ===> CSR
 ****** ***************************** Top of Data ******************************
 000001 //CPRSCOPY JOB (999,POK),'PAOLOR2 ',CLASS=A,MSGCLASS=T,
 000002 // NOTIFY=PAOLOR2,TIME=1440,REGION=0M
 000003 /*JOBPARM L=999,SYSAFF=SC63
 000004 //JOBLIB  DD  DSN=DSN610A.SDSNEXIT.SC63,DISP=SHR
 000005 //        DD  DSN=DSN610A.SDSNLOAD,DISP=SHR
 000006 //*
 000007 //*        STEP  2: ESTABLISH A QUIESCE POINT
 000008 //*STEP002 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)
 000009 //* NOTE: CONDITION CODE 4 INDICATES AN IMAGE COPY CANNOT BE TAKEN
 000010 //*SYSUT1   DD  UNIT=SYSALLDA,SPACE=(40,(20,20),,,ROUND)
 000011 //*SYSIN    DD  *
 000012 //*
 000013 //* QUIESCE TABLESPACE CIPROS.CPRSBASE
 000014 //*
 000015 //*        STEP  3: TAKE IMAGE COPY OF SAMPLE TABLES
 000016 //STEP003 EXEC DSNUPROC,PARM='DB2X,DSNTEX',COND=(4,LT)
 000017 //DSNTRACE DD  SYSOUT=*
 000018 //SYSCOPY  DD  DSN=CIPROS.SYSCOPY.CPRSBASE,
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Figure 125. ISPF Edit panel

Edit commands may be entered in the command column and on the command line. The command column is the column of numbers down the left side of the panel. The numbers can be over-typed with commands. See the manuals mentioned earlier for a complete set of commands and their use.

D.2 SDSF output messages

This section discusses the process we used to view messages for our jobs held in the JES2 output queue. The following figures are used to demonstrate the process we used to view our messages.

After starting SDSF, step one is to select option H to look at jobs held in the JES2 output queue. See Figure 126.

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
HQX1900------------------  SDSF PRIMARY OPTION MENU  --------------------------
COMMAND INPUT ===> h                                          SCROLL ===> CSR

    LOG       - Display the system log
    DA        - Display active users in the sysplex
    I         - Display jobs in the JES2 input queue
    O         - Display jobs in the JES2 output queue
    H         - Display jobs in the JES2 held output queue
    ST        - Display status of jobs in the JES2 queues
    PR        - Display JES2 printers on this system
    INIT      - Display JES2 initiators on this system
    MAS       - Display JES2 members in the MAS
    LINE      - Display JES2 lines on this system
    NODE      - Display JES2 nodes on this system
    SO        - Display JES2 spool offload for this system

    Licensed Materials - Property of IBM

    5647-A01 (C) Copyright IBM Corp. 1981, 1997. All rights reserved.
    US Government Users Restricted Rights - Use, duplication or
PF 1=ifind     2=rchange   3=END       4=swap      5=split     6=help
PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

Figure 126. SDSF step one

Step two is to place a question mark next to your job name, then press enter, as shown in Figure 127.

```
 Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF HELD OUTPUT DISPLAY ALL CLASSES  LINES 23,818     LINE 1-3 (3)
 COMMAND INPUT ===>                                          SCROLL ===> CSR
 NP   JOBNAME  JOBID    OWNER    PRTY C ODISP DEST                TOT-REC  TOT-
 ?    CPRSSQLC JOB09641 PAOLOR3    64 T HOLD  LOCAL                16,837
      CPRSGLOB JOB09642 PAOLOR3    96 T HOLD  LOCAL                 6,667
      CPRSRTDI JOB09643 PAOLOR3   144 T HOLD  LOCAL                   314




 PF 1=ifind     2=rchange   3=END       4=swap      5=split     6=help
 PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

Figure 127. SDSF step two

Step 3 is to place an S next to the DDNAME of the DD you want to view, then press enter. We viewed the following DDNAMEs in Figure 128 for the following messages:

- JESYSMSG for JCL messages
- SYSTERM for precompile messages
- SYSCPRT for compile messages
- SYSPRINT for link messages

