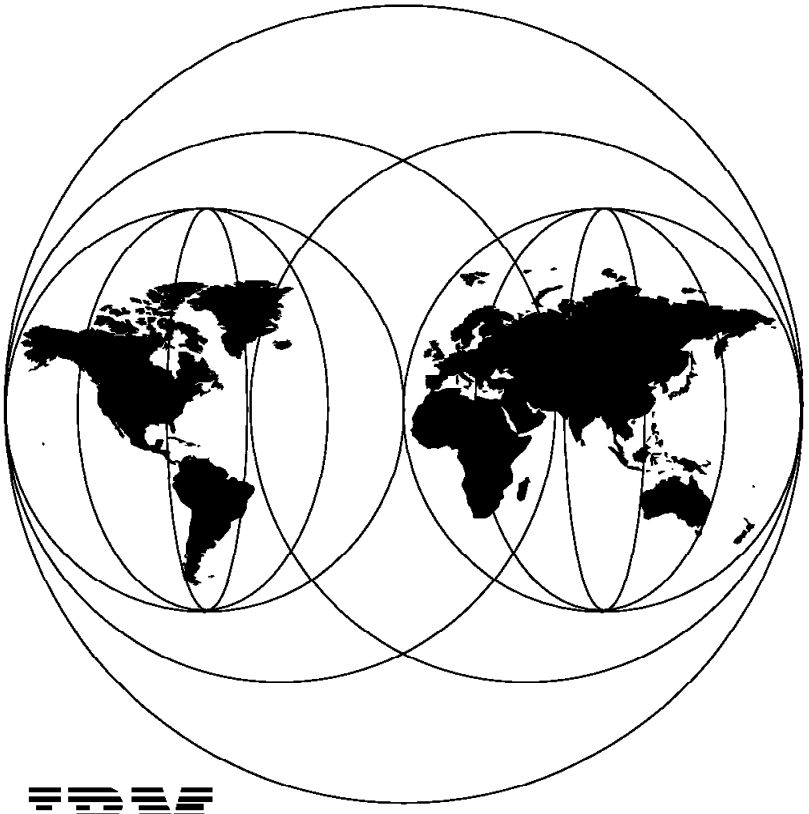
International Technical Support Organization

SG24-2567-00

DATABASE 2 for AIX Conversion Guide Oracle 7.1 to DB2 Version 2

August 1995



IBM

International Technical Support Organization
Austin Center

International Technical Support Organization

SG24-2567-00

DATABASE 2 for AIX Conversion Guide Oracle 7.1 to DB2 Version 2

August 1995

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xiii.

First Edition (August 1995)

This edition applies to Version 2 , Release 1 of DB2, for use with the AIX Operating System.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. JN9B Building 821 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

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.

Abstract

This document is unique in its detailed coverage of the conversion process from Oracle 7 to DATABASE 2 for AIX Version 2. It focuses on the technical considerations and methodologies involved in performing a database conversion. Information about the different data types and the implementation of features, such as Stored Procedures and Triggers, is provided. Application programming and conversion considerations are discussed along with the differences in features and functionality of the two products.

This document was written for people involved in the planning and implementation of a database conversion. Some knowledge of AIX and relational databases is assumed.

(178 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help IBM system engineers and their customers perform a database conversion from Oracle 7 to DATABASE 2 for AIX Version 2. The information in this publication is not intended as the specification of any programming interfaces that are provided by DATABASE 2 for AIX Version 2. See the PUBLICATIONS section of the IBM Programming Announcement for DATABASE 2 for AIX Version 2 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. 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.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ADSTAR	AIX
DATABASE 2	DB2
DRDA	IBM
MVS/ESA	OS/2
OS/400	

The following terms are trademarks of other companies:

Windows is a trademark of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

C + +	American Telephone and Telegraph Company, Inc.
IPX	Novell, Inc.
Lotus 1-2-3	Lotus Development Corporation
Microsoft	Microsoft Corporation
Oracle	Oracle Corporation
X/Open	X/Open Company Limited
Micro Focus	Micro Focus Limited
SQL*NET	Oracle Corporation
Pro*C	Oracle Corporation

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to assist in the conversion from an Oracle 7 relational database environment to a DATABASE 2 for AIX Version 2 relational database environment. It contains a description of the conversion process and suggestions on how the mapping of database features may be accomplished.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Conversion Overview”

  This chapter covers the general concepts involved in performing a conversion in a relational database environment.

- Chapter 2, “Packaging and Installation”

  This chapter describes the packaging of DATABASE 2 for AIX Version 2 products and how to map from the Oracle 7 product set to the DATABASE 2 for AIX Version 2 product set.

- Chapter 3, “Relational Database Model”

  This chapter discusses the relational database models in both DATABASE 2 for AIX Version 2 and Oracle 7. Both the structure of the database and the process model are covered.

- Chapter 4, “Storage”

  This chapter looks at how data is stored in the different environments. It covers the physical storage model, the logical storage model and the database storage model.

- Chapter 5, “Data Types”

  This chapter looks at the differences in the data types of Oracle 7 and DATABASE 2 for AIX Version 2. A variety of methods for handling the differences are also discussed.

- Chapter 6, “Database Schema”

  This chapter covers the different schema objects in Oracle 7 and DATABASE 2 for AIX Version 2. Objects ranging from aliases to tablespaces are covered, and how they map between the two environments is discussed.

- Chapter 7, “SQL Language Elements”

  This chapter discusses the differences in the SQL language elements. These include functions, SQL operations and syntax, reserved words, and special registers.

- Chapter 8, “Database Security”

  This chapter discusses the different security issues that need to be covered when converting from the Oracle 7 environment to a DATABASE 2 for AIX Version 2 environment.

- Chapter 9, “Applications”

  This chapter discusses the application-development process and how it differs between the two database environments.

- Chapter 10, “Backup and Restore”

  This chapter looks at the backup/restore process, and how to implement a backup/restore strategy on the DATABASE 2 for AIX Version 2 environment.

- Chapter 11, “Performing a Conversion”

  This chapter guides you through an actual conversion process. Looking at the different operations involved, it provides scripts that may help you in the conversion process.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document. They are available as online documents with the DATABASE 2 for AIX Version 2 products.

- *Information and Concepts Guide*, S20H-4664
- *Administration Guide*, S20H-4580
- *Database System Monitor Guide and Reference*, S20H-4871
- *Command Reference*, S20H-4645
- *API Reference*, S20H-4984
- *SQL Reference*, S20H-4665
- *Application Programming Guide*, S20H-4643
- *Call Level Interface Guide and Reference*, S20H-4644

International Technical Support Organization Publications

- *Planning for Conversion to the DB2 Family: Methodology and Practice*, GG24-4445

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office. For guidance on ordering, send a PROFS note to BOOKSHOP at DKIBMVM1 or E-mail to bookshop@dk.ibm.com.

Customers may order hardcopy ITSO books individually or in customer sets, called BOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on "The ITSO World Wide Web home page." To access the ITSO Web pages, point your Web browser to the following URL:

```
http://www.redbooks.ibm.com/redbooks
```

IBM employees may access LIST3820s of redbooks as well. The intenal Redbooks home page may be found at the following URL:

```
http://w3.itsc.pok.ibm.com/redbooks/redbooks.html
```

Acknowledgments

This project was designed and managed by:

Frank Rusconi
International Technical Support Organization, Austin Center

The authors of this document are:

Jean-Christophe Brun
IBM France

Cinzia De Persio
IBM Italy

Peter A Wood
IBM England

This publication is the result of a residency conducted at the International Technical Support Organization, Austin Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Marcus Brewer, Editor
International Technical Support Organization, Austin Center

Bernhard Schiefer, Reviewer
IBM Canada

Richard Swagerman, Reviewer
IBM Canada

Frank Pellow, Reviewer
IBM Canada

Chapter 1. Conversion Overview

This chapter looks at the conversion process and the different steps involved in the planning and execution of a successful conversion. For further information, refer to *Planning for Conversion to the DB2 Family: Methodology and Practice.* The topics covered are:

- Strategies Available
- Conversion Methodologies
- Planning the Conversion
- Conversion Considerations

1.1 Strategy and Conversion Methodologies

In performing a conversion, you need to decide on the following.

1. The overall strategy

   This is the general approach to the question of moving to DB2. Is speed the most important parameter, or perhaps it's the method involving the least amount of risk?

2. The starting point

   What is the first application to be moved to DB2? Is it just part of a total application? How will coexistence be handled?

1.1.1 Strategy Definition

There are several strategies that could be used to convert the system, each may be better suited to different environments. Table 1 on page 2 gives a summary of the strategies available and the advantages and disadvantages of each.

1.1.1.1 Big Bang

This is where all applications and data are moved to DB2 and go live simultaneously. This is useful when speed of conversion is the overriding consideration; however, this strategy has the highest risk.

1.1.1.2 Piece by Piece

This is where definable applications or pieces of them and the data that they use are migrated to DB2 one at a time. This strategy implies that all applications and data are eventually going to be converted to DB2. No enhancements are made to the application during its conversion, and future enhancements are made in DB2. This strategy is indicated when the primary considerations are to reduce risk, keep track of projects and gain experience as the work proceeds.

1.1.1.3 Tight Coexistence

This strategy means that the old systems are kept and new systems are developed with new enhancements for using data in DB2. As systems come up for redevelopment, they are redeveloped using DB2.

1.1.1.4 Loose Coexistence

This strategy usually involves a management information system (MIS) or reporting system. A copy of the data is loaded into DB2 and accessed using report programs running on DB2. This strategy means some of the data is kept multiple times, but operations continue as normal. Copies of the data can be used as part of conversion, and they can be kept up-to-date by asynchronous update or periodic copy.

1.1.1.5 Combinations

A combination of strategies may be the best solution.

One option is to set up a read-only MIS system first to handle reporting, convert a few existing report programs to use the MIS and provide extra reporting through DB2.

Where the Big Bang is not appropriate, it may be necessary to migrate some applications piece by piece and then follow up with a series of little bangs where two or three applications are migrated together with their common data.

It may be that some applications are in good shape, but others are in poor shape and need rewriting. This may mean different methods will be employed for different applications, but it does not alter the basic strategies.

The following table summarizes the main advantages and disadvantages of each strategy.

Table 1. Summary of Strategy Characteristics

Strategy	Advantages	Disadvantages
Big Bang	Fastest for total conversion	High risk
	No coexistence problems	Long time before first benefits
	No duplicated work force	Complex change management
Piece by Piece	Low risk	Long project
	Fast way to get some benefits of DB2	Need to handle coexistence
	Workforce adjusts over time	Dual platforms for some time
Tight coexistence	Enhancements added	Need very good dual update mechanism
	CASE tools possible	Need to open programs on multiple occasions
	Each conversion separately justified	Dual platforms indefinitely
Loose Coexistence	Management information greatly enhanced	Does not fix operational problems
	Low risk	Hard to get up-to-the-minute information
	Few coexistence problems	

1.1.2 Conversion Methodologies

There are several methodologies that could be considered, any of which could be correct in different circumstances. Once a conversion methodology has been selected, it will then set the way for data design: programs, testing, tools, and any data cleaning.

Table 2 on page 5 gives a summary of the conversion methodologies available and their advantages and disadvantages.

1.1.2.1 Translation

Translation occurs when the application Database Management System (DBMS) calls are translated one-for-one into DB2 SQL calls. When applied to data, translation means that the data is also copied one-for-one into DB2 with no allowances made for the advantage of the large number of DB2 data types.

Translation means modifying the programs to access the new database and using the old data layout for the data; it is a one-for-one line translation of the data access language in the source code. Translation tries to make no changes to the business or data logic parts of the programs and no structural changes to the data layout. Where speed of conversion is the highest priority, translation allows for fast migration.

1.1.2.2 Transparency

With transparency, the data is migrated to DB2, and a special program is written to intercept calls to the old DBMS. This program will translate the calls into DB2 SQL transparently and routes the call to DB2. The program remains unchanged and can be executed with the data in either database system. Transparency enables all the data to be moved to DB2 and the programs to be rewritten or changed one at a time. It also offers a migration path for businesses with very large databases that require very high availability and administrators who do not have time to unload and reload the data.

DataJoiner is a product from IBM that can help with this method. It allows transparent access to data wherever it may reside with the application unaware of where the data is coming from. By using DataJoiner, applications can access the DB2 and Oracle databases at the same time, thus allowing you to join a DB2 table with an Oracle table in one SQL statement. DataJoiner can also be used to assist in some of the other methodologies which use a coexistence strategy. It can also be used as a tool to move the data from Oracle into DB2.

1.1.2.3 Re-engineering

Re-engineering allows us to alter the data and programs to be compatible with good DB2 design without having to rethink the whole design.

The data is adapted to a new data model, and the program data calls are adapted to fit the new target data model. For the applications, only the database calls and data logic parts of the program are changed with limited alterations to the DB logic consistent with the new design. Re-engineering would have only minimal impact on the business-logic part of the program.

This method is good for testing for equivalent function and for future enhancements. It should perform well, but will take longer than translation.

1.1.2.4 Reverse Engineering

Reverse engineering is a form of re-engineering that involves capturing the old design into models, modifying them, and generating new programs and a new DBMS design. Reverse engineering usually means using tools to recover the design of the original data into a reconstructed data model and recover the application into a new process model. These models can then be modified to improve the design. The new improved models are then used to generate (forward engineer) DB2 Data Definition Language (DDL) for the database and new code for the applications.

Reverse engineering is a very valid way to improve applications and data, but to be really effective, good tools are needed.

Tools such as *AMC*DESIGNER* are available to help automate this process.

1.1.2.5 Redevelopment

Here, the whole application is redeveloped according to user requirements. In the case of data, it means that the data model is rethought from scratch using requirements from the users. Redevelopment can use traditional methods or the new CASE tools. If CASE tools are used to model the process, future enhancements mean only changing the model and regenerating code.

Choosing this course of action requires the purchase of a suitable package since this is equivalent to writing an application suite from scratch.

1.1.2.6 Corporate Model

This refers to the generation of corporate-wide models. Instead of just one application suite, the entire business can be modelled into a corporate business data model and a corporate business process model. The DBMS design and code may be generated from the models and any changes or enhancements needed are reflected in the model′s original and newly generated code. Each application suite may be rewritten or regenerated one by one to access a new corporate-wide database.

Although costly in time and effort, it is a method that potentially provides the greatest benefits. It allows systems to be brought together using an integrated corporate database.

1.1.2.7 Combinations

To some extent, the methods can be mixed. It is possible to use one method for data and a different method for the applications. There is the possibility of using one method and switching to another. It may be that time is of the essence and therefore a quick translation just to move the DBMS is required. As soon as that conversion is complete, sections of the DBMS and relevant programs can be re-engineered.

The following table summarizes the main advantages and disadvantages of each conversion methodology.

Table 2. Summary of Conversion Methodologies

Methodology	Advantages	Disadvantages
Translation	Easiest Method	Few advantages of DB2
	Easy for tools	Design inflexible for future
	Possible to use in two-stage approach	
Transparency	Data may be restructured	Performance maybe a problem
	Applications can be rewritten later	Module difficult to write
	Low risk	Conversion takes much longer
Re-engineering	Obtain advantages of DB2	Relatively longer time
	Allows limited redesign	
Reverse Engineering	Design may be optimized for performance	Tools do not handle all situations
	Later enhancements easier	Less efficient code from tools
Redevelopment	Known process	Much longer development time
	Future maintenance easier	Existing investment lost
Corporate Models	Business modeled as a whole	Large upfront investment
	Less redundant code	Existing investment lost
	Case tools available	Need to manage changes

1.2 Planning the Conversion

A DBMS conversion can sound simple—just collect all the programs and change the DBMS calls from the old data language to DB2 SQL calls. However, consider these questions:

- Which application will be converted first?
- What is known about the makeup of the programs?
  - How many are there?
  - Are they all the same language?
  - Do they all make DBMS calls?
  - How good is the current documentation?
  - Do all programs need converting?
- Are the applications independent?
- If the data is moved to DB2 for use by the new system, how will updates be synchronized?
- Can a single application system be split so that it can be moved a piece at a time?
- How much time is allocated for the conversion process?
- How good is the data model of the existing system?
- How will the results be tested?

These are just some of the considerations that must be fully addressed to have a successful conversion. A decision must also be made as to which strategy and conversion methodologies to use.

These issues are addressed via the Three Stage Approach as shown in Figure 1 on page 7

1. Stage One

   Before any conversion can be contemplated, there is a need to take stock of the current systems. This stage takes an overview (portfolio analysis) of the system and considers what options will fulfill most requirements to produce a cost effective conversion strategy.

2. Stage Two

   The second stage tests the feasibility of the output produces by stage one. It puts plans in place and tests them with a pilot conversion to see if the assumptions made are correct.

3. Stage Three

   Stage three is the main conversion. It is where the majority of the work takes place. This is implementing the plan worked out in Stage Two, then cutting the new system over to production.

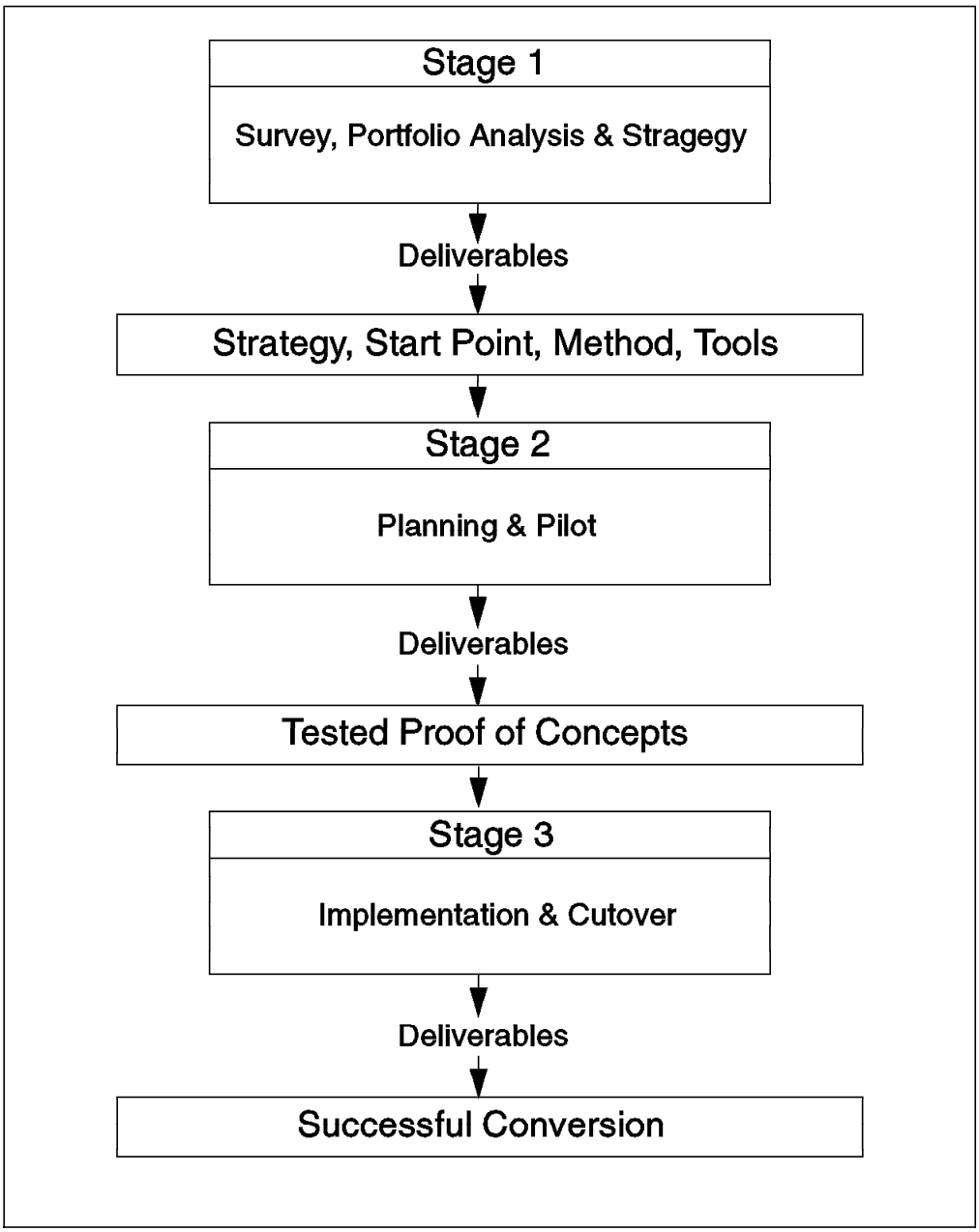


Figure 1. Three Stages of Conversion

1.2.1 Stage One: Defining the Strategy

Before any conversion can be contemplated, current systems need to be evaluated. The information obtained in stage one is essential for deciding which strategy and which conversion method to use.

1.2.1.1 Survey

This is a short task where the area of the proposed conversion is set down on one page. The purpose of this short document is to set the boundaries of the subsequent project. It sets out which system or systems are candidates for the conversion, the programming languages involved and the data sources.

1.2.1.2 Business Requirements

The most important aspect of conversion is contained here. This sets out what is expected from the conversion and is used to guide later decisions. It effectively sets down the business requirements and other criteria that must be achieved for the project to be a success.

1.2.1.3 Portfolio Analysis

The goal of the portfolio analysis is to get an idea of the state of the systems, the size of the task, the effects on data used by multiple systems and any special difficulties to account for.

1.2.1.4 Strategy Definition

There are several ways that you could go forward and many methods you could use to actually convert the system, all of which could be correct in some instances.

Now that the present position is understood, the main decisions can be made about the applications. These include:

- Whether to adopt a strategy to move everything at once (Big Bang) or to move small pieces at a time (Piece by Piece).
- Whether to move all the reporting programs
- Which application to move first
- Which programs and data to use for a pilot

1.2.1.5 Conversion Methods

With the overall strategy set, the method can then be selected and the requirements set for the data layout and DB2 features, including:

- What method of conversion will suit the programs and data best
- What to do about data cleaning
- What testing strategy to use
- What enhancements should be made, and when

1.2.1.6 Deliverables

The deliverables of this first stage are:

- A description of the business benefits expected
- An understanding of the migration process
- An overall strategy
- The place to start the migration process
- Scope of the pilot
- List of issues

The biggest factor influencing success is going into the new territory with awareness of both the pitfalls and the rewards. Working through Stage One will give an understanding of the process, a set of requirements, and the first gross estimate of the conversion task.

1.2.2 Stage Two: Testing the Concept

The objective of stage two are to put proper plans in place and testing the theories with a pilot conversion, the results of which are then used to adjust the plans for the final stage. There are a number of relatively small, but vital, tasks that need to be done:

Data layout The database design must be carefully checked and a first cut DB2 design produced.

Programs The programs need checking too. An inventory of where they are by categorizing them into Online Transaction Processing (OLTP), batch and reporting applications.

Testing The test strategy decided in stage one needs to be written, set up and tested.

Performance A performance plan needs to be established. It should identify critical processes and specify how changes are to be made.

Change Control A change control system must be put in to place.

Data Movement A plan needs to be put in place for data movement—that is, the unloading, reformatting and loading of the actual data.

Set Up DB2 If DB2 is not installed, then installation needs to take place before any pilot can run. If DB2 is already in use, then it is preferable to establish a separate system for the conversion work.

Pilot The pilot takes the previously chosen applications and runs them through the conversion process.

Review At the end of the pilot, a thorough review should be made, and the results should be fed back into the plans and adjustments made.

1.2.2.1 Deliverables

The deliverables for stage two can really be expressed as a tested proof of concept. Specifically, there should be;

- Tested proof that all the components work
- A program inventory
- A data inventory
- Old data column to new data column cross reference
- Old program name to new program name cross reference (if applicable)
- A project plan for the entire conversion:
  - An application plan showing how the changes will be tackled
  - Data movement plan
  - A performance test plan
  - Change control plan
  - Data cleaning
  - A test plan
- Database design

- Critical process list
- A data movement plan
- Documented results of the pilot
- The first few converted programs
- Results of the review

Done properly, the second stage shows that the project is viable, highlights any areas of weakness that can be corrected and provides feedback to be used in the main and final stages of the plan.

1.2.3 Stage Three: Implementation and Cutover

By this time, although there is much left to do, the rest should be straightforward. It is now a matter of making sure the plan is correctly implemented.

1.2.3.1 Implementation

This is the implementation of the plans worked out in stage two. Here the programs are converted, tested and changed, if required, via the change control plan already set up.

1.2.3.2 Data Cleaning

The accuracy of the actual data needs to be assessed for quality. Depending on the errors, this may require anything from a small to a significant investment. The most common corruption is that data is entered incorrectly in the first place. Sometimes data has been entered incompletely, and the additional data was never entered.

During DB2 Load, the utility checks the properties of the data, and it will refuse to load any records that are obviously wrong, placing them instead in a discard file. This is one method that can be used for assessing errors; however, the utility cannot detect where the wrong values have been entered.

The data cleaning strategy identified in stage one and tested in stage two should be used to verify the data and correct any errors.

1.2.3.3 Cutover

Here the data is unloaded, reformatted and loaded into DB2. The plans for the applications are bound, the libraries switched and the new system brought up.

1.2.3.4 Testing

Testing is very likely to take 50 percent or more of the time taken to convert the DBMS. During conversion, testing differs from normal application development in these ways;

1. There is a need to test every function and every part of the code to ensure that it all still performs in the same way.
2. The tests must be repeatable and repeated to ensure that errors are not reintroduced.
3. The tests must ensure that the program still functions the same.
4. Automated test tools can shorten the time taken for testing by providing repeatable automatic comparison and regression testing.

5. Stress testing needs to be a part of the testing to ensure that the whole application will function well and to check that things, such as deadlocks, have not been introduced.

1.2.3.5 Fallback Plan

In case of unforeseen problems, a fallback plan needs to be set up. This should be business-as-usual as customers are used to changing levels of software, both from themselves and from software vendors.

1.2.3.6 Consolidation

The period of post-live consolidation needs to be defined. The system needs to run for a month or so before any new enhancements are added. Further, when the conversion is declared complete, do not delete the old system; archive it.

1.2.3.7 Deliverables

This is, of course, the new system working as planned with DB2.

1.3 Conversion Considerations

This section highlights some of the considerations found during the many conversions to DB2 that have already been completed. These are not unique to a particular environment and may very depending on your specific environment; therefore any, all or none of these considerations may apply.

- One percent of source code is lost.

  When it comes to actually converting every program, some source code cannot be found.

- Ten percent of the source code does not match the production object code.
- Sixty percent of the program inventory needs to be converted.

  This relates to redundant programs, unused programs, reporting programs, programs without any DBMS calls, and those that need rewriting anyway.

- Testing and fixing will take over 50 percent, and may take up to 80 percent, of the project time.
- Sixty percent of the programs problems are unrelated to conversion.

  Problems due to unsuitable conversion do occur, but the majority of issues that surface are due to:

  - Bad source code
  - Known bug in original code
  - Latent bug that is found due to new environment
  - Latent bug that is found due to more comprehensive testing

Chapter 2. Packaging and Installation

This chapter discusses the Oracle product packaging and relates it to the product packaging used by DB2. The chapter has been written for the user who is starting a conversion process and needs help determining the following:

How to map from Oracle products to DB2 products

The function of each DB2 module

How to choose and install the required DB2 components

The correct installation sequence to follow

Disk storage requirements for the DB2 product components

The directory structure after installation

The licensing management for each product

The chapter consists of the following topics:

- Product packaging
- Installation
- Licensing overview

2.1 Oracle Products and Packaging

It is possible to group Oracle products into the following six categories:

- Database Engine
- Database Administration Products
- Application Development Tools
- Case & Designer Tools
- Communications Support Tools
- General Tools

Each of these categories contain a number of products that perform a specific function. Figure 2 on page 14 groups the Oracle products into these categories and shows how they interact with each other.

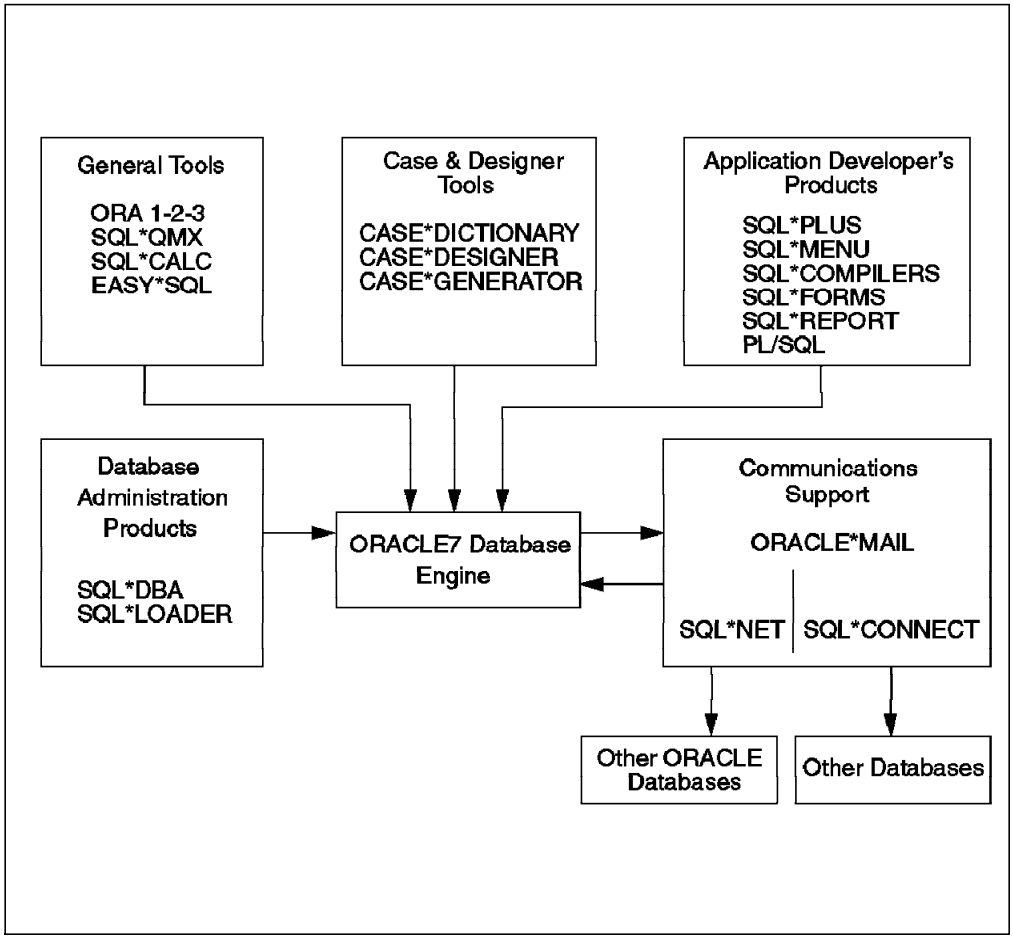


Figure 2. Oracle Product Categories

The function of each of these Oracle products is described in the following list.

Oracle7 Server Database engine

SQL*DBA Base command line processor for database administration

SQL*Loader Loader tool for non-ORACLE data

SQL*Plus Command line processor for SQL statements

SQL*FORMS Forms generator

SQL*Menu Menu generator

SQL*Report Writer/SQL*Report
Report writer and generator/graphic report writer

Pro*Ada Support for Ada compiler

Pro*C Support for C compiler

Pro*COBOL Support for COBOL compiler

Pro*FORTRAN Support for FORTRAN compiler

PL/SQL Procedural Language support

CASE*DICTIONARY Tool for maintenance of database information and objects

CASE*DESIGNER Graphic interface for the Case*Dictionary data

CASE*GENERATOR	Tool for generation of simple forms using dictionary information
SQL*NET	Features and tools for the connectivity
SQL*CONNECT	Access to non-Oracle databases
Oracle*Mail	Feature and tools for office mail
ORA 1-2-3	Electronic foil, like Lotus 1-2-3
SQL*QMX	Tool for easy SQL reports
SQL*Calc	Electronic foil with Oracle layout
EASY*SQL	Easy command line processor for SQL statements

2.2 DB2 Products and Packaging

The DB2 family of products is composed of product kits and components. The product kits can be purchased with a range of licensing options. The kits available include:

- IBM DATABASE 2 for AIX Version 2.1 Server
- IBM DATABASE 2 for AIX Version 2.1 Single-User
- IBM DATABASE 2 Software Developer's Kit for AIX Version 2.1
- IBM DDCS for AIX Version 2.3 Multi-User Gateway
- DB2 Client Application Enabler for AIX
- DB2 Product Messages
- DB2 Product Library

Components are the features included in a product kit. They can be considered as installable options for the related DB2 products. The product kits are made up of the various components to suit different environments, and they provide various levels of functionality. Table 3 on page 16 contains a cross reference of the DB2 product kits and the components they include.

Table 3 (Page 1 of 2). DB2 Product Kit and Component Cross Reference	
IBM DATABASE 2 for AIX Version 2.1 Server	
db2_02_01.client	Client Application Enabler
db2_02_01.clp	Command Line Processor
db2_02_01.db2.rte	DB2 executables
db2_02_01.db2.misc	DB2 Utilities and Examples
db2_02_01.dd	DB2 Database Director
db2_02_01.cs.rte	DB2 Communications Support - Base with TCP/IP
db2_02_01.cs.sna	DB2 Communications Support - SNA
db2_02_01.cs.drda	DB2 Communications Support - DRDA Application Server
db2_02_01.cs.ipx	DB2 Communications Support - IPX
IBM DATABASE 2 for AIX Version 2.1 Single-User	
db2_02_01.client	Client Application Enabler
db2_02_01.clp	Command Line Processor
db2_02_01.db2.rte	DB2 executables
db2_02_01.db2.misc	DB2 utilities and examples
db2_02_01.sdk.c	DB2 SDK C include files and sample programs
db2_02_01.sdk.fortran	DB2 SDK FORTRAN include files and sample programs
db2_02_01.sdk.cobol	DB2 SDK COBOL include files and sample programs
db2_02_01.sdk.cli	DB2 SDK Call Level Interface samples
db2_02_01.sdk.misc	DB2 SDK utilities and samples
db2_02_01.dd	DB2 Database Director
db2_02_01.ve	DB2 Visual Explain
IBM DATABASE 2 Software Developer's Kit for AIX Version 2.1	
db2_02_01.client	Client Application Enabler
db2_02_01.clp	Command Line Processor
db2_02_01.sdk.c	DB2 SDK C include files and sample programs
db2_02_01.sdk.fortran	DB2 SDK FORTRAN include files and sample programs
db2_02_01.sdk.cobol	DB2 SDL COBOL include files and sample programs
db2_02_01.sdk.cli	DB2 SDK Call Level Interface samples
db2_02_01.sdk.misc	DB2 SDK utilities and samples
db2_02_01.dd	DB2 Database Director
db2_02_01.ve	DB2 Visual Explain
IBM DDCS for AIX Version 2.3 Multi-User Gateway	
db2_02_01.client	Client Application Enabler
db2_02_01.clp	Command Line Processor
db2_02_01.db2.rte	DB2 executables
db2_02_01.cs.rte	DB2 Communications Support - Base with TCP/IP
db2_02_01.cs.sna	DB2 Communications Support - SNA
db2_02_01.cs.drda	DB2 Communications Support - DRDA Application Server
db2_02_01.cs.ipx	DB2 Communications Support - IPX
db2_02_01.ddcs	Distributed Database Connection Services

Table 3 (Page 2 of 2). DB2 Product Kit and Component Cross Reference	
db2_02_01.dd	DB2 Database Director
DB2 Client Application Enabler for AIX	
db2_02_01.client	Client Application Enabler
DB2 Product Messages - %L (%L represents locale)	
db2_02_01.msg.%L.client	DB2 Product Messages
db2_02_01.msg.%L.dd	DB2 Database Director Messages and Help
DB2 Product Library	
db2_02_01.doc.%L.pscript	DB2 manuals in postscript
db2_02_01.doc.%L.ipfx	DB2 manuals in IPF format
IPF/X Viewer	
ipfx.runtime	IBM Information Presentation Facility
IPF/X National Language Support Messages and Resources	
ipfx.nls.%L	Messages and Resources - %L

Each of the DB2 product components provide a defined set of functions. This allows us to select only the components required by your environment. By doing this, we are able to keep the size and cost of the DB2 products to a minimum. The DB2 components are listed below along with a description of the function they provide.

IBM DB2 for AIX Server

The server engine is a full-function, relational database management system that includes SQL query optimization, based on actual database usage. It provides support for user defined functions and types, triggers, stored procedures, constraints, Large Objects (LOBS), and recursive SQL. DATABASE 2 for AIX Version 2 also provides support for a distributed unit of work. The server is also able act as a client to other DB2 servers.

IBM DB2 for AIX Single-User

The single-user version of the engine provides the same functionality as the server, but only for local applications. There is no network server capability built into the product.

IBM Distributed Database Connection Services (DDCS) for AIX

DDCS provides the ability to access host databases running on systems such as DB2 for MVS/ESA, DB2 for VSE and VM, DB2 for OS/400 and most systems that support the Distributed Relational Database Architecture (DRDA) protocol. There are two versions of DDCS. The first is a single-user version; this allows only local applications to access the host databases. The second, multi-user gateway allows both local and remote clients to access the host.

DB2 Client Application Enabler

The remote clients are able to run in a number of environments, including OS/2, DOS, Windows, and several UNIX-based environments. The DB2 Client Application Enabler component is build into each of the server products. This

component allows applications to access any DB2 server over a number of supported communications protocols.

DB2 Command Line Processor

The command line processor allows you to prototype SQL statements. It also provides the capability to backup or restore a database, configure database or database manager parameters, and a number of other administrative tasks.

IBM DB2 Software Developer′s Kit for AIX

The Software Developer′s Kit (SDK) is a collection of tools that meet the needs of an application developer. Support for the creation of character-based, multimedia or object-oriented applications is provided. The SDK allows you to develop applications using embedded SQL, application programming interfaces (APIs) or a call-level interface. The call-level interface is compatible with Microsoft′s Open Database Connectivity (ODBC).

DB2 Software Developer′s Kit for C

The DB2 Software Developer′s Kit for C provides the necessary include files and some sample programs for developing applications using the C programming language. The actual compiler is not included with the component.

DB2 Software Developer′s Kit for FORTRAN

The DB2 Software Developer′s Kit for FORTRAN provides the necessary include files and some sample programs for developing applications using the FORTRAN programming language. The actual compiler is not included with the component.

DB2 Software Developer′s Kit for COBOL

The DB2 Software Developer′s Kit for COBOL provides the necessary include files and some sample programs for developing COBOL applications. The actual compiler is not included with the component.

DB2 Software Developer′s Kit Call-Level Interface Samples

The Call-Level Interface (CLI) is based on the Microsoft Open Database Connectivity specification and the X/Open Call-Level Interface specification. It provides a development environment for creating DB2 CLI applications. The DB2 Client Application Enablers provide run-time support for executing these applications.

DB2 executables

The DB2 executable component is the run-time environment for DB2. The run-time environment comes with the DB2 server products.

DB2 Communications Support-Base with TCP/IP

This component is required if remote clients are going to require access to a DB2 server. The DB2 Communications Support is installed on the server platform, and allows clients to connect using the TCP/IP protocol.

DB2 Communications Support - SNA

The SNA communications support allows clients to connect using the SNA protocol.

DB2 Communications Support - DRDA Application Server

This product allows DB2 to perform server functions across a network using the DRDA protocol.

DB2 Communications Support - IPX

Provides support for Netware's IPX protocol.

DB2 Database Director

Provides an easy-to-use graphical interface that will display database objects, such as databases, tables and packages, and the relationships between them.

DB2 Visual Explain

The Visual Explain tools can be used for analyzing and tuning SQL statements. They allow developers to view the access plan chosen by the database manager's optimizer for a given SQL statement. They also provide the ability to tune the SQL statements for better performance and model the impact of environment changes on SQL statements.

DB2 Product Library (INF/POSTSCRIPT)

Provides the DB2 product manuals in either Postscript or INF format. Manuals in INF format can be viewed by using the IPF/X viewer.

Table 4 provides a guideline for matching Oracle products with the corresponding DB2 product or component.

Table 4 (Page 1 of 2). DB2 and Oracle Product Cross Reference

DB2	**Oracle**
Database 2 Engine	Oracle7 Server
DB2 Utilities and Samples	Oracle7 Server
Product Family Messages	Oracle7 Server
Database Administrator's Messages	no single component
DB2 Visual Explain	no product
DB2 Database Director	SQL*DBA
DB2 Client Application Enabler	SQL*NET
DB2 Command Line Processor	SQL*PLUS
DB2 Software Developer's Kit Executable	no single component
DB2 C Language include Files and Samples	PRO*C
DB2 FORTRAN Language include Files and Sample Programs	PRO*FORTRAN
DB2 COBOL Language	PRO*COBOL
DB2 Call-Level Interface Samples	PL/SQL
DB2 SDK Utilities and Samples	no single component

Table 4 (Page 2 of 2). DB2 and Oracle Product Cross Reference	
DB2	**Oracle**
DB2 Communication Support - Remote Client Support	SQL*NET
DB2 Communication Support - SNA Support	SQL*NET
DB2 Communication Support - DRDA Application Server	SQL*NET
DB2 Communication Support - IPX Support	SQL*NET
DB2 Distributed Database Connection Services	Oracle Transparent Gateway for IBM DRDA
DB2 Product Library - manuals in INF format	Oracle*Book
DB2 Product Library - manuals in Postscript format	Oracle*Book
IBM Information Presentation Facility/6000 Executable	Oracle*Book

2.2.1.1 The DB2 Client/Server Environment

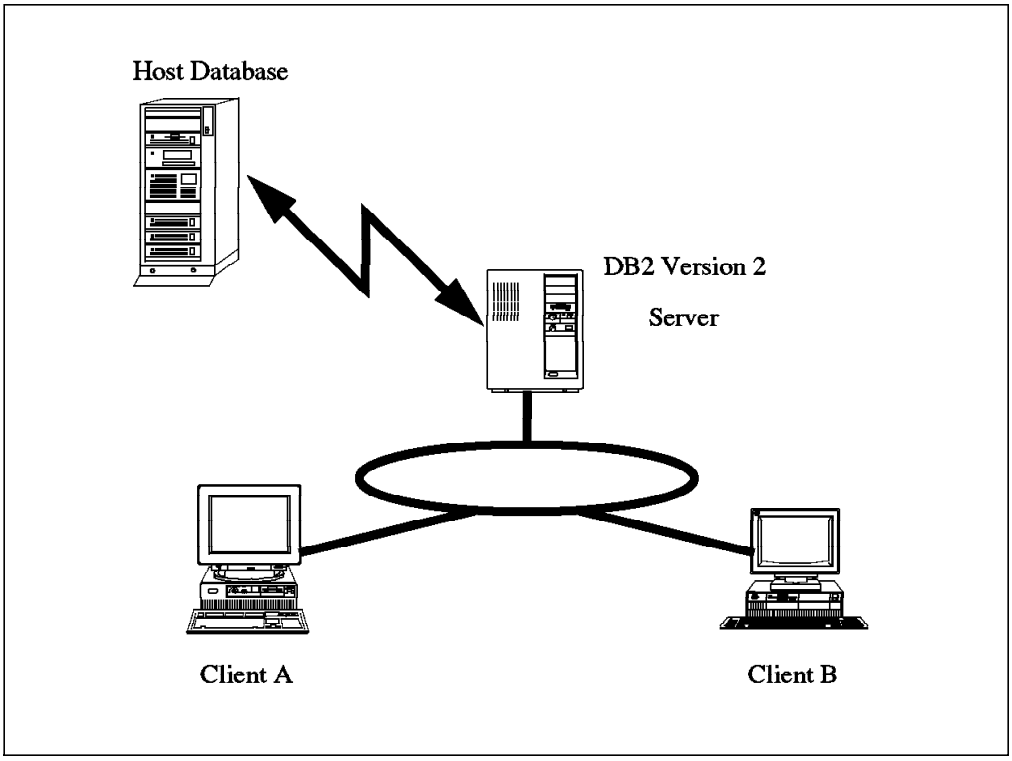


Figure 3. DB2 Client/Server Environment Example

Figure 3 shows an example of a client/server environment. If both the AIX server and the host contain databases, the data on these machines can be accessed from either client. For the example, we will assume that Client A is an AIX development machine and Client B is an OS/2 client.

The DATABASE 2 for AIX Version 2 server would need the Version 2.1 Server Kit. This would include the command line processor and the communications software that allows the clients to connect to the server. For access to the host database, the DDCS Multi-User Gateway Kit would also have to be installed.

Client A would require the Software Developers Kit. This kit includes the necessary tools for developing DB2 applications. It also includes the command line processor and the Client Application Enabler components.

Client B would only require the Client Application Enabler Kit. This kit allows and application to communicate with the DATABASE 2 for AIX Version 2 server.

2.3 DB2 Product Installation

After selecting the appropriate DB2 products, we need to install them on the machine. The following steps are required to perform a standard installation of the DB2 products. For further installation instructions, refer to the documentation provided with DATABASE 2 for AIX Version 2.

Step 1 Logon as *root* and use `installp` command or the SMIT interface to install the desired products.

Step 2 Create or assign groups and users. DB2 will use the AIX groups and user names for database security. Refer to Chapter 8, “Database Security” on page 107, or the DB2 documentation for more information on Users and Groups in DB2

Step 3 Create the instance using the `db2icrt` command. Remember, a single DB2 instance can contain multiple databases.

For information on a DB2 instance, refer to Chapter 6, “Database Schema” on page 61.

Step 4 Configure license information. This may be a nodelock file, or you may use a iFOR/LS License server. For detailed information about licensing methods, see section 2.4.2, “DB2 Licensing” on page 24.

Step 5 Configure the DB2 environment using the db2profile file. Refer to the *Installation and Operation Guide* for details on the DB2 environment variables.

Step 6 Create links for DB2 libraries using the `db2ln` command. This may be required if doing application development.

2.3.1 Directory Structure

When all the DB2 products are installed, the following directories will be created under /usr/lpp/db2_02_01/:

adm System administrator executable files

adsm ADSTAR Distributed Storage Manager files

bin Binary executable files

bnd Bind files

cfg Configuration files

dba Database Director

deinstl Files used to reject applied software

doc/%L Postscript and online books in INF format

function User-defined functions

include C and FORTRAN include files

include/cobol_mf COBOL COPY files for Micro Focus COBOL

include/cobol_a COBOL COPY files ANSI COBOL

instance Instance scripts

lib Libraries

map Map files for DDCS for AIX

misc Utilities and examples

msg/%L Message catalogs

netls iFOR/LS files

readme/%L README files

samples/c C sample programs

samples/cli	DB2 Call-Level Interface samples
samples/clp	Command line processor examples
samples/db2sampl	Sample database
samples/fortran	FORTRAN sample programs
samples/rexx	DB2 REXX samples

2.3.2 Hardware Requirements

The disk and memory requirements are going to vary depending on the products you choose to install. For a guideline on the recommended memory and disk requirements, refer to the *IBM Database 2 for AIX Planning Guide*.

The point to note with DB2 is that a single machine is capable of running multiple instances. When a new instance is created, the disk storage requirements does not increase. This is because the instance actually creates symbolic links to the product installation directories. Multiple instances are then able to share the single copy of the product code. The memory requirement will go up because each instance will have its own set of processes and shared memory areas.

Table 5. DB2 Hardware Requirements

Function	Recommended Memory (MB)	Recommended Disk (MB)
DB2 for AIX Single-User	3.2	26.4
DB2 for AIX Server	3.1	26.9
DB2 Client Application Enabler for AIX	0.15	9.2
DB2 Software Developer's Kit for AIX	0.15	13.8
DDCS for AIX Multi-User Gateway	0.10	25.6
Database Director		34
Documentation - IPF		10
Documentation - Postscript		20

As an initial guideline Table 5 provides an indication of the memory and disk requirements for different environments.

2.4 Licensing Overview

The following sections discuss the differences between ORACLE and DB2 licensing methods.

The licensing in DATABASE 2 for AIX Version 2 is different to that of Oracle 7. DB2 has the ability to use a nodelock license or the iFOR/LS product to control its licensing. Oracle uses its own licensing that is built into the Oracle products.

2.4.1 Oracle Licensing

Oracle products have two types of license tracking, both of which are checked and activated internally by the database. They are:

- **Concurrent-session licensing**

This method checks and set a limit for the number of concurrent sessions that can connect to an instance. To set this, you need to set the value, LICENSE_MAX_SESSIONS, in the database parameter file. You can also set the LICENSE_SESSIONS_WARNING parameter. Once this value is reached, an Oracle user is able to connect, but a warning message is received.

- **Named-users licensing**

  This method allows your to create a set number of users that can access the database. The number is determined by the LICENSE_MAX_USERS parameter. When this limit is reached, Oracle does not allow you to create any more Oracle users for that instance. The users referred to here are Oracle users, not operating system users.

The prices are lower for the named users license. The base license for named users licenses 1 to 8 users, and this can be increased by lots of three.

2.4.2 DB2 Licensing

The DATABASE 2 for AIX Version 2 family of products are licensed programs that operate under the control of iFOR/LS, a network licensing system produced by Gradient Technologies. This system checks and controls all the accesses to products. A license key is obtained and entered into the iFOR/LS server. The product is composed of a pre-generated iFOR/LS Product License Key which is shipped with the product on a Product License Key label. Before you can use DB2, saveSDK and DDCS, you have to enter the Pre-generated Product License Key. For detailed information on the Installation method, refer to the *"Installation and Operation Guide*.

This kind of licensing is called a *Nodelock License* A nodelock license allows the DB2 product to run on a specific machine because the license is tied to the machine's CPU. A nodelock license is required for each of the DB2 products on the machine, and it is stored in a single text file. This type of license does not require the use of a license server, and this makes it suitable for smaller environments.

2.4.3 Default Licenses

DB2 and DDCS base products allow up to five concurrent users. The number of concurrent users is defined as the maximum number of client workstations, running one or more applications, that are connected to a DATABASE 2 for AIX Version 2 server at a given point in time. If you need to support more than five users, you can acquire entitlements for additional users. These entitlements are available in configurations of 1, 5, 10, and 50 users.

By default, the DB2 Software Developer's Kit for AIX base product allows access for only one user. If you need to support more than one user, additional entitlements are available in configurations of 1, 5 and 10 users.

The DATABASE 2 for AIX Version 2 Single-User Kit includes a single developer entitlement for the DB2 Software Developer's Kit for AIX.

Chapter 3. Relational Database Model

Although Oracle 7 and DATABASE 2 for AIX Version 2 are both Relational Database Management Systems, there are some differences in the way that they are implemented. This chapter discusses these differences, and how it may effect the conversion process.

The topics covered include:

- Instance and Database Structure
- Database Creation
- Process Model
- Client / Server Model

3.1 Instance and Database Structure

The relational database structure can be divided into physical and logical parts. The physical part is determined by the operating system. It includes the files, directories and other physical storage elements. The logical part is made up of the objects that are referenced by the database. The logical elements include tables, tablespaces and other elements that make up the relational database model.

3.1.1 Oracle Database Structure

An Oracle database is generally divided into tablespaces and schema objects such as tables, views and indexes. Figure 4 shows the logical structure of a sample Oracle database.

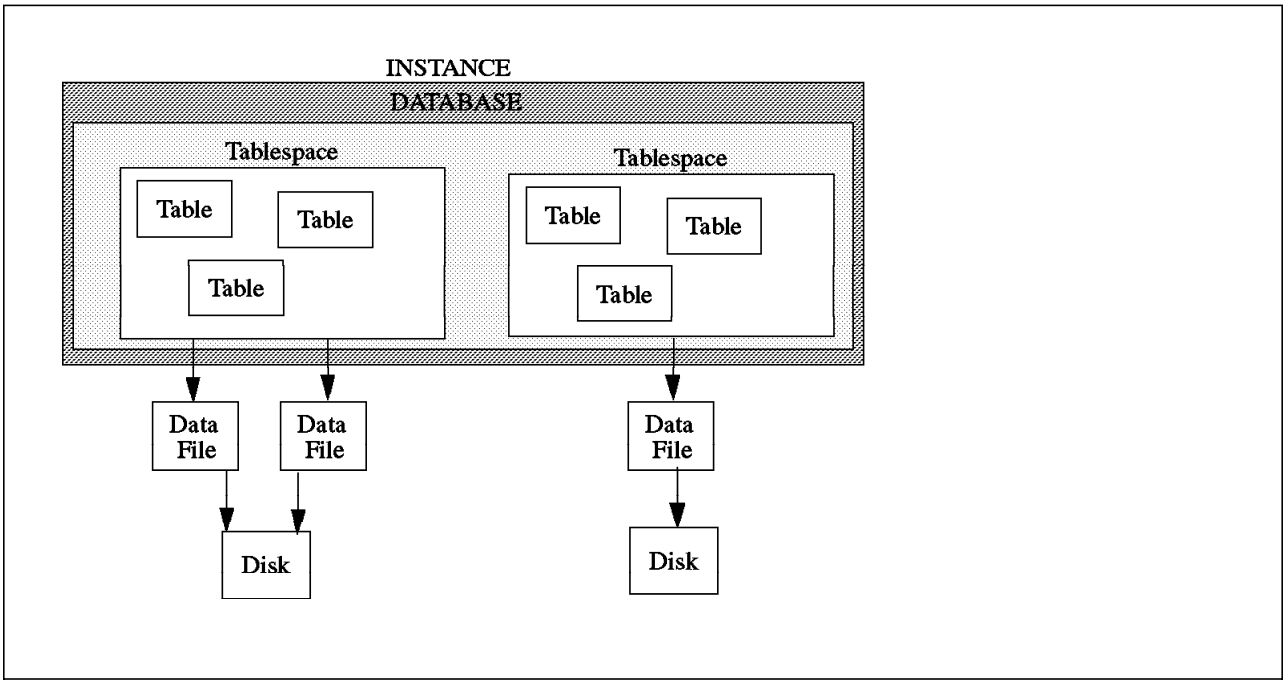


Figure 4. Oracle Sample Instance Structure

Each Oracle instance is also a unique database. The database can be divided into one, or more tablespaces and each tablespace may contain several tables. However, a table must be fully contained within a single tablespace.

3.1.2 DB2 Database Structure

In DB2 an instance can contain multiple databases. Each database may have a different logical structure by using different combinations of tablespaces. However, a tablespace cannot be shared between databases. There are two different logical tablespaces. They are System Managed Spaces (SMS) and Database Managed Spaces (DMS). Both types of tablespaces appear to be equivalent to an application running under DB2. The difference lies in the management and physical storage of the data within the tablespace itself. These differences are discussed in Chapter 4, “Storage” on page 35. Figure 5 shows a sample structure of a DB2 instance.

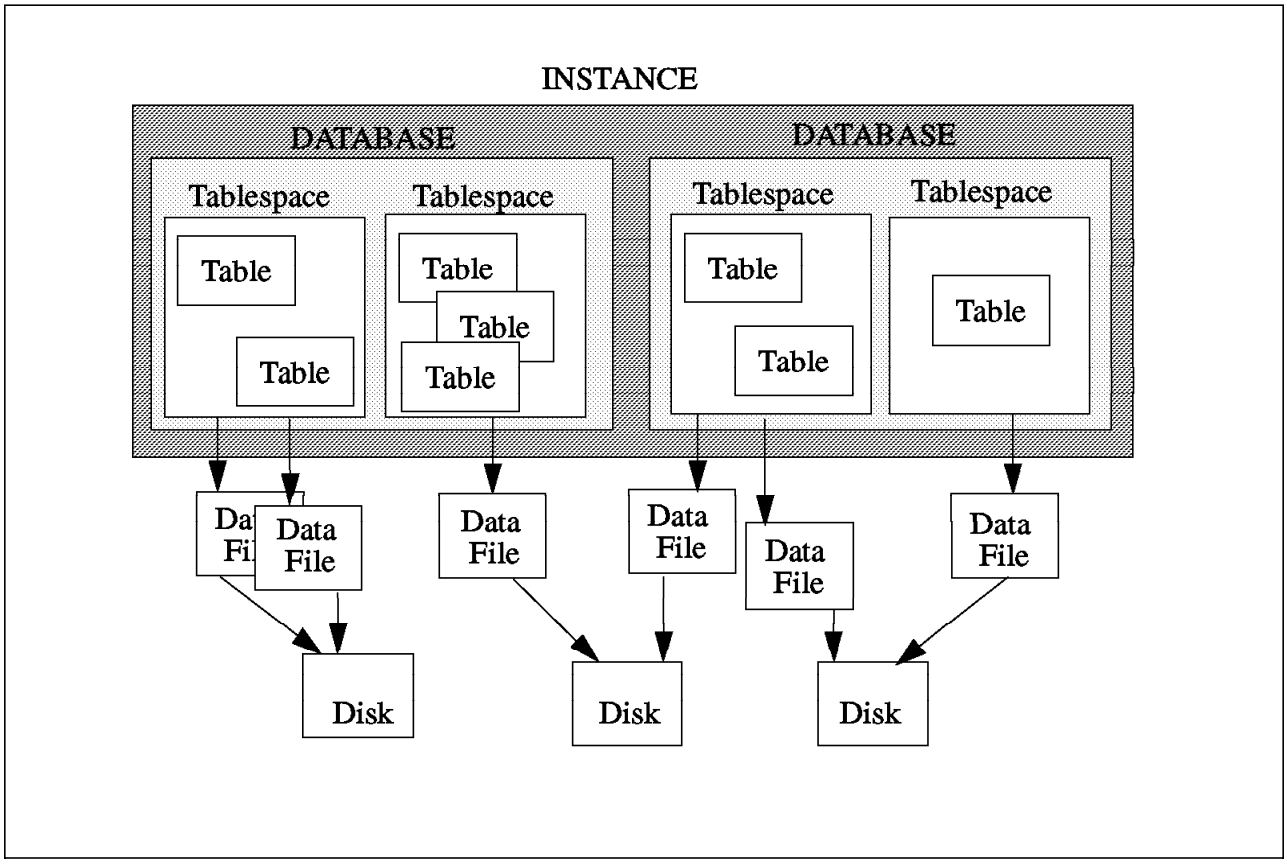


Figure 5. DATABASE 2 for AIX Version 2 Sample Database Structure

3.2 Process Model

This section discusses the differences between the Oracle and DB2 process models. The main difference between these two models is that in Oracle an instance corresponds to a single database. While in DB2, an instance is an environment that may contain multiple databases. Because of this, both models will have database level processes, but DB2 will also have processes at the instance level.

3.2.1 Oracle Process Model

In an Oracle environment, an instance can be defined as a combination of the Oracle background processes and the System Global Area (SGA). The System Global Area is a shared memory region allocated by Oracle, and contains data and control information for the Oracle instance.

These processes are used to perform all the operations requested by the user processes. The principle processes include:

reco (Recoverer) Used in distributed database environments to recover pending transactions after a network failure.

LGWR (Log Writer) Writes redo log records to the disk.

SMON (System Monitor)
Performs instance recovery and startup.

DBWR (Database Writer)
Writes the modified blocks from the database buffer cache to the physical data files.

CKPT (Checkpoint) Is an optional process, and is responsible for maintaining the indication of the most recent DBWR operation.

PMON (Process Monitor)
Performs process recovery in the case where a user process fails.

ARCH (Archiver) Copy the redo log files to archive storage when they become full.

ARH (Asynchronous Reader)
Performs asynchronous reading of data.

3.2.2 DB2 Process Model

In DB2, an instance is an environment that may contain multiple databases. It maintains a system directory of all databases and provides a level of detachment from other DB2 instances.

Each instance is a unique database manager environment. It is actually a virtual copy of the DB2 code. This is made by creating symbolic links to the shared physical copy of the installed product. This allows multiple instances to exist on a single machine with minimal overhead for each additional instance. There are several reasons for maintaining multiple instances on a single machine. These could include:

Maintaining distinct test and production environments

Restricting the access SYSADM authority has to certain databases

Exploit different database configurations

Simulate more than one client/server configuration on the same physical system.

Each instance is attached to an AIX user, and this UserID becomes the instance owner. The instance owner will have ownership of the instance files and will have system administration authority over all the databases that are contained within the instance. Most of the instance processes will be owned by the instance owner, the exception is the watchdog process which is owned by the root user.

A user cannot be the owner of more than one instance. Administrative users and groups exist to provide the capability for a single user to administer multiple instances. These groups include SYSADM, SYSCTRL, SYSMAINT and DBADM.

Because of the more complicated structure, the process model is also more complicated. The processes within a DB2 environment can be grouped into startup or instance processes, per database processes and per client or user processes. Figure 6 provides an example of the DATABASE 2 for AIX Version 2 process model.

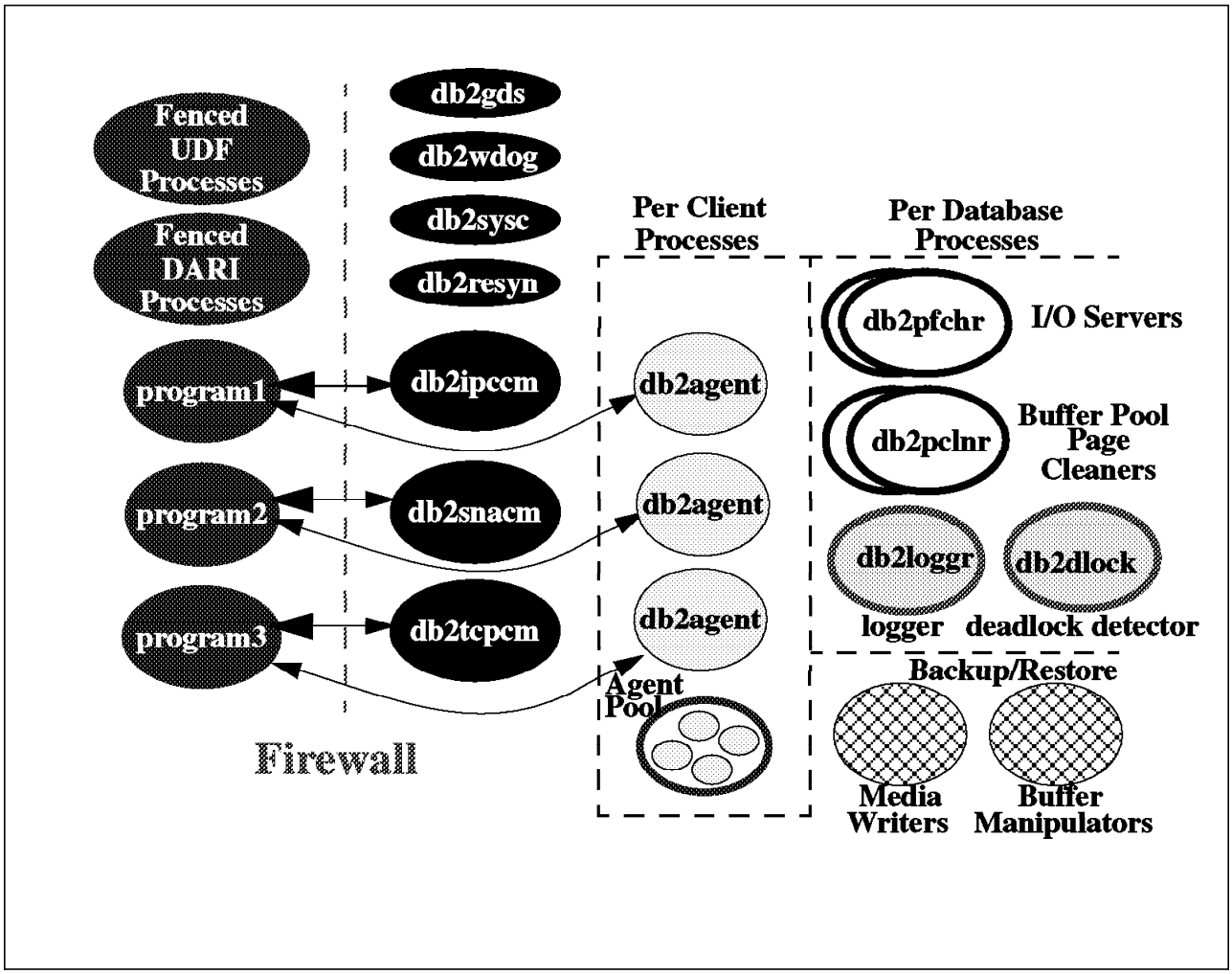


Figure 6. DATABASE 2 for AIX Version 2 Process Model

DATABASE 2 for AIX Version 2 implements a firewall when accessing the database. This firewall provides protection against application errors. All user applications access the database through a DB2 agent process. This stops a users application from overwriting or corrupting an internal database buffer. It also prevents a application from crashing the database manager or another application.

We can see in Figure 6 that programs and fenced processes exist outside the firewall. These processes are the user level processes. Inside the firewall are the instance level processes, the per client processes and the per database processes. The instance level processes are started when the database manager or instance is started. These processes include:

db2wdog The ″watchdog″ process monitors all other processes and cleans up an resources if a process terminates abnormally.

db2sysc The ″System Controller″ process handles the housekeeping tasks. These include starting and shutting down the instance.

db2gds The ″Generic Daemon Spawner″ is responsible for creating most of the DB2 demons.

db2ipccm The ″IPC Communications Manager″ handles the connection requests from the local clients. It will spawn an agent process for each local connection.

db2tcpcm The ″TCP/IP Communications Manager″ handles the connection requests from remote TCP/IP clients.

db2tcpim The ″TCP/IP Interrupt Manager″ handles interrupt requests from remote TCP/IP clients.

db2snacm The ″SNA Communications Manager″ handles the APPC connection requests from remote clients.

db2isxcm The ″IPX Communications Manager″ handles the IPX/SPX connection requests from remote clients.

db2resyn The ″resync″ will handle the recover of any indoubt DUOW transactions.

Each client application that connect to the database will have an agent process started. This agent process exists inside the firewall and will handle all the database requests from its client.

db2agent The ″agent″ process handle all the SQL processing for the application that it is associated with. each application has its own agent running inside the firewall.

There also exists some per database processes. These processes handle database operations such as buffer manipulation, logging, deadlock detection and the backup or restore of a databases. The database processes are:

db2dlock The ″database deadlock detector″ process looks for any deadlock situations on a database. If one is determined the it will take action to resolve it.

db2loggr The ″database logger″ process handles all of the log I/O for the database.

db2pclnr The ″page cleaner″ process also helps to increase efficiency by asynchronously writing dirty pages when the CPU would otherwise be idle. The number of pages cleaners is configurable.

db2pfchr The ″prefetch″ process performs read-ahead, big-block and parallel I/O. This allows for more efficient processing of data.

3.3 Database Creation

As DB2 allows for multiple databases to exist within a single instance there are some extra commands and structures used to keep track of the database locations. As Oracle only has a single database per instance, this additional information is not required.

In DB2, after the instance has been created, it is necessary to create the database or databases. A database creation will cause the following actions to be performed by the database manager:

- Creation of System Catalog tables

  Each database includes a set of system catalog tables. These contain information about the objects contained within the database. Each database has its own set of system catalogs.

- Creation of Database recovery log

  The database recovery log keeps a record of all the changes made to the database. Each database has its own recovery log.

- Creation of Configuration files

  The configuration files for a database are also stored in the same location as the database itself. Each database has its own configuration information and may be viewed or updated using the database director or the command line processor.

- Creation of System Database Directory

  The system database directory is created for each instance. Refer to 3.3.1, “Directories in DB2” for further information on the system database directory.

- Creation of Local Database Directory

  The local database directory is created in every location that contains a database. Refer to 3.3.1, “Directories in DB2” for further information on the local database directory.

- Binding of Utilities to Database

  Before a database application can be executed it must be bound to the database. The database utilities are bound as a part of the database creation process.

3.3.1 Directories in DB2

DB2 uses four directories to contain information about database locations and connections. These directories are not the same as a filesystem directory. The four directories are:

System Database Directory

Local Database Directory

Node Directory

DCS Directory

The *system database directory* is used to identify the name, alias and physical location of each cataloged database. For a user or application to access a

database, it must be in the system database directory for both the client and the server.

The *local database directory* contains the name of a database and the location in which the database files are stored. There is a local database directory in every location that contains a database.

The *node directory* contains an entry for all nodes that can be connected to. It lists the node's name along with some communication and instance information. The term node is used to specify a system in the network. When the "CONNECT TO" statement is issued, the database directory and the node directory are accessed to determine if the database is local or remote. The communications protocol and connection type are also determined by looking at the directories.

The *database connection services* or DCS directory is used for databases that are accessed via the DDCS product. It contains an entry for each DRDA database that can be accessed by your node or instance.

The node directory, DCS directory and system database directory are all stored within the instance directory. As a database can theoretically be stored anywhere in the AIX file system structure, the local database directory can also be anywhere. It is stored in the database directory, which by default is the home directory of the instance owner.

3.4 Client/Server Models

There are a number of different client and server configurations. This chapter will look at the three of the most common environments and look at the configurations in each.

3.4.1 DB2 Stand-Alone Configuration

For a small business or a development environment, it is common to have a single machine running DB2. Access to the database may be limited to local users. For this environment you would only need to install either the server kit or the single-user kit. This would depend on the number of users and what the machine is to be used for. There is no requirement for additional client code, as the client application enabler code is built in to the server code.

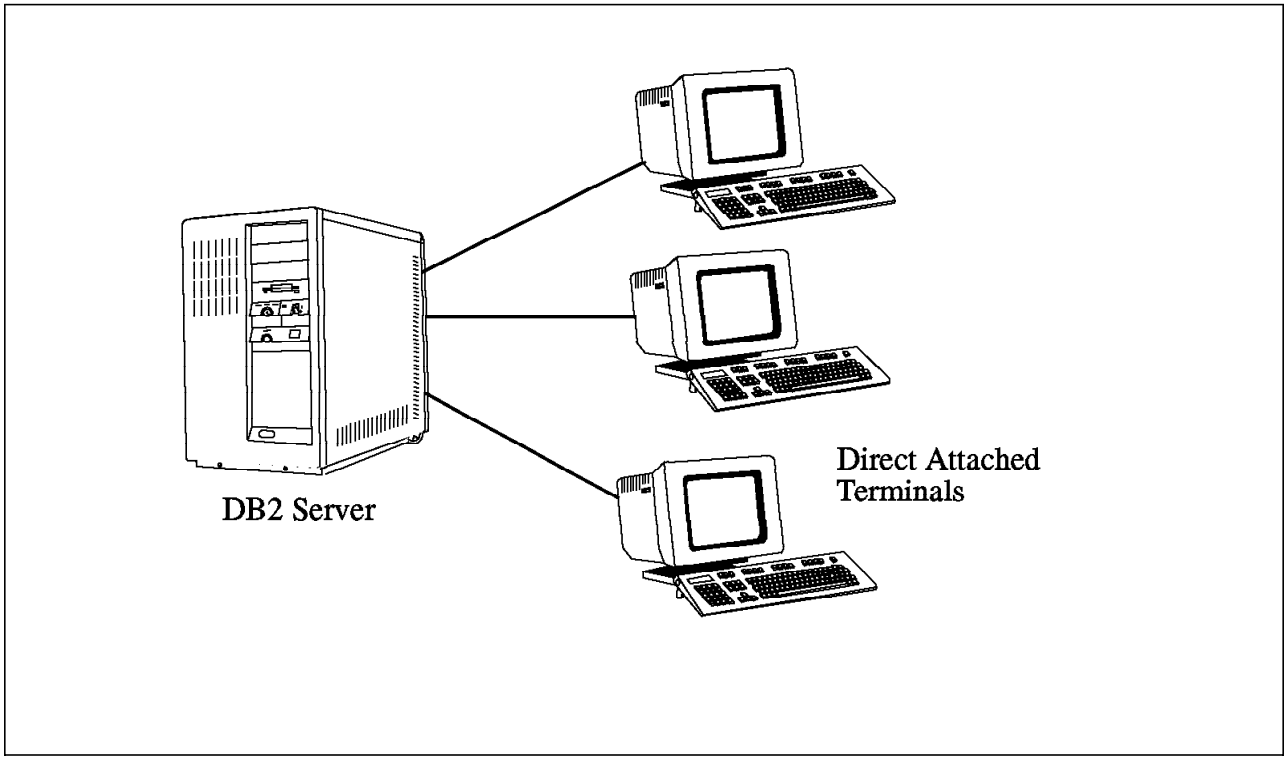


Figure 7. DB2 Stand-alone Configuration

3.4.2 DB2 Server in a LAN Environment

In a Local Area Network (LAN) it is possible to support both locally attached users, and LAN clients. To do this the server kit would need to be installed on the server machine, along with the appropriate communications protocol in the communications support kit.

Depending on the operating system running on the client, you would need to install the appropriate client application enabler kit. This will allow applications to connect to the server database.

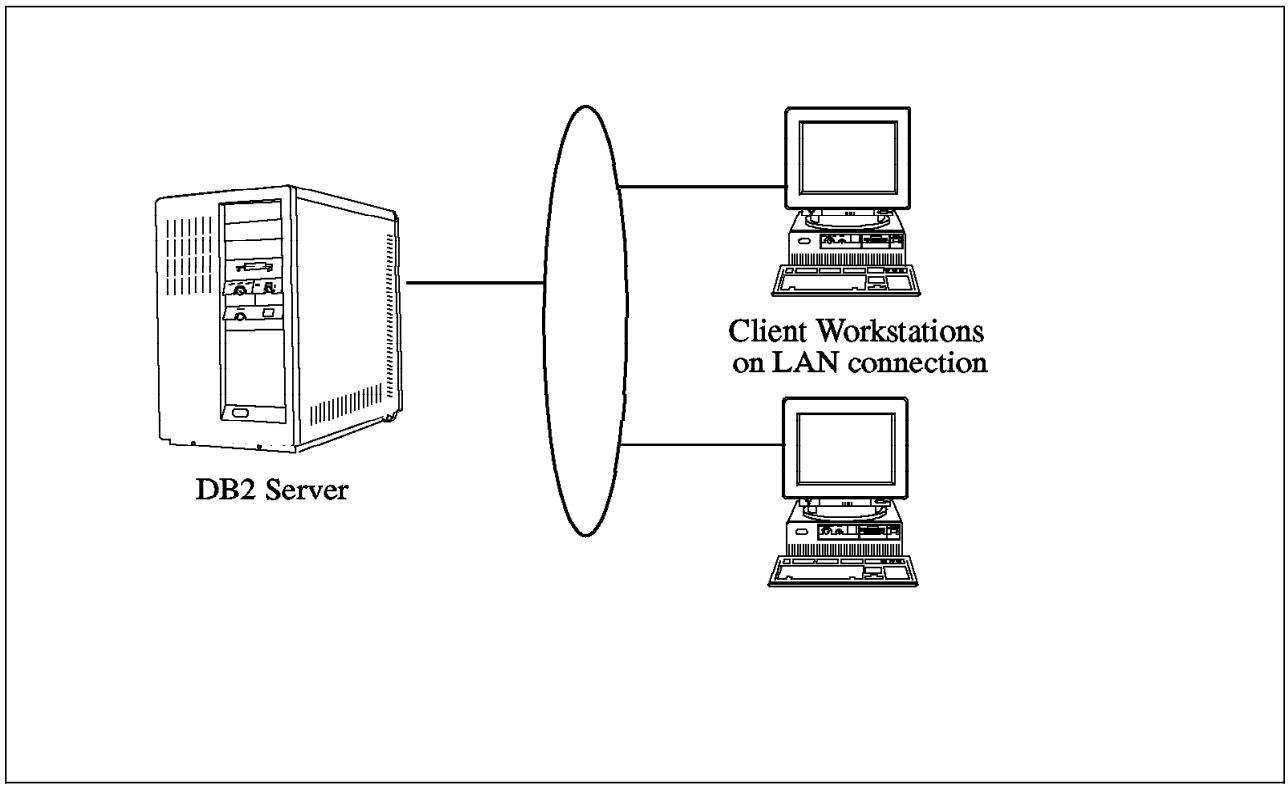


Figure 8. DB2 Server in a LAN Environment

3.4.3 DB2 Server in a LAN Environment with Host Connection

If your DATABASE 2 for AIX Version 2 server needed to support both clients and a connection to a host database, then you would need to set up an environment similar to that in 3.4.2, “DB2 Server in a LAN Environment” on page 32. In addition to this environment you would need to install the DDCS for AIX Version 2.3 Multi-User Gateway. This will allow the DB2 server and its clients to connect to the host database.

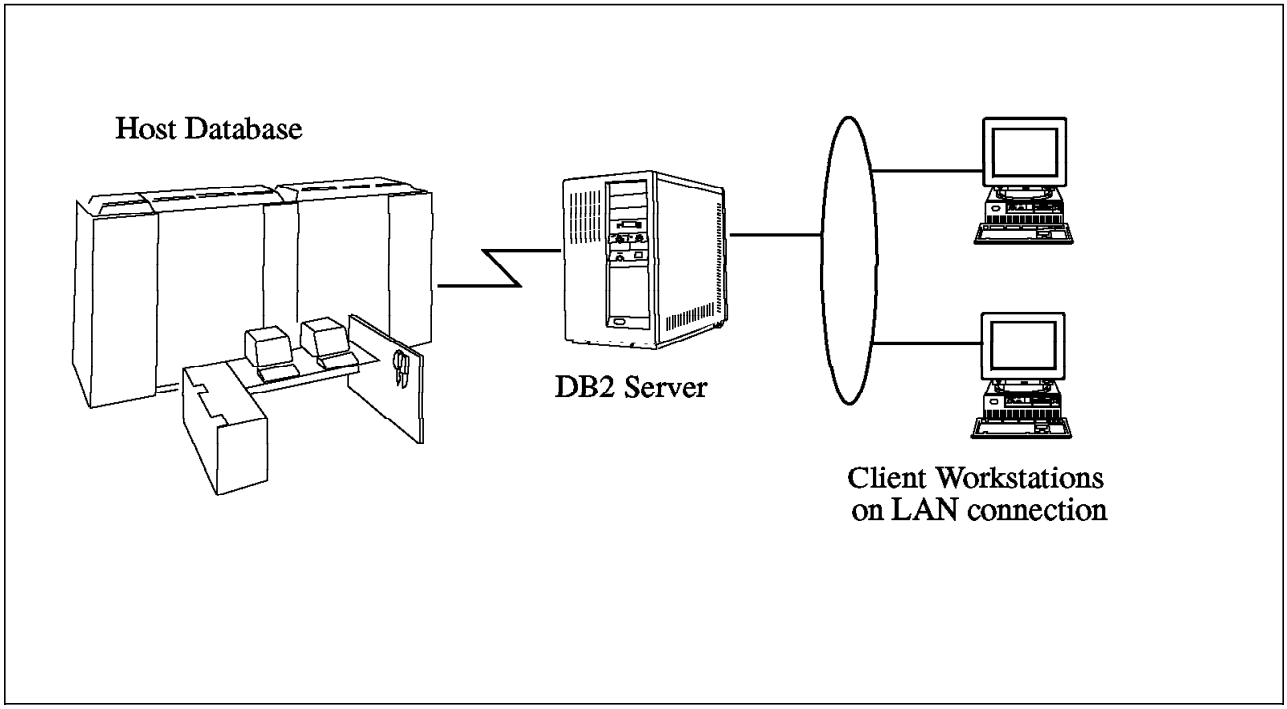


Figure 9. DB2 Server in a LAN Environment with a Host Connection

Chapter 4. Storage

This chapter discusses the differences in data and object storage for Oracle 7 and DATABASE 2 for AIX Version 2.

The topics that are covered include:

Physical Storage Elements

Logical Storage Elements

Database Storage Elements

4.1 Physical Storage Devices

The physical storage devices are the set of elements used to store the different database objects. Storage devices include UNIX files, devices and directories. The are many similarities between Oracle and DB2 in the area of physical and logical storage. However, the terminology may be different. This chapter will cover the different ways the database systems use physical storage devices to build a logical storage area for the database system.

4.1.1 Oracle 7 Physical Structure

An Oracle database's physical structure is determined by the operating system files that constitute the database. Each Oracle database is comprised of four types of files:

- Data files
- Redo log files
- Control files
- Configuration files

The files of a database provide the actual physical storage for database information. The file may simply be a file within the filesystem, or it may be a raw device, such as a logical volume.

4.1.1.1 Data Files

Every Oracle database has one or more physical data files. These data file contain all the data stored in the database system. Information on database structures, such as tables and indexes, is also stored in the data files allocated for a database.

Database file have the following characteristics:

- A data file can be associated with only one database.
- Once created, a data file cannot change in size.
- One or more data files form logical units of database storage called a tablespace.

4.1.1.2 Redo Log Files

Every Oracle database has a set of two or more redo log files. The set of redo log files for a database are collectively known as the database's redo log. The primary function of the redo log is to record all changes made to the database data. Should a failure prevent modified data from being permanently written to the data files, the changes can be obtained from the redo log. This ensures information or data is never lost. One or more copies of the redo log can be maintained on different disks.

4.1.1.3 Control Files

Every Oracle database has a control file. A control file records the physical structure of the database. Control files contain the following types of information:

- The database name
- Names and locations of a database's data files and redo log files
- Time stamp of the database's creation

Oracle allows the control file to be mirrored for protection of the control file.

Each time an instance of an Oracle database is started, its control file is used to identify the database, the physical structure of the data and the redo log files that must be opened for the database operation to proceed. It is also used for recovery, if required.

4.1.1.4 Configuration Files

Every Oracle database has two configuration files. A configuration file records database parameters values, such as:

- Redo log files location
- Control files location
- System Global Area size (for more information, refer to the chapter on the relational database model)

These files are config.ora and init.ora. These files are plain text files and can be viewed with a text browser, such as vi. A sample of these files is shown in Figure 10 on page 37.

```
(Oracle)/u/oracle/dbs> cat init.ora
# $Header: initx.orc 7001300.3 93/06/16 12:28:26 mkrishna Osd<unix>
# $ Copyr (c) 1992 Oracle
# include database configuration parameters
ifile                   = /u/oracle/dbs/configA.ora
#rollback_segments      = (r01,r02,r03,r04)
# tuning parameters
db_files = 20
db_file_multiblock_read_count = 8              # SMALL
db_block_buffers = 200                         # SMALL
shared_pool_size = 3500000                     # SMALL
# mts_dispatchers="ipc,1"
# mts_max_dispatchers=10
# mts_servers=1
# mts_max_servers=10
# mts_service=A
# mts_listener_address="(ADDRESS=(PROTOCOL=ipc)(KEY=A))"


(Oracle)/u/oracle/dbs> cat config.ora
# $Header: cnfg.orc 7001200.2 93/04/26 14:58:22 eruben Osd<unix>
# $ Copyr (c) 1992 Oracle
# cnfg.ora - instance configuration parameters
control_files            = (/u/oracle/dbs/ctrl1A.ctl,
                            /u/oracle/dbs/ctrl2A.ctl,
                            /u/oracle/dbs/ctrl3A.ctl)
# Below for possible future use...
#init_sql_files          = (?/dbs/sql.bsq,
#                           ?/rdbms/admin/catalog.sql,
#                           ?/rdbms/admin/expvew.sql)
background_dump_dest     = /u/oracle/rdbms/log
core_dump_dest           = /u/oracle/dbs
user_dump_dest           = /u/oracle/rdbms/log
#log_archive_dest        = /u/oracle/dbs/arch/arch.log
#db_block_size           = <blocksize>
db_name                 = A
```

Figure 10. Example of the config.ora and init.ora Files

4.1.2 DATABASE 2 for AIX Version 2 Physical Structure

A DB2 instance can manage multiple databases. When a database is created, the database manager creates a separate subdirectory to store control files and containers for the default tablespaces. Objects associated with the database, including devices such as logical volumes, are not always stored in the database directory. They can be stored in various locations.

The naming scheme for the database is /instance_name/SQL00001 through /instance_name/SQLnnnnn, where SQL00001 contains objects associated with the first database created; SQL00002 contains objects for the second database, and so on. These subdirectories are created within the PATH specified in the CREATE DATABASE command or, by default, in the instance owner's home directory.

4.1.2.1 Data Files

In DB2, the physical structure of the data files is dependent upon the type of tablespace that you define. There are two types of managed tablespaces, and both types can be utilized within a single database. The two types of managed tablespaces are:

- **System Managed Space (SMS) Tablespace**

  The operating system's file manager controls the storage space within a file system.

- **Database Manager Space (DMS) Tablespace**

  The database manager controls the storage space within a logical volume or file.

The SMS tablespace is a generalization of the storage model found in DB2 Version 1. Version 1 database files are stored under a configured number of subdirectories. All the data files are located under the database subdirectory. The database or system administrator can tell DB2 where to create the database subdirectory. The name and location of all the segment subdirectories is automatic. The number of subdirectories created can be specified at database creation time and cannot be changed afterward. In DB2 Version 2, SMS allows the database administrator to specify any directory that is accessible by the system, as a location for storing database files. These directories are called *containers*. In an SMS tablespace, one container maps to a single directory. Given this, the maximum size of an SMS tablespace would be the number of containers multiplied by the maximum file system size supported by the operating system.

The following files are found within an SMS tablespace directory:

SQLxxxxx.DAT	Table file. All rows of a table are stored here, with the exception of LONG VARCHAR, LONG VARGRAPHIC, CLOB, BLOB, and DBLOB data.
SQLxxxxx.LF	Files containing LONG VARCHAR or LONG VARGRAPHIC data. This file is only created if LONG VARCHAR or LONG VARGRAPHIC columns exist in the table.
SQLxxxxx.LB	Files containing CLOB, BLOB, or DBBLOB data. This file is only created if CLOB, BLOB, or DBBLOB columns exist in the table.
SQLxxxxx.LBA	Files containing allocation and free space information about the SQLxxxxx.LB file.
SQLxxxxx.INX	Index files for a table. All indexes for the corresponding table are stored in this file. It is only created if indexes have been defined. When an index is dropped, the space is not physically freed from the index file until the index file is deleted. This occurs when all indexes for the table have been deleted.
SQLxxxxx.EIX	Damaged INX file for a table.
SQLxxxxx.DTR	Temporary data files for a REORG of a DAT file.
SQLxxxxx.LFR	Temporary data files for a REORG of an LF file.
SQLxxxxx.RLB	Temporary data files for a REORG of an LB file.
SQLxxxxx.RBA	Temporary data files for a REORG of an LBA file.

For more information about SMS tablespaces, refer to the documentation supplied with DATABASE 2 for AIX Version 2.

The DMS tablespaces are built on pre-allocated disk partitions or files. This pre-allocated space can be a large single file or a logical volume. The file or logical volume need to have the owner and group set to match the DB2 instance owner. Figure 11 is an example of how to create a DMS tablespace. The database manager then becomes responsible for the management of this space. The table objects for the data, indexes and long columns of a table can be stored in the same tablespace or in different tablespaces. The size of a DMS tablespace can be increased by adding storage elements called *containers*.

In DMS tablespaces, a container can be either a logical volume or a file. The implementation of a DMS tablespace is similar to an Oracle tablespace.

```
o Create a 16 MB Logical Volume called lobA in the root volume group.

     # mklv -y'lobA' rootvg 4
     lobA

o Change the ownership to match the DB2 instance owner.

     # chown db2.db2gr /dev/rlobA
     # chmod 600 /dev/rlobA
     # ls -al /dev/rlobA
     crw-------   1 db2      db2gr     10, 13 Apr 28 13:22 /dev/rlobA
```

Figure 11. Creating Raw Devices

4.1.2.2 Choosing an SMS or DMS Tablespace

There are a number of points to consider when determining which type of tablespace you should use to store your data. Table 6 summarizes these points.

Table 6. Choosing an SMS or DMS Tablespace

	SMS Tablespace	DMS Tablespace
Tablespaces can share containers	YES	NO
Increase number of containers in tablespace	NO	YES
Store Index data in separate tablespace	NO	YES
Store long data in separate tablespace	NO	YES
One table (index, data, LOB) can span several tablespaces	NO	YES
Container can be a raw device	NO	YES

4.1.2.3 Log Files

Like Oracle, DB2 maintains a set of log files. The database recovery log keeps a record of all changes made to a database, whether it is the addition of new data or the update of existing data. This database log can be used to ensure that a failure does not leave the database in an inconsistent state. The naming convention used for the database log files is *Syyyyyyy.LOG*. These logs are stored within the database directory by default. It is possible to configure the log file path so that the logs are stored elsewhere.

4.1.2.4 Control Files

These files are used by the database manager to verify that the database is complete and consistent. The following list contains the control files used by the database:

- **SQLOGCTL.LFH**: This file is used to help track and control all of the database log files.
- **SQLINSLK, SQLTMPLK**: These files are used to help ensure that a database is only used by one instance of the database manager.
- **SQLSPCS.1**: This file contains the definition and current state of all tablespaces in the database.
- **SQLSPCS.2**: This file is a copy of SQLSPCS.1.
- **SQLTAG.NAM**: There is one of these files in each container subdirectory. They are used by the database manager when you connect to the database to verify that the database is complete and consistent. These files are only used for SMS tablespaces.

4.1.2.5 Directory Files

The directory files are used for accessing both local and remote databases. These directories contain information which ensures that access to a database is transparent to users and applications, regardless of physical location of the database. These directories are not the same as an AIX filesystem directory. They are stored as structured files within the database environment. Oracle does not have an equivalent directory structure.

DB2 uses three types of database directories to identify the location of the databases, nodes and connection information.

The three directories can be summarized as follows:

- The *Database Directories* identify the name, alias and physical location of each cataloged database. There is a System database directory and Local database directory
- The *Node Directory* contains an entry for all nodes to which your database client can connect. Each entry contains the node′s name, its communication information, and instance information if it exists.
- The *Database Connection Services (DCS) Directory* is only used if DDCS is installed on your system. It contains an entry for each DRDA database that your node can access.

Figure 12 on page 41 is an example of how the various directories interact. In the example, a database called DB2DB exists on a server machine. The hostname of the server machine is ′db2serv′. This database has also been

cataloged on the server under the name 'TORONTO3'. A user local to the server could use either of these names to connect to the database.

The client workstation has a database called 'TOR3'. From the Database Directory, we see that 'TOR3' is an alias for the database 'TORONTO3' on the node 'TORONTO' When a user on the client connects to 'TOR3', this information is extracted from the Database Directory along with the node information from the Node Directory. From this information, the location of the database is determined, and a remote connection is established.

By doing this, is it possible to access both local and remote databases in exactly the same way. The user or application does not have to keep track of the physical location of a database.

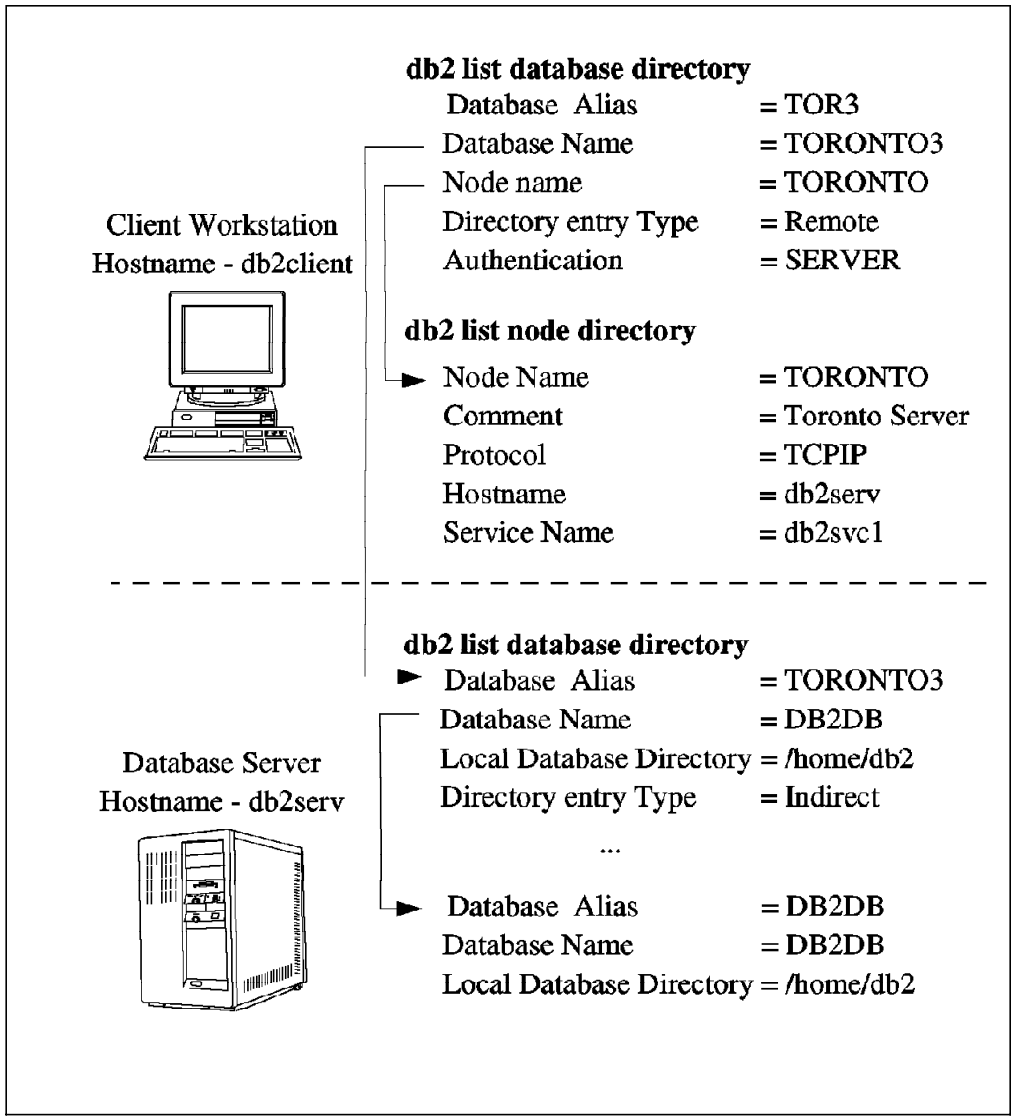


Figure 12. Example of Directories Interacting

The directory information can be viewed or modified using the following commands:

- `LIST DATABASE DIRECTORY`
- `LIST DCS DIRECTORY`

- CATALOG

Figure 13 shows the results of the List Databases Directory command.

```
$ db2 list database directory

        System Database Directory

        Number of entries in the directory = 1

        Database 1 entry:

            Database alias                 = TITI
            Database name                  = TITI
            Local database directory       = /home/jcb
            Database release level         = 6.00
            Comment                        = A Test database
            Directory entry type           = Indirect
            Authentication                 = SERVER
```

Figure 13. Example of a List Database Directory

4.1.2.6 Configuration Files

Configuration files contain parameter values that define the allocated resources for the DB2 instance and for the individual databases.

There are two types of configuration files.They are the *database manager configuration* file for the DB2 instance as a whole and the *database configuration file* for each individual database.

The database manager configuration file is created when an instance of DB2 is created. Its parameters define the available system resources at a global or instance level. This is independent of any individual database stored within the instance. Many of these parameters can be changed. There is one database manager configuration file for each installation of a instance.

The database configuration file is created when a database is created, and it resides where that database physically resides. There is one database configuration file per database. Its parameters specify the amount of resources available for that database.

These files cannot be directly edited. The following commands are used to update and list the configuration.

- GET DATABASE MANAGER CONFIGURATION
- UPDATE DATABASE MANAGER CONFIGURATION ...
- GET DATABASE CONFIGURATION FOR dbname
- UPDATE DATABASE CONFIGURATION FOR dbname ...

4.1.2.7 Estimating Space Requirements for Tables

Table 7 provides a general rule for estimating the size of a database. Caution should be taken when allocating space for the system catalogs if they reside in a DMS tablespace. The size of a table in DMS tablespaces is also dependent on the tablespace extent size.

Table 7. Sizing the Database

Object	Formula
System catalog tables	When a database is initially created, about 420 KB of system tables are created. These system tables will grow as user tables, views, indexes, authorizations, and packages are added to the database.
User data tables	(average row size + 8) * numbers of rows * 1.5 The average row size is the sum of the average column sizes. For long field data and large object data, refer to the product documentation for information on how these are stored.
Index space	(average index key size + 8) * number of rows * 2 The average index key size is the byte count of each column in the index key. Temporary space is required when creating the index. The maximum estimation is (average index key size + 8) * number of rows * 3.2.
Log file space	The amount of space (in bytes) required for logging is minimally (logprimary * (logfilsiz + 2) * 4096) + 8192 If the database is configured for circular logging and secondary logs are used during interaction with the database, space must be added for these files during run time: (logsecond * (logfilsiz + 2)).
Temporary work space	The amount of required disk space will be totally dependent on the queries, and therefore space is difficult to estimate.

4.2 Logical Storage Devices

Both Oracle and DB2 divide the database into separate storage elements. This division allows an independence between the database manager's conceptual storage model and the physical storage.

4.2.1 Logical Storage Devices in Oracle

A database is divided into multiple logical storage units called *tablespaces*. The usable data of an Oracle database is logically stored in the tablespace and physically stored within the data files associated to the tablespace.

When an Oracle database is created, space is pre-allocated for the tablespace data files. The logical allocation of space is broken down into data blocks, extents and segments.

At the finest level of granularity, an Oracle database's data is stored in *data blocks* (also called logical blocks, Oracle blocks or pages). One data block corresponds to a specific number of bytes of physical space on disk. The data block size is specifically set for each Oracle database at the time of database creation.

The next level of logical database space is called an *extent*. An extent is a specific number of contiguous data blocks that area allocated for storing data.

The level of logical database storage above an extent is called a *segment*. A segment is a set of extents that have been allocated for a specific type of data structure and stored within the same tablespace. There are five segment types:

- Data segment
- Index segment
- Rollback segment
- Temporary segment
- Bootstrap segment

Oracle allocates segment space using extents. Therefore, when the existing extents of a segment is full, another extent for that segment is allocated. Because the extents are allocated as required, the extents of a single segment may not be contiguous on disk and may exist in multiple data files. However, a single extent must be fully contained within a single data file. The allocation space is dependent on the parameter defined by the storage clause. Such as MINEXTENTS, MAXEXTENTS, PCTINCREASE, INITIAL, or NEXT.

Figure 14 shows the relationships between the different data structures that Oracle uses to allocate physical space and create the logical database storage.

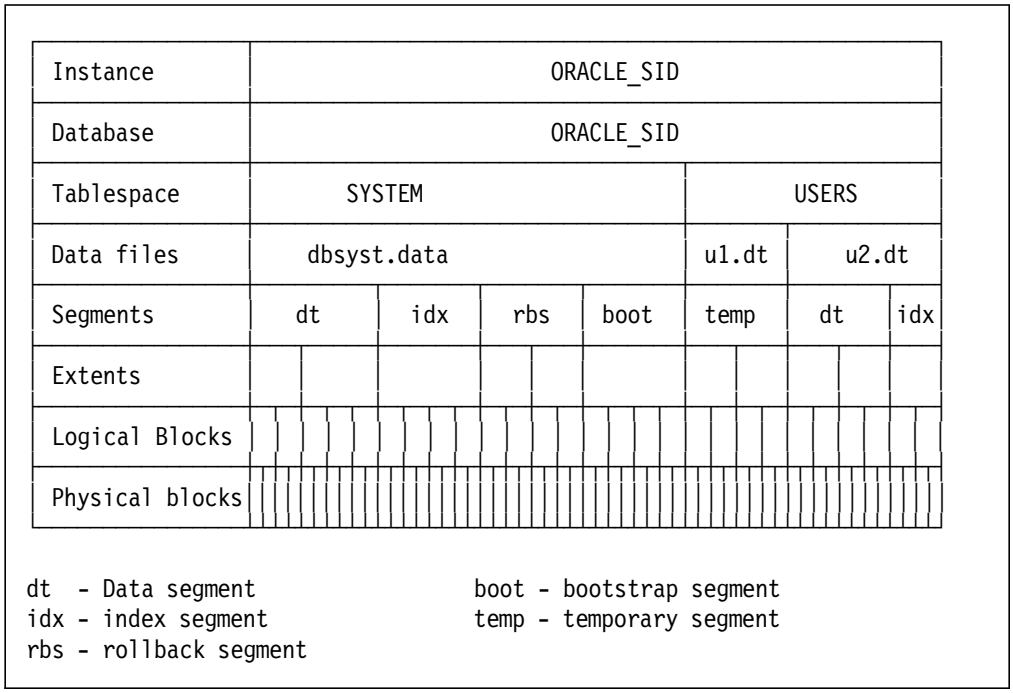


Figure 14. Logical Storage Structure in Oracle

Each Oracle database contains a tablespace named *SYSTEM*, which is automatically created when the database is created. The SYSTEM tablespace contains the data dictionary tables for the entire database. The SYSTEM tablespace must always be kept online.

It is possible to enlarge the size of a database by adding an additional data file to an existing tablespace. However, the new data file will be used only when the first data file of this tablespace is full.

Note: In DB2 Version 1, there exist segment directories. These directories are not related to the segments in Oracle. The DB2 Version 1 segments were subdirectories where the data files were stored. In DATABASE 2 for AIX Version 2, when you use an SMS tablespace, these subdirectories are called containers, rather than segments.

4.2.2 Logical Storage Devices in DB2

4.3 logical

Like Oracle, a DB2 database is divided into logical storage units called *tablespaces*. The data of a DATABASE 2 for AIX Version 2 database is logically stored in the tablespace and physically stored in the *containers* associated with the tablespace. A container is a generic term used to describe the allocation of space to a tablespace. The container in DB2 is much like the data files found in Oracle A container can be any of the following:

- File (DMS tablespace container)
- Directory (SMS tablespace container)
- Logical Volume (DMS tablespace container)

To increase a tablespace (DMS only), you can add more containers to it.

There are two commands to list tablespaces and their containers:

- `LIST TABLESPACES [SHOW DETAIL]`
- `LIST TABLESPACE CONTAINERS FOR tablespace-ID [SHOW DETAIL]`

Figure 15 on page 46 shows the results of the `List Tablespaces [SHOW DETAIL]` command.

```
$ db2 LIST TABLESPACES

          Tablespaces for Current Database

Tablespace ID                        = 0
Name                                 = SYSCATSPACE
Type                                 = Database managed space
Contents                             = Any data
State                                = 0x0000

Tablespace ID                        = 1
Name                                 = TEMPSPACE1
Type                                 = Database managed space
Contents                             = Temporary data
State                                = 0x0000

Tablespace ID                        = 2
Name                                 = USERSPACE1
Type                                 = Database managed space
Contents                             = Any data
State                                = 0x0000

$ db2 LIST TABLESPACE CONTAINERS FOR 2 SHOW DETAIL

           Tablespace Containers for Tablespace 2

Container ID                         = 0
Name                                 = /u/jcb/user.fl
Type                                 = File
Number of tablespaces                = 1
Total pages                          = 1000
Useable pages                        = 992
Accessible                           = Yes
```

Figure 15. Example of Tablespace and Container Lists

An *extent* is a contiguous allocation of space within a tablespace container. The extent is allocated to a single type of database object. This allocation consists of multiple *pages* of 4 KB each. The extent is given a default size at the time of database creation. The default size of an extent is 32 pages, and each page is 4 KB. The extent size for a tablespace indicates the number of pages of table data that will be written to a container before data will be written to the next container.

Figure 16 on page 47 shows the relationship between these DATABASE 2 for AIX Version 2 data structures.

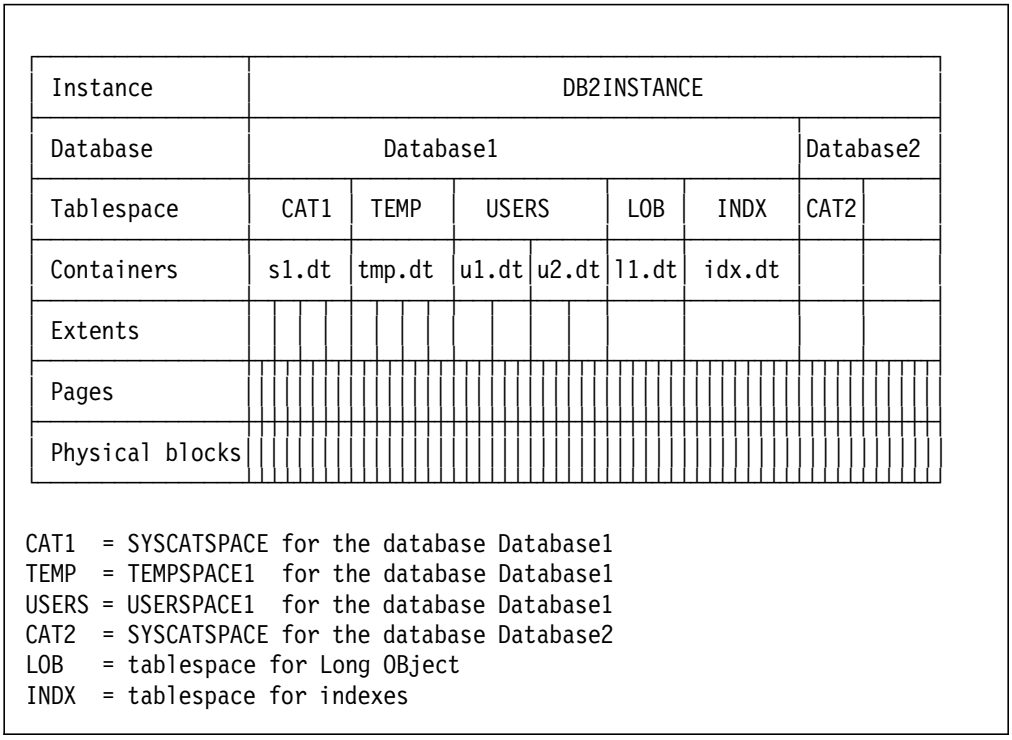


Figure 16. Logical Storage Structure in DATABASE 2 for AIX Version 2

A DATABASE 2 for AIX Version 2 database contains at least three tablespaces:

- *Catalog* tablespace, which contains all the system catalog tables for the database. This tablespace is similar to the Oracle SYSTEM tablespace. This tablespace is called SYSCATSPACE and it cannot be dropped.
- *User* tablespace, which contains all user-defined tables. The USERSPACE1 tablespace is created at the time of database creation. This is the default user tablespace.
- *Temporary* tablespace, which contains all the temporary tables. By default, one tablespace, called TEMPSPACE1, is created. A database must have at least one temporary tablespace.

It is important to note that in DB2 the tablespace containers are are written to in a round-robin manner. The database manager writes the first extent of the object to the first container in the tablespace, the second extent of the object on the second container in the tablespace, and so on until all containers have had an extent written to them. This is different from Oracle, which only writes to a second data file only when the first data file has been filled.

4.4 Database Storage Elements

Both Oracle and DB2 use tablespaces to contain the database data. Within a tablespace are tables and indexes. The allocation of space for these object may be different between the two systems. This chapter will discuss how the space is allocated for these objects.

4.4.1 Tables

In Oracle, you can specify a tablespace name when you create a table. If you do not specify a name, the table is placed in the table owner′s default tablespace or in the default SYSTEM tablespace.

When creating a table, a data segment is automatically allocated in the associated tablespace. You can control the allocation of space for a table′s data segment and the use of this reserved space in the following ways:

- You can control the amount of space of the data segment′s extents by setting the storage parameters for the data segment (INITIAL, NEXT, PCTINCREASE, and so on).
- You can control the use of the free space in the data blocks that constitute the data segment′s extents by setting the PCTFREE and PCTUSED parameters for the data segment.

In DB2, as in Oracle, you can specify a tablespace name when you create a table. If you do not specify a tablespace name, the table is created in the first user-created user tablespace. If none are found, the table is placed into the default system-created user tablespace, USERSPACE1. If USERSPACE1 has been dropped, the table creation fails.

In addition, DB2 allows you to place the data contained in the LONG columns of a table into a separate tablespace. This is specified when the table is created.

4.4.2 Indexes

In Oracle, you can specify a tablespace name for an index during the create index statement. Otherwise, the indexes will be placed in the default tablespace of the user.

In DB2, if you specify a tablespace name for an index during the create table statement, indexes for that table will be stored in the named tablespace. Otherwise, the indexes will be stored in the same tablespace as the table.

4.4.3 Extent Allocation in DATABASE 2 for AIX Version 2

One of the new configuration parameters in DATABASE 2 for AIX Version 2 is DFT_EXTENT_SZ. This is defined at the database level and may vary between databases. It determines how many pages are written to a container before writing to the next container. The default size for DFT_EXTENT_SZ is 32 (4 KB page). If you do not alter this value, all your tablespaces within the database will have this default value. The range of values for DFT_EXTENT_SZ is between two and 256 pages.

You may still change the number of pages written before writing to another container at the tablespace level. This change can be done at tablespace creation with the parameter *EXTENTSIZE*. Care should be taken to determine the correct size since, once it is set for a tablespace, it cannot be altered. This size may have an impact on space utilization and performance.

The database manager will try to evenly distribute the table′s data among the containers. In doing so, the database manager writes up to a defined number of pages to each container before writing to the next container. The number of pages written to a container before writing continues in the next container is called EXTENTSIZE. Once the database manager has written to all the containers

allocated to the tablespace, it will write back to the one it started with. This round-robin process of writing to the containers is designed to balance the load.

Figure 17 shows an example of the extent allocations for tablespace that contains two tables. The first table is the *DEPT* table, and it fills four extents. The second table is the *EMPLOYEE* table that contains only three extents. The extents that make the tables have been evenly spread over the three containers.

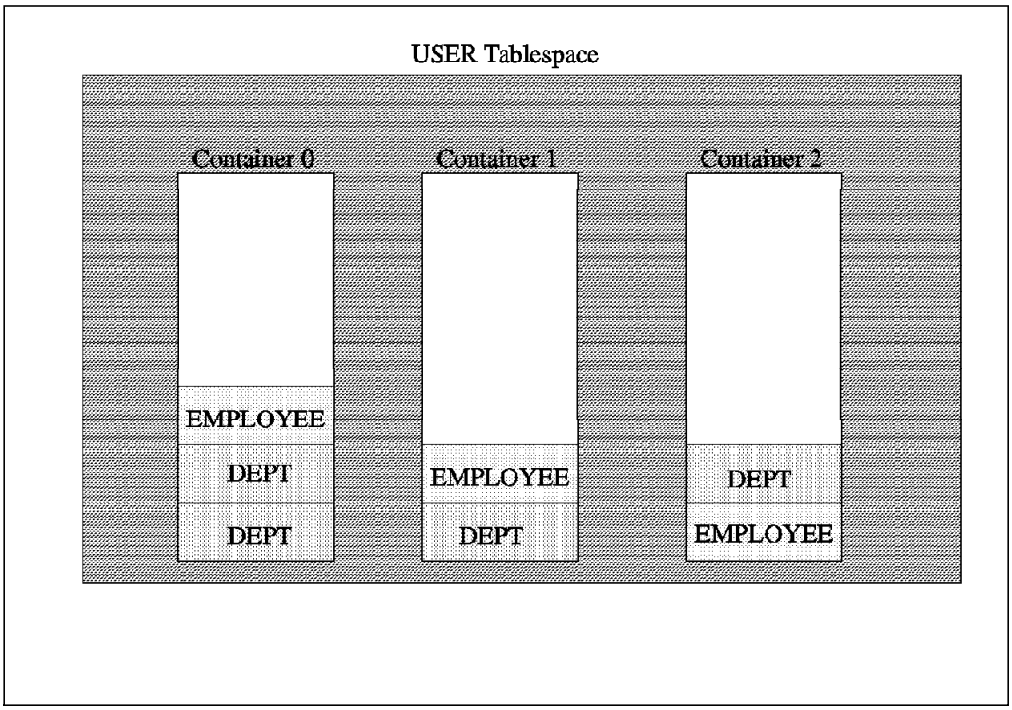


Figure 17. Use of Containers and Extents in DATABASE 2 for AIX Version 2

When selecting an extent size, you should consider:

- The size of the tables in the tablespace

  Space in DMS tablespaces is reserved for a table an extent at a time. If you have a table that is much smaller than an extent in the tablespace, some space will not be used. If, on the other hand, you have a very large table that has a high growth rate, and you are using a DMS tablespace with a small extent size, you could have unnecessary overhead related to the frequent allocation of additional extents.

- The type of access to the tables

  If access to the tables includes many queries or transactions that process large quantities of data, pre-fetching data from the tables may provide significant performance benefits.

- The minimum number of extents required

  There must be enough space in the containers for five extents of the tablespace; otherwise, the tablespace will not be created.

4.4.4 Tablespace Translation Example

The following example translates the Oracle tablespace organization into DB2. In this example, there are five tablespaces in Oracle:

- SYSTEM tablespace contains the data dictionary

  Location : /data/systA.dbf
  Size : 6400 * 4 KB pages

- RBS tablespace contains the rollback segments

  Location : /data/rbsA.dbf
  Size : 1024 * 4 KB pages

- TOOLS tablespace contains the tables for SQL*FORMS

  Location : /data/toolA.dbf
  Size : 6144 * 4 KB pages

- USERS tablespace contains the user's tables

  Location : /data/usrA.dbf
  Size : 512 * 4 KB pages

- TEMP tablespace contains the temporary segments

  Location : /data/tempA.dbf
  Size : 138 * 4 KB pages

It is possible to create a similar organization in DB2. In DB2, we do not need to create the tablespaces TOOLS and RBS because they are not required by DB2. We can also add a tablespace for the LOB data.

Given the above, for DB2, we would create the following.

- The SYSTEM tablespace is called SYSCATSPACE in DB2.

  Location : /data/systA.dbf
  Size : 6400 * 4 KB pages

- The USERS tablespace contains the user's tables.

  Location : /data/usrA.dbf
  Size : 512 * 4 KB pages

- The TEMP tablespace contains the temporary tables.

  Location : /data/tempA.dbf
  Size : 138 * 4 KB pages

- The LOBA tablespace is added to store LOB columns.

  Location : /dev/rlobA
  Size : 4000 * 4 KB pages

The location of the DB2 containers or data files could be the same as the files used by ORACLE, or you could choose to restructure by using a combination of SMS and DMS tablespaces.

An example of creating this new DB2 environment is shown in the following steps. In this case, we will assume the instance owner is the user db2i, and the primary group is db2adm. The first two steps should be done as the root user, while the remaining steps should be done as the instance owner.

1. Create a logical volume for the LOB tablespace container.

- mklv -y lobA rootvg 4
- chown db2i.db2adm /dev/rlobA
- chmod 600 /dev/rlobA

This will create the logical volume, and set the correct ownership and permissions on the device.

2. Change the ownership and group for the existing containers of Oracle.

   - chown db2i.db2adm /data/*.dbf
   - chmod 600 /data/*.dbf

This corrects the ownership for all the 'dbf' files in the data directory. The files that are not being used can be deleted from this directory.

3. Create the database.

   - CREATE DATABASE A
     CATALOG TABLESPACE
     MANAGED BY DATABASE USING (FILE '/data/systA.dbf' 6400);

     This will create the new database, with the name A, with the system catalog in the container specified. The user and temporary tablespaces will default to SMS tablespaces.

4. Drop the default user and temporary tablespaces, and create new tablespace containers by using the old Oracle containers.

   - CONNECT TO A
   - CREATE TABLESPACE USERS
     MANAGED BY DATABASE USING (FILE '/data/usrA.dbf' 512)
   - CREATE TEMPORARY TABLESPACE TEMP
     MANAGED BY DATABASE USING (FILE '/data/tempA.dbf' 138)
   - DROP TABLESPACE TEMPSPACE1
   - DROP TABLESPACE USERSPACE1

   Note: We created the new tablespaces before dropping the old. The order of creation did not matter for the user tablespace, but creation of temporary tablespaces must be done in this order. This is because there must exist at least one temporary tablespace within the database environment.

5. Create the LOB tablespace.

   - CREATE LONG TABLESPACE LOBSPACE
     MANAGED BY DATABASE USING (DEVICE '/dev/rlobA' 4000);

4.5 Log and Dump Devices

In DB2, as in Oracle, there are some files which collect the different events received by the database manager.

In Oracle, you can find log and dump file in locations specified by the config.ora and init.ora parameters:

- core_dump_dest
- background_dump_dest
- user_dump_dest

In DB2, you can find log and dump file in files specified in the database manager configuration parameter:

Diagnostic data directory path (DIAGPATH) = /u/jcb/sqllib/db2dump

Chapter 5. Data Types

This chapter covers the differences between the data types of Oracle 7 and DATABASE 2 for AIX Version 2. It discusses the data types that map directly from Oracle to DB2 and how to work with the data types that do not have a direct mapping.

The topics that are covered include:

Data-Type Cross Reference

Incompatible Data Types

DB2 User-Defined Types

5.1 Data Type Comparisons

The different data types managed by Oracle and DB2 are described in the following sections. The aim of this is discussion is to help point out what modifications are required to map the Oracle data types to DB2 data types.

5.1.1 Oracle Internal Data Types

Table 8 lists the Oracle data types. Each data type has a code which is used internally by Oracle. The data type code of a column is returned by the Oracle DUMP function. These codes are only for the internal use of Oracle, and they do not play a role in the conversion to DB2.

Table 8. Oracle Internal Data Types

Data Type	Description
VARCHAR2(n)	Variable length character string having maximum length n bytes. The maximum length is 2000.
NUMBER(p,s)	Number having precision ′p′ and scale ′s′. The precision can range from 1 to 38. The scale can range from -84 to 127.
LONG	Character data of variable length up to 2 GB or 2E31 - 1.
DATE	Valid date range from January 1, 4712 BC to December 31, 4712 AD.
RAW(n)	Raw binary data of length n bytes. Maximum n is 255 bytes.
LONG RAW	Raw binary data of variable length up to 2 GB.
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.
CHAR(n)	Fixed-length character data of length n bytes. Maximum n is 255. Default n is 1 byte.
MLSLABEL	Binary format of an operating system label. This data type is used primarily with Trusted Oracle.

The Oracle pre-compilers recognize additional data types to those listed in Table 8. These data types are called external data types, and they are shown in Table 9 on page 54.

Table 9. Oracle External Data Types	
Data Type	**Oracle Data Type**
CHARACTER(n)	CHAR(n)
NUMERIC(p,s), DECIMAL(p,s), DEC(p,s)	NUMBER(p,s)
INTEGER, INT, SMALLINT	NUMBER(38)
FLOAT(p), DOUBLE PRECISION, REAL	FLOAT
VARCHAR(n)	VARCHAR(n)
LONG VARCHAR	LONG

5.1.2 DATABASE 2 for AIX Version 2 Data Types

DB2 supports a large number of data types, including support for the following data types:

- BLOB - Binary Large Object
- CLOB - Character Large Object
- DBCLOB - Double Byte Large Object

There can by any number of BLOB, CLOB or DBCLOB columns in a DB2 table. DB2 makes it possible for users to define their own data types. This is done by creating a User-Defined Types (UDTs). Refer to 5.3, “DB2 User-Defined Types” on page 58 for further information about UDTs.

Table 10 gives an overall view of the different data types supported by DB2.

Table 10 (Page 1 of 2). DB2 Data Types	
Data Type	**Description**
INTEGER, INT	For a large integer. The range is -2147483648 to +214783647.
SMALLINT	For a small integer. The range is -32768 to +32767.
DOUBLE, DOUBLE PRECISION, FLOAT	For a floating-point number. The number can be zero, or it can range from -1.79769E+308 to -2.225E-307 or from 2.225E-307 to 1.79769E+308.
DECIMAL(p,s), DEC(p,s), NUMERIC(p,s), NUM(p,s)	For a decimal number.
CHARACTER(n), CHAR(n)	For a fixed-length character string, of length n, which may range from 1 to 254
VARCHAR(n), CHARACTER VARYING(n), CHAR VARYING(n)	For a variable character string, of length n, which may be range from 1 to 4000.
LONG VARCHAR	For a variable character string with a maximum length of 32700.
GRAPHIC(n)	For a fixed-length graphic string, of length n, which may range from 1 to 127.

Table 10 (Page 2 of 2). DB2 Data Types	
Data Type	**Description**
VARGRAPHIC(n)	For a variable graphic string, of maximum length n, which may range from 1 to 2000.
LONG VARGRAPHIC	For a variable graphic string with a maximum length of 16350.
DATE	For a date. A three-part value (year, month, and day).
TIME	For a time. A three-part value (hour, minute, and second).
TIMESTAMP	For a timestamp (year, month, day, hour, minute, second, microsecond).
BLOB	Binary Large OBject is a variable string measured in bytes that can be up to 2 GB long.
CLOB	Character Large OBject is a variable string measured in bytes that can be up to 2 GB long. A CLOB is considered to be a character string.
DBLOB	A Double Byte Large OBject is a variable string of double-byte characters that can be up to 1073741823 characters long. A DBLOB is be considered to be a graphic string.

5.1.3 Mapping Conversion

This section looks at each Oracle data type and translates it into a DB2 data type. Care should be taken when looking at each type since the name of the data type may be equivalent; however, the function may not be equivalent. The mapping or translation of the Oracle types has been broken down into four separate tables. These are:

- Number data type mapping (Table 11)
- Character data type mapping (Table 12 on page 56)
- Binary data type mapping (Table 13 on page 56)
- Date/Time data type mapping (Table 14 on page 57)

NOTE: In each table, the shaded rows indicate the data types in Oracle that are different from those in DB2.

5.1.3.1 Number Data Type

The NUMBER data type is used to represent many numerical data types in Oracle. These data types do not exist in DB2. Table 11 shows the function of Oracle numerical data types and the equivalent in DB2.

Table 11 (Page 1 of 2). Number Data Types	
Oracle Data Types	**DB2 Data Types**
FLOAT(p) default p = 128	**FLOAT**
REAL	**FLOAT**
DOUBLE PRECISION	DOUBLE PRECISION

Table 11 (Page 2 of 2). Number Data Types	
Oracle Data Types	**DB2 Data Types**
NUMBER(p,s) default p = 38	**NUMERIC(p,s) p <= 31**
NUMBER(p,0)	**INTEGER**
DECIMAL(p,s)	DECIMAL(p,s)
DEC(p,s)	DEC(p,s)
INT, INTEGER	INT, INTEGER
SMALLINT	SMALLINT

Be careful of the range value. If the value is out of the correct range, Oracle will still insert the row. However, DB2 will reject it by trying to insert a number greater than 32767 in a column which has the data type SMALLINT.

Also, Oracle rounds up numbers (n,0), whereas DB2 ignores all the numbers after the decimal point. For example, Oracle would round 3.8 up to 4, while DB2 would return the value 3.

5.1.3.2 Character Data Types

The maximum size for the CHARACTER data type in DB2 is 254. With Oracle, the maximum size is 255.

Table 12. Character Data Types		
Oracle Data Types	**DB2 Data Types**	**Comments**
VARCHAR2(n)	**VARCHAR(n)**	**1 <= n <= 2000**
VARCHAR(n)	VARCHAR(n)	1 <= n <= 2000
CHAR VARYING(n)	CHAR VARYING(n)	1 <= n <= 2000
CHARACTER VARYING(n)	CHARACTER VARYING(n)	1 <= n <= 2000
CHARACTER(n)	CHARACTER(n)	n <= 254
CHAR(n)	CHAR(n)	n <= 254
CHARACTER(255)	**VARCHAR(255)**	
CHAR(255)	**VARCHAR(255)**	
LONG VARCHAR	LONG VARCHAR	
LONG	**CLOB(2 GB)**	

5.1.3.3 Binary Data Types

Oracle supports only one column of the LONG type per table. DB2 does not have this limitation since any table may contain multiple columns of the equivalent DB2 type. The mapping of these types can be seen in Table 13.

Table 13 (Page 1 of 2). Binary Data Types		
Oracle Data Types	**DB2 Data Types**	**Comments**
RAW(n)	CHAR(n) FOR BIT DATA	n <= 254

Table 13 (Page 2 of 2). Binary Data Types		
Oracle Data Types	**DB2 Data Types**	**Comments**
RAW(255)	VARCHAR(255) FOR BIT DATA	
LONG RAW	BLOB(2 GB)	

5.1.3.4 Date/Time

The Oracle data type DATE indicates year, month, day, hour, minute, and second. It does not correspond to the type DATE of DB2 because the DATE data type in DB2 contains only the year, month and day. In DB2, the data type TIMESTAMP contains all the information from the year through to the seconds and fractions of a second. The mapping of the Oracle DATE will depend on the amount of information you require in the DB2 data type.

To extract the DATE of Oracle, you must use a format function, such as `to_char`.

- The TIMESTAMP data type format in DB2 is YYYY-MM-DD-HH24.MI.SS.mmmmmm. The function in Oracle which creates this format is called to_char(a_date,'YYYY-MM-DD-HH24.MI.SS').
- The DATE data type format in DB2 is MM/DD/YYYY. The function in Oracle which creates this format is to_char(a_date,'MM/DD/YYYY')
- The TIME data type format in DB2 is HH24:MI:SS. The function in Oracle which creates this format is to_char(a_date,'HH24:MI:SS')

There are additional formats for date and time to the ones shown in the above examples.

Table 14. Date/Time Data Types		
Oracle Data Types	**DB2 Data Types**	**Comments**
DATE	**TIMESTAMP**	**All information. Be careful of the format**
DATE	**DATE**	**Only the date**
DATE	**TIME**	**Only the time**

5.2 Data Type Incompatibilities

Some types, such as ROWID (block.row.tuple), are unique to Oracle. We can assimilate some of these types. For example, DB2 has a unique row identifier (RID), but it is not specified by the user in the Data Manipulation Language (DML) and is therefore slightly different from the ROWID in Oracle.

Some of the of Oracle data types have no equivalent in DB2, for example, ROWID and MLSLABEL. But for each of them, there is a type that can be considered to be an equivalent.

The data types that pose the most problem during the conversion from Oracle are:

- NUMBER
- DATE

DB2 also offers some additional data types that have no corresponding types in Oracle. These data types include:

- GRAPHIC
- VARGRAPHIC
- LONG VARGRAPHIC
- TIME
- TIMESTAMP
- CLOB
- DBLOB
- BLOB

5.3 DB2 User-Defined Types

DB2 offers the possibility to define your own data types. This is achieved by using the features in DATABASE 2 for AIX Version 2 to create a User-Defined Type (UDT), which is a unique type based on standard DB2 data types.

We can use this functionality to create types corresponding to the Oracle data types or to create our own types that more closely match the values being stored.

As an example, this capability could be used to create a type that matches the Oracle ROWID type. This would be done by using the following statement:

```
CREATE DISTINCT TYPE ROWID AS CHAR (18) WITH COMPARISONS
```

Another use of the UDT capability would be the creation of a type, such as INCH. To do this, you would use the statement:

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

These UDTs are strongly typed. This means that a UDT, such as INCH, cannot be compared to an INTEGER, even though it is based on the integer type. To do such a comparison, a casting function must be used. By creating the User-Defined Function (UDF) with comparisons, the casting functions will automatically be generated. For further information on User-Defined Functions and Types you should refer to the documentation supplied with DATABASE 2 for AIX Version 2.

5.4 Data Type Conversion Example

This example translates an Oracle table to a DB2 table. The definition of the table can be seen in Figure 18 on page 59, and it involves the data types `date`, `number`, `raw`, and `char` (character).

```
Oracle> cat DDL_for_oracle.sql
drop table tab;
create table tab (
        a1 date,
        a2 number(5,3),
        a3 raw(255),
        a4 char(5));
insert into tab values (SYSDATE,15.86,HEXTORAW('AABBCCDDEEFF'),'TITI');
insert into tab values (SYSDATE,11.186,HEXTORAW('0011223344FF'),'TOTO1');
exit;
Oracle> sqlplus scott/tiger @DDL_for_oracle.sql
```

Figure 18. The File DDL_for_oracle.sql

The steps in the table translation are as follows:

1. Use the file extract_insert_for_db2.sql, as shown in Figure 19, to extract the INSERT order. This will create a file containing the statements that will be used to insert the data into DB2. The statements will be saved in the file insert_for_db2.sql.

```
Oracle> cat extract_insert_for_db2.sql
set head off;
set line 1048;
spool insert_for_db2.sql;

select 'INSERT INTO tab VALUES (''' || rowid || ''','''  ||
        to_char(a1,'YYYY-MM-DD-HH24.MI.SS') || ''','  ||
        a2 || ',x''' || RAWTOHEX(a3) || ''','''  || a4 || ''');'
from tab;
exit;
Oracle> sqlplus scott/tiger @extract_insert_for_db2.sql
Oracle> cat insert_for_db2.sql

INSERT INTO tab VALUES ('00000502.0000.0001','1995-04-17-16.59.24',15.86,
x'AABBCCDDEEFF','TITI ');
INSERT INTO tab VALUES ('00000502.0001.0001','1995-04-17-16.59.24',11.186
,x'0011223344FF','TOTO1');
```

Figure 19. Files extract_insert_for_db2.sql and insert_for_db2.sql

2. Modify the file DDL_for_oracle.sql, as shown in Figure 18, to create the DDL_for_db2.sql. The file is created with the assistance of the information contained in Table 11 on page 55, Table 12 on page 56, Table 13 on page 56, and Table 14 on page 57.

```
$ cat DDL_for_db2.sql
drop table tab;
drop distinct type ROWID;
create distinct type ROWID as character(18) with comparisons;
create table tab (
        rowid ROWID,
        a1 timestamp,
        a2 numeric(5,3),
        a3 varchar(255) for bit data,
        a4 char(5));
quit;
```

Figure 20. The File DDL_for_db2.sql

3. Run DDL_for_db2.sql to create the table in DB2.
4. Run insert_for_db2.sql to load the data into the table.

```
$ db2 connect to SAMPLE
$ db2 -tvf DDL_for_db2.sql
$ db2 -tvf insert_for_db2.sql
```

Figure 21. Run the DDL Statements

Chapter 6. Database Schema

This chapter covers the differences between database schema objects in Oracle and DB2. It also covers the differences that exist between DB2 and Oracle in the following database objects:

- Tablespaces
- Tables, Views and Indexes
- Clusters and Constraints
- Aliases
- Schema
- Users, Packages and Privileges
- Stored Procedures and Triggers
- Catalogs

6.1 Tablespaces

A database can be logically divided into tablespaces. A tablespace can be used to help partition system information from user information, or it could be used to help partition information across physical storage media to help improve performance.

6.1.1 Oracle Tablespaces

In Oracle a tablespace can exist in one of two states. The tablespace can be online or offline. When the tablespace is online, it is accessible by the users, and the data contained within that tablespace may be updated. In the offline state, the tablespace in not available to the users. A tablespace may be places in the offline state so that the administrator can perform administrative tasks.

Each Oracle database contains a tablespace called SYSTEM. The SYSTEM tablespace is created when the database is created. It contains the data dictionary of the database tables and objects. This tablespace cannot be taken offline because it is required for access to any of the database objects.

Tablespaces are made up of data files. It is possible to increase the size of a tablespace by adding additional data files to it. For further information on tablespaces and datafiles, refer to 4.1.1, “Oracle 7 Physical Structure” on page 35

6.1.2 DB2 Tablespaces

DATABASE 2 for AIX Version 2 divides the database into tablespaces in the same way that Oracle does. However, DATABASE 2 for AIX Version 2 supports two different types of tablespaces. The first type is a System Managed Space (SMS) and the second a Database Managed Space (DMS). The DMS tablespace is similar to an Oracle tablespace. It is possible to use both types of tablespaces within a single database. It is possible to increase the size of a DMS tablespace in a similar way that Oracle is able to increase the size of its tablespaces. However, there are some differences in the way space is allocated for the

database objects within the tablespace. This is covered in more detail in 4.1.2, “DATABASE 2 for AIX Version 2 Physical Structure” on page 37.

6.2 Tables, Views and Indexes

Within a tablespace, the tables, views and indexes exist. In general, data is stored within the table. This may be user data or system data. Indexes are created to speed-up access times on table information. Views are used to look at existing tables in different ways. A view looks like another table; however, it is actually referencing existing tables.

This section discusses each of these topics and outlines the differences in the implementation of them in Oracle and DB2.

6.2.1 Tables

A table is a collection of data, logically arranged into rows and columns. The basic way in which you use tables in Oracle and DB2 is the same. However, there are some differences in the way that tables can be defined and how the table information is stored within the database.

Oracle Tables

A single table in Oracle is unable to span across multiple tablespaces. The only division of an object across tablespaces is the ability to place the indexes of a table into a separate tablespace. There is no limit on a the number of rows, indexes or columns in a table other than the fact that each table may only contain one column of data type LONG.

DB2 Tables

In DB2, a table will not usually span across a tablespace. Like Oracle, it is possible to store the indexes for a table in a separate tablespace. However, a DB2 table supports multiple columns of the equivalent data types to the Oracle LONG data type. It is also possible to store these types in a separate tablespaces. Also, like Oracle, there are no limits on the number of rows, indexes or columns that a table may contain.

The other difference between Oracle and DB2 is in the system catalogs that are created when a database is created. In DB2, most of these tables are read-only. However, some of the tables may be updated by a user. These tables involve statistical information, and the values are used by DB2 to optimize SQL statements. By updating these tables, it is possible to simulate different environments and test the performance of an SQL statement under the set conditions.

6.2.2 Views

A view is an alternative representation of data from one or more tables. The data is not duplicated since the view accesses the original table data. To the user, a view appears to have all the properties of a table. The view may include any number of columns from multiple tables or other views. Because a view accesses the data contained within other tables, the only space it requires is for its definition.

Any statement performed on a view is actually being performed on the table or on tables the view accesses. This means that if a view is used in an SQL update statement, the table is actually being updated. Similarly, if the table is changed, then these changes are reflected in the view.

Both Oracle and DB2 provide for a 'with check option' clause when creating a view. This will check that any updates or inserts of data from a view conform to the select clause of the view. Oracle also allows a specific constraint to be specified when the view is created. DB2 does not allow you to specify a constraint at the view level. However, the *with check* clause in DB2 may be specified as either local or cascading. A local check clause will only check that the view's select clause is satisfied when performing an SQL operation, while a cascading check clause will also check the selects on any of the tables or views that the current view is dependent upon.

6.2.3 Indexes

An *index* is a list of the locations of rows, sorted by the contents of one or more specified columns. Indexes are used to speed-up access to a table, but they can be also used to keep a check on the rows of a table. For example, to restrict the entry of duplicate values in a column or group of columns.

Oracle Indexes

When processing an SQL request, Oracle can choose to use some or all of the defined indexes to perform the request. An index is created on one or more columns of a table and is automatically maintained and used by Oracle. Changes to table data are automatically incorporated into all the indexes defined for that table. Each index in Oracle is contained within its own segment. This segment is allocated when the index is created. You are able to specify the storage parameters for allocation of the index. These parameters include the number of extents and tablespace name. You can also refer to a schema name or a cluster when creating an index. An Oracle index can contain up to sixteen columns, and there can be any number of indexes on a table.

DB2 Indexes

Indexes in DB2 are the same in function as an Oracle index. The main differences between indexes in Oracle and DB2 are that the indexes in DB2 for a particular table are stored within a single file when using an SMS tablespace. Also, a DB2 index may be specified as unique. This uniqueness means that the column or columns that make up the index may not contain duplicates. Oracle specifies the uniqueness of a column or columns as a constraint on a table. The index can be defined when the table is created or at a later time.

6.3 Clusters and Constraints

A cluster is a concept available only in the Oracle environment. It involves multiple tables that have one or more columns in common that share the same data. These common columns are physically stored in the same datablocks on disk.

A constraint is a rule or restriction on the values of a table's data that is enforced at the database level, rather than at the object or application level. By

using constraints, we can guarantee that values of a column conform to the rules we specify for that table, before the value is inserted, updated or removed.

6.3.1 Oracle Clusters and Constraints

Clustering is a method used to store table data. A cluster is a group of one or more table columns from multiple tables that are physically stored together because they share common data and may often be used together. Disk access improvements may be seen due to the related rows being physically stored in the same data blocks.

The columns included in a cluster are called cluster keys. It is recommended that you should only cluster tables that are frequently joined on the cluster key columns in SQL statements. This is because clustering multiple tables can improve the performance of joins, but is likely to reduce the performance of full table scans and insert or update statements that modify the cluster′s key values.

Integrity constraints are defined at table level and stored as part of the table′s definition in the database′s data dictionary area. In this way, all database applications must adhere to the rule, and any changes to a constraint need to be made at the database level and will take effect on every application.

When a constraint is defined for an existing Oracle table and some existing data does not satisfy the constraint, the constraint is not created. The statement is rolled back and an error message returned. Oracle supports the following integrity constraints:

NOT NULL	These columns must contain a value other then null.
UNIQUE	Duplicate values are not allowed in this column or set of columns.
PRIMARY KEY	Duplicate and null values are not allowed in these columns.
FOREIGN KEY	Each value in these columns must match a value in a related table′s unique or primary key column. Creating a foreign key also creates an index on its columns.
CHECK	Check for values that do not satisfy the logical expression of the constraint specified.

Oracle Keys

In Oracle, the term ″key″ is used to indicate a column or set of columns included in a constraint definition to describe the relationships between the different tables and columns of the database. There are different kinds of keys:

Primary key	Is a unique identifier of the rows in a table.
Unique key	Identifies the columns included in a unique constraint definition.
Foreign key	Identifies the columns included in a referential integrity constraint definition. A referential integrity constraint can be defined as a constraint that guarantees that the values in the foreign key also exist in the primary key′s table.
Referenced key	Identifies the unique or primary key of the table that is referenced by a foreign key.

6.3.2 DB2 Clusters and Constraints

As previously stated, DATABASE 2 for AIX Version 2 does not support clusters. This section will concentrate on constraints and referential integrity.

DATABASE 2 for AIX Version 2 referential integrity constraints similar to those found in Oracle. The difference in Oracle constraints and DB2 constraints lies in the actual definition of the constraint. DB2 defines referential integrity constraints and table check constraints.

Referential integrity constraints allow us to define any required constraints between or within tables. The referential integrity constraint is a relationship between a primary and a foreign key.

Table check constraints are conditions defined as part of the table definition, and they verify that changed or inserted data in a table does not violate the conditions. They also are used to specify conditions which are checked for each row of a table. A check constraint designates the values or range of values that a specific column of a base table can contain. These constraints may be turned off by using the `set constraints off` statement. This will place the table in a CHECK PENDING state. For more information on DB2 constraints, refer to the *DB2 for AIX Version 2 Information and Concepts Guide*.

6.4 Synonyms and Aliases

Both Oracle and DB2 provide synonyms and aliases. However, they appear to mean different things in each environment. This section looks at both synonyms and aliases in each environment and describes their function.

An *alias* is an indirect method of referencing a table or view. It is commonly used in SQL statements to allow independence from the qualified names of these database objects. If a table or view changes, only the alias definition must be changed. An alias can be used in any SQL statement except for the constraint definition. An alias must be unique and can only refer to a table or view within the same database. An alias can be defined for a table, view or alias. A table or view can be referred to in an SQL statement by its name or by an alias that has been defined for its name. Thus, aliases can be thought of as alternate names for tables and views. When a database has been cataloged, it can be referred to by an alias. Two databases with the same name on different nodes can be cataloged with different aliases . The Oracle *alias* concept corresponds to the *correlation* or *synonym* concept in DB2.

6.4.1 Oracle Aliases and Synonyms

An Oracle alias and synonym are similar in function. They differ in the way they are defined and in the longevity of the alias or synonym.

Aliases

An alias in Oracle is an alternative name assigned to an existing database object during the execution of an SQL statement. The alias exists only during the SQL statement. An alias may refer to objects, such as tables, views, columns, or values within the SQL statement.

In Oracle, there is no CREATE ALIAS; so you must specify the alias for a table, view or column during the SQL statement definition, and that alias will be valid until the end of the SQL statement execution.

Synonyms

An Oracle synonym is different from the alias because you can create a synonym for a table, view, sequence, procedure, function, package, or snapshot. The synonym can be private or public and will last until it is dropped with the DROP SYNONYM statement.

Like the alias, a synonym does not require any storage other than for its definition. A public synonym is owned by all the users in the group public, while a private synonym is owned by a specific user and is only available to that user.

6.4.2 DB2 Aliases and Synonyms

DATABASE 2 for AIX Version 2 provides both the alias capability and the synonym capability as defined by Oracle. An alias in DB2 is the same as a synonym. The statements CREATE ALIAS and CREATE SYNONYM are identical. What Oracle refers to as an alias is called a *correlated reference* in DB2

Correlated Reference (Oracle Alias)

A correlated reference in DB2 can apply to a table, column, view or value within an SQL statement. Like Oracle aliases, this value only holds for the duration of the SQL statement. For detailed information on correlated references and their use, refer to the *DB2 SQL Reference for common servers Version 2*.

Synonyms

As mentioned previously, a synonym in DB2 may also be referred to as an alias within the DB2 documentation. Either term refers to a synonym that functions in much the same was as an Oracle synonym.

A DB2 synonym or alias is created with the CREATE [ALIAS | SYNONYM] statement. When an alias is created, it is available to all users connected to that database. This alias or synonym remains effective until it is dropped by the DROP [ALIAS | SYNONYM] statement.

6.5 Schema

A schema is a logical classification of all the objects owned by a user. A schema is associated to a database user and is used to perform more than one DDL operation on a set of objects owned by a user. A schema is not linked to any particular tablespace. A single schema may be spread across multiple tablespaces, and a single tablespace may contain multiple schemas.

6.5.1 Oracle Schema

In Oracle, the concept of a schema is used to indicate a collection of objects, such as tables, views, synonyms, packages, and so on, that are related to a database user.

6.5.2 DB2 Schema

DATABASE 2 for AIX Version 2 has a similar concept of schema as found in Oracle. All objects in DATABASE 2 for AIX Version 2 belong to a user. If an object belongs to another user, it may be referenced by placing the user's login ID before the object name. For example, if there is a table 'addr1' owned by the user 'fred', another user may reference this table by using the name 'fred.addr1'. You may also prefix an object with a schema name in the same way as a user.

Schema in Oracle can be used to grant a user access to a group of objects. To accomplish this under DB2, you may create a new operating system group, and rather than adding objects to a schema, you can grant the appropriate permission to this group. When a user requires access to the object, you may add the user's ID to the group.

6.6 Users and Groups

This section will discuss the ways in which Oracle and DB2 manage users, groups, roles, and privileges.

In the DB2 environment, a user directly equates to an operating system user that has been created by the system administrator. To access the database, the user only needs to set up the environment variables. Oracle defines a user within the database environment itself. This means that a user at the operating system level needs to set up the environment variables as well as enter another user ID and password for Oracle.

This difference in user definition also leads to differences in the way privileges are allocated to a user. DB2 allows privileges to be allocated to a user or to a group. The group is again a direct mapping to the operating-system groups. Oracle does not use groups; instead, it defines roles. The role is a group set of privileges that can then be granted to a user. More details on the way users, groups and roles are defined by the database managers can be found in 8.2, "Users, Groups and Roles" on page 108.

6.6.1 Oracle Users and Groups

Oracle creates a set of users when a database instance is created. These users are for system administration. The default users are:

INTERNAL The internal user is allowed to startup and shutdown the database.

SYS The sys user is the owner of the base tables and views of the data dictionary.

SYSTEM The system user owns all additional tables and views that contain administrative information used by the Oracle tools.

Oracle also create a default group called *PUBLIC*. This group provides access to specific schema objects, like tables and views, to users belonging to this special group. Every Oracle user belongs to the public group by default. Membership to this group allows users to select from some of the data dictionary tables, grant or revoke privileges to the public group, create links and synonyms, and assign to the public group. As mentioned earlier, the public group is a special case. Oracle would normally use roles to distribute privileges to multiple users.

6.6.2 DB2 Users and Groups

DATABASE 2 for AIX Version 2 also has a default user in the database environment. This user is the instance owner. When a DATABASE 2 for AIX Version 2 instance is created, an operating system user ID is required. This user becomes the instance owner and inherits all the database administration privileges.

DB2 also has three default groups. These groups are not mapped to any operating system group when the instance is created. It is up to the instance owner to decide the operating system group that will inherit the database authorizations that they provide. These groups are:

SYSADM Performs all system control tasks and database administration activities.

SYSCTRL Performs operation on system resources. Has no access to database data.

SYSMAINT Performs database maintenance operations.

Apart from these groups, there is also the normal database user and a special authority, called DBADM.

DBADM Performs database administration only on the database where the authority has been granted.

For a complete reference of the authorizations that SYSADM, SYSCTRL, SYSMAINT, and DBADM provide, refer to figure Table 20 on page 110.

6.7 Packages

Packages under DB2 and Oracle are very different in the database objects that they describe. Oracle refers to a package as a collection of objects, while DB2 uses the term package to describe the information stored in a database about an SQL statement.

An Oracle package is a grouping of possibly related procedures, functions, cursors, and variables, and it stores them as a package in the database. This allows you to grant access to this package and in doing so, to grant access to all of its contents. Oracle packages are created by using the `CREATE PACKAGE` command, and the information about the package is stored in the system tables.

DATABASE 2 for AIX Version 2 defines a package as the information stored in the database that is required to process specific SQL statements from a single source file or application. The package is created when the source file is precompiled during the binding of an application program.

The binding process is covered in Chapter 9, “Applications” on page 117.

6.8 Stored Procedures and Triggers

A stored procedure is a way to enable multiple users to perform an operation using the same piece of code. This allows us to minimize the risk of data corruption through incorrect coding and provide a single point of change should the code need to be changed. When a user wishes to run this procedure or function, they call it, and the database manager will retrieve the code and handle

its execution. The implementation of stored procedures is different between the Oracle and DB2 environments.

A trigger is a piece of code that will be executed automatically by the system upon a specified event, such as an insert statement, update statement or delete statement. Like stored procedures, the implementation of triggers is different between the two database environments.

The following sections look briefly at the implementation of stored procedures and triggers in the two database environments. For more details on the DATABASE 2 for AIX Version 2, refer to Chapter 9, “Applications” on page 117 and the *DATABASE 2 Application Programming Guide for common servers Version 2*.

6.8.1 Oracle Stored Procedures and Triggers

A stored procedure in Oracle logically groups a set of SQL and other PL/SQL programming languages statements together to perform a specific task. It can be executed interactively (for example, in an SQL*DBA environment) called by name from an application or called from another procedure or trigger Oracle also allows you to create and store functions which are similar to procedures but which return a value.

Oracle stored procedures and functions are stored within the database itself and are written in PL/SQL. Because these may contain PL/SQL they would have to be re-written for DB2., as PL/SQL is not supported outside Oracle.

Types of triggers Oracle triggers can have special features that are defined at execution time, such as the number of times the trigger is to be executed. A trigger is invoked upon an SQL insert, update or delete statement. There are also several types of Oracle triggers. They are:

Row Triggers	Executed one for each row changed in a table.
Statement Triggers	Executed once for each statement executed.
Before Triggers	Executed just before the statement′s execution.
After Triggers	Executed just after the statement′s execution.

It is possible to combine the before and after triggers with the row and statement triggers. Finally, a trigger may be in an enabled state, or it may be disabled.

6.8.2 DB2 Stored Procedures and Triggers

In DB2, a stored procedure may be written in almost any supported programming language. Some of these include C, COBOL and FORTRAN. A DATABASE 2 for AIX Version 2 stored procedure is actually stored at the database server, but outside the database. When a stored procedure is created, it is bound to the database. This binding process creates a package in the database that contains information about the external procedure. Once a procedure has been bound to the database, it is accessible by the database clients.

Like Oracle, DB2 also provides trigger capabilities. The triggers are stored in the database and automatically called by the database manager, when required. Triggers ensure that the business rules defined at the database creation time are always respected, and if one of the rules changes, the only change needed is made in the trigger statement, not in every application that uses it. In addition

to triggers, DB2 provides constraints. A constraint may be able to replace a trigger or a trigger and stored procedure combination.

As in Oracle, DB2 provides for before triggers and after triggers on SQL insert, delete or update operations. Also, a trigger can be directed to perform the operation for each row that is affected or for the entire statement.

By comparison, the triggers in both database managers are similar in functionality and differ mainly in the way a trigger is defined. Stored procedures are quite different in the way they are implemented, but perform a similar function in the two database environments.

6.9 Catalogs

System catalogs are the tables that constitute the database′s data dictionary. The catalog tables contain all the information about the database structures, objects and definitions.

6.9.1 Oracle Catalogs

In Oracle, the system catalogs are another name for data dictionary. System catalogs are a set of tables and views that are used in read-only reference about the database. The catalogs store information about the logical and physical structure of the database. The owner of the data dictionary tables is user SYS, and the owner is the only user that can modify the tables.

The data dictionary is composed of base tables and views. The base tables are accessed by the Oracle engine only and cannot be directly accessed by users. The views provide a summary of the base table′s contents and simplify the contents. This allows users to view the types of information stored in the system catalogs. A complete list of the system catalogs has been provided in Table 27 on page 163.

6.9.2 DB2 Catalogs

DATABASE 2 for AIX Version 2 has the same concept of system catalogs. These tables perform much the same function as in Oracle. The system catalog are again divided into the base tables and a collection of simplified views.

The majority of the system catalog tables in DATABASE 2 for AIX Version 2 are read-only. However, there are a number of tables that the user is able to update. These tables contain statistical information used to optimize SQL statements before they are executed. For a complete list of DB2 system catalogs, refer to Appendix B, “IBM SQL Reserved Words” on page 159 Table 28 on page 166.

Chapter 7. SQL Language Elements

This chapter compares the SQL language of Oracle 7 to that of DATABASE 2 for AIX Version 2. There are examples of how to perform the different conversions necessary and a discussion about the differences on the following topic areas:

Functions

DDL and DML Syntax

Constraints

Joins

Cursors

Reserved Words

DB2 Special Registers

7.1 Functions

This section discusses the functions available in Oracle and how they map to DB2 functions. The conversion topics have been broken into the following topics:

- Oracle functions that map directly to DB2
- Functions that have differences in syntax or output
- Oracle functions that have no DB2 equivalent
- Additional DATABASE 2 for AIX Version 2 functions

In many cases, the functions are similar in the task they perform, but may be cataloged under a different name. A complete list of Oracle 7 functions and the equivalent DATABASE 2 for AIX Version 2 function can be found in Table 26 on page 161.

For further information on topics covered in this chapter, you should refer to the documentation supplied with your copy of DB2. Descriptions, such as that contained in *Database 2 SQL Reference for common server Version 2* , should be referenced for further information on the DB2 functions.

Many of the functions used in Oracle map directly to a DB2 function. However, care needs to be taken with functions that involve the following:

- Displaying output of numeric data
- Use of implicit data type conversion
- Use of the DATE data type or format

These areas are discussed in detail later in this chapter.

7.1.1 Compatible Functions

Table 15 on page 72 is a listing of the Oracle functions that map directly to a DB2 function. These functions are equivalent in name, syntax, functionality and output. Therefore, they require no modification in their use.

Table 15. Functions That Map from Oracle Directly to DB2	
Oracle 7	**DATABASE 2 for AIX Version 2**
ABS	ABS or ABSVAL
ASCII	ASCII
CHR	CHR
COUNT	COUNT
MAX	MAX
MIN	MIN
LENGTH	LENGTH
POWER	POWER
SIGN	SIGN
SOUNDEX	SOUNDEX
SUM	SUM
USER	USER

Some Oracle functions have an equivalent DB2 function, but the function may be listed under a different name in DB2. The Oracle SQL could be changed to use the DB2 name. Alternatively, a User Defined Function (UDF) could be created in DATABASE 2 for AIX Version 2 to map internally to the same function name used by Oracle.

For example, to translate a character string to uppercase in Oracle, the UPPER function is used. In DB2, it is the UCASE function. A UDF, called UPPER, could be created in DB2 and based on the DB2 UCASE function, as follows:

```
CREATE FUNCTION UPPER (CHAR(40)) RETURNS CHAR(40)
       SPECIFIC UPPER SOURCE UCASE
```

The UDF would have the same name as the Oracle function, and so it allows the Oracle code to remain unchanged.

Table 16 lists the Oracle functions that map to equivalent DB2 functions which have different names.

Table 16. Oracle Functions with Different Names in DB2	
Oracle 7	**DATABASE 2 for AIX Version 2**
LENGTHB	LENGTH
LOWER	LCASE
SYSDATE	CURRENT DATE
UPPER	UCASE
VSIZE	LENGTH

Most arithmetic functions in Oracle and DB2 are compatible. There are some syntactical differences which are discussed later. The main difference is in the output that is displayed by the function. For example, given the following SQL statements:

```
CREATE TABLE tab1 (col1 NUM(3,2))

INSERT INTO tab1 values(1.0)

SELECT AVG(col1) from tab1
```

We would find that the select statenebt in Oracle would return the value 1, while DB2 would return the value 1.0. To get the same displayable output from DB2, the INT function could be used as follows:

```
SELECT AVG(INT(col1)) from tab1
```

An example where the output has greater differences is in the CEIL function. Given the statement:

```
SELECT CEIL(15.7) from tab1
```

Oracle would return 16. However, DB2 would return +1.60000000000000E+001. To get the same displayable output from DB2, the INT function could be used in the following manner:

```
SELECT INT(CEIL(15.7)) from tab1
```

As shown in the previous examples, you need to take care when dealing with numerical output. The output may need to be modified via DB2 functions, such as INT or DEC. Otherwise, the program may need to be altered to account for the possible changes in output size.

Internal arithmetic operations in Oracle 7 and DB2 are similar; it is just the manner in which the results are displayed that needs to be taken into consideration during the conversion.

Table 17 lists the functions that are equivalent in syntax and function; however, if the SQL were to remain unchanged, the output displayed by the statement would be different in the two environments.

Table 17. Functions with Different Output Formatting

Oracle 7	**DATABASE 2 for AIX Version 2**
AVG	AVG
CEIL	CEIL or CEILING
COS	COS
EXP	EXP
FLOOR	FLOOR
LN	LN or LOG
SIN	SIN
SQRT	SQRT
TAN	TAN

If the following Oracle functions are used, they may need to be modified when converting to DB2 due to differences in the output produced, the syntax used or how the function operates:

- CONCAT

  The function, syntax and output is the same in DB2 as it is in Oracle. The difference is that Oracle has implicit data conversion and will concatenate

columns of character and number types automatically. In DB2, the columns must be compatible. For example, if you wish to concatenate a character and a number, you must explicitly do the data conversion. An example of this would be as follows:

```
SELECT CHAR(number1) CONCAT char2 from tab1
```

Implicit and explicit data conversion is discussed later in this chapter.

- INSTR and INSTRB

  These Oracle functions map to LOCATE in DB2. The syntax is quite different in Oracle. In Oracle, the syntax is `INSTR(char1,char2[,n[,m]]),`, while in DB2, the syntax is `LOCATE(source_string,search_string,[n])`.

- LOG

  The function is similar; however, Oracle allows the specification of base and number. In DB2, the function `LOG(n)` returns the natural logarithm of ′n′, while `LOG10(n)` will return base 10 logarithm of the argument ′n′. The Oracle function `LOG(m,n)`, returns base ′m′ of the value ′n′. The format of the output is also different. As previously mentioned, you can use the `INT` function to produce identical output from DB2.

- LTRIM

  The operation and output is identical; however, the syntax is different between the two environments. In Oracle, the syntax is `LTRIM(char[,set])`, while in DB2 the syntax is `LTRIM(char)`. In Oracle, if the optional parameter ′SET′ is not used, then the two functions are the same.

- MOD

  The function, syntax and output are the same for this function on both Oracle and DB2. The syntax is `MOD(m,n)`, and the the difference is in that Oracle will return ′m′ if ′n′ is 0, whereas DB2 will issue a message saying division by zero attempted.

- NVL

  This Oracle function maps to the `NULLIF` in DB2. The main consideration here is that the data types must be compatible in DB2, while implicit data type conversion is done in Oracle.

- REPLACE

  The function and output are the same; however, the requirement of parameters is different. Given the syntax `REPLACE(char, search_string [replacement_string ])`, the third parameter is optional in Oracle, while if it is not specified in DB2, an error message will be returned.

- ROUND

  The function is the same; the difference is in the syntax and output. The syntax in DB2 is `ROUND(n,m)`. In Oracle, `m` is optional. If `m` is not specified in DB2, an error message is returned. If the output is a displayable numeric result, then, as discussed previously, the use of the `INT` or `DEC` functions may format the displayable output the same way Oracle does.

- RTRIM

  The operation and output are identical; however, the syntax is different between the two environments. In