```
 Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF JOB DATA SET DISPLAY - JOB CPRSSQLC (JOB09641)    LINE 1-6 (6)
 COMMAND INPUT ===>                                          SCROLL ===> CSR
 NP   DDNAME   STEPNAME PROCSTEP DSID OWNER    C DEST              REC-CNT PAGE
      JESMSGLG JES2                 2 PAOLOR3  T LOCAL                 19
      JESJCL   JES2                 3 PAOLOR3  T LOCAL                 77
      JESYSMSG JES2                 4 PAOLOR3  T LOCAL                124
      SYSPRINT PC                 102 PAOLOR3  T LOCAL              5,294
      SYSTERM  PC                 103 PAOLOR3  T LOCAL                  8
 s    SYSCPRT  C                  106 PAOLOR3  T LOCAL             11,315
     SYSOUT   PLKED              108 PAOLOR3  T LOCAL                320
SYSPRINT LKED               110 PAOLOR3  T LOCAL                  419




 PF 1=ifind     2=rchange   3=END       4=swap      5=split     6=help
 PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

Figure 128. SDSF step three

Appendix E. Special notices

This publication is intended to help managers and professionals understand and evaluate the activities involved in performing a database conversion from Oracle 7 and AIX to DB2 UDB for OS/390 Version 6. The information in this publication is not intended as the specification of any programming interfaces that are provided by DB2 UDB for OS/390 Version 6. See the PUBLICATIONS section of the IBM Programming Announcement for DB2 UDB for OS/390 Version 6 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AS/400
AT	CT
DATABASE 2	DataJoiner
DB2	Distributed Relational Database Architecture
DRDA	eNetwork
GDDM	IBM
IMS	Intelligent Miner
MQ	MQSeries
MVS/ESA	Netfinity
OpenEdition	OS/2
OS/390	QMF
RACF	RETAIN
RMF	RS/6000
S/390	SP
SP1	System/390
VisualAge	XT
3090	3890

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

Oracle, Net8, SQL*Plus, SQL*Net are trademarks of Oracle Corporation in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries. (For a complete list of Intel trademarks see www.intel.com/tradmarx.htm)

Uniface and Polyserver are registered trademarks of Compuware Company in the United States and/or other countries and is used by IBM under special agreement.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix F. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

F.1 International Technical Support Organization publications

For information on ordering these ITSO publications see “How to get ITSO redbooks” on page 273.

- *Developing Cross-Platform DB2 Stored Procedures: SQL Procedures and the DB2 Stored Procedure Builder*, SG24-5485
- *Wow! DRDA Supports TCP/IP: DB2 Server for OS/390 and DB2 Universal Database*, SG24-2212
- *DATABASE 2 for AIX Conversion Guide Oracle 7.1 to DB2 Version 2*, SG24-2567
- *Planning for Conversion to the DB2 Family: Methodology and Practice*, GG24-4445
- *DB2 UDB for OS/390 Version 6 Performance Topics*, SG24-5351
- *Porting Applications to the OpenEdition MVS Platform*, GG24-4473
- *DataJoiner Implementation and Usage Guide*, SG24-2566
- *Mining Relational and Nonrelational Data with IBM Intelligent Miner for Data Using Oracle SPSS and SAS As Sample Data Sources*, SG24-5278
- *Ready for e-business: OS/390 Security Server Enhancements*, SG24-5158
- *Selecting a Server - The Value of S/390*, SG24-4812-01

F.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at `http://www.redbooks.ibm.com/` for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

F.3 Other publications

These publications are also relevant as further information sources:

- *DB2 UDB for OS/390 Version 6 SQL Reference*, SC26-9014
- *DB2 UDB for OS/390 Version 6 Administration Guide*, SC26-9003
- *DB2 UDB for OS/390 Version 6 Installation Guide*, GC26-9008
- *DB2 UDB for OS/390 Version 6 Command Reference*, SC26-9006
- *DB2 UDB for OS/390 Version 6 Utility Guide and Reference*, SC26-9015
- "*DB2 UDB for OS/390 Version 6 ODBC Guide and Reference*, SC26-9005
- *DB2 UDB for OS/390 Version 6 Messages and Codes*, GC26-9011
- *DB2 UDB for OS/390 Version 6 Application Programming and SQL Guide*, SC26-9004
- *OS/390 V2R6.0 C/C++ User's Guide*, SC09-2361-03
- *AIX Version 4.2 Installation Guide*, SC23-1924
- *AIX Version 4 System Management Guide: Operating System and Devices*, SC23-2525
- *AIX Version 4.2 Network Installation Management Guide and Reference*, SC23-1926
- *AIX Version 4 System User's Guide: Communications and Networks*, SC23-2545
- *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*, SC23-2533
- *ORACLE7 The Complete Reference*, G. Koch and K. Loney, Oracle Press, ISBN 0-07-882285-8
- *Oracle7 Release 7.3 for AIX Installation Guide*, A43771-1
- *Oracle7 Programmer's Guide to the Pro*C Precompiler,* A21021-2
- *Understanding SQL*Net Release 2.3*, A42484-1
- *OS/390 eNetwork Communications Server: IP Configuration*, SC31-8513
- *OS/390 V2R7.0 MVS Planning: Workload Management*, GC28-1761-08
- *OS/390 V2R7.0 MVS Workload Management Service*s, GC28-1773-06
- *OS/390 V2R7.0 TSO/E User's Guide*, SC28-1968-01
- *OS/390 V2R7.0 TSO/E Command Reference*, SC28-1969-02
- *OS/390 V2R7.0 Planning for Installation,* GC28-1726-06
- *OS/390 eNetwork Communications Server: IP Configuration*, SC31-8513

F.4 Web sites

These web sites provide further up-to-date information sources:

- IBM Home Page
  - http://www.ibm.com/
- ITSO Home Page
  - http://www.redbooks.ibm.com/
- DB2 for OS/390 Home Page
  - http://www.software.ibm.com/data/db2/os390/
- DB2 Family
  - http://www.software.ibm.com/data/db2
- DB2 Family Performance
  - http://www.software.ibm.com/data/db2/performance

How to get ITSO redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM redbooks from the redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM redbook fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of abbreviations

AIX	Advanced Interactive eXecutive from IBM
APAR	authorized program analysis report
ARM	automatic restart manager
ASCII	American National Standard Code for Information Interchange
BLOB	binary large objects
CCSID	coded character set identifier
CEC	central electronics complex
CD	compact disk
CF	coupling facility
CFRM	coupling facility resource management
CLI	call level interface
CPU	central processing unit
CSA	common storage area
DASD	direct access storage device
DB2 PM	DB2 performance monitor
DBAT	database access thread
DBD	database descriptor
DBRM	database request module
DCL	data control language
DDCS	distributed database connection services
DDF	distributed data facility
DDL	data definition language
DML	data manipulation language
DNS	domain name server
DRDA	distributed relational database architecture
EA	extended addressability
EBCDIC	extended binary coded decimal interchange code
ECS	enhanced catalog sharing
ECSA	extended common storage area
EDM	environment descriptor management
ERP	enterprise resource planning
ESA	Enterprise Systems Architecture
GB	gigabyte (1,073,741,824 bytes)
GUI	graphical user interface
IBM	International Business Machines Corporation
ICF	integrated coupling facility
ICMF	internal coupling migration facility
IFCID	instrumentation facility component identifier
IFI	instrumentation facility interface
IRLM	internal resource lock manager
ISPF	interactive system productivity facility
I/O	input/output
ITSO	International Technical Support Organization
JDBC	Java Database Connectivity
JFS	journaled file systems
KB	kilobyte (1,024 bytes)
LPAR	logically partitioned mode
LOB	large object
LRSN	log record sequence number
LVM	logical volume manager
MB	megabyte (1,048,576 bytes)
OBD	object descriptor in DBD
ODBC	Open Data Base Connectivity
OS/390	Operating System/390
PDS	partioned data set
PTF	program temporary fix
QMF	Query Management Facility
RACF	Resource Access Control Facility
RBA	relative byte address
RID	record identifier
RS	read stability
RR	repeatable read
SMIT	System Management Interface Tool

SPB	DB2 Stored Procedure Builder
SQLJ	Structured Query Language in Java
Sysplex	system complex
TCP/IP	Transmission Control Protocol/Internet Protocol
TSO	time-sharing option
UDB	universal database
UDF	user-defined function
UDT	user-defined data type
URL	uniform resource locator
WLM	workload manager

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

ITSO redbook evaluation

Converting from Oracle AIX to DB2 for OS/390
SG24-5478-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to `redbook@us.ibm.com`

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

SG24-5478-00
Printed in the U.S.A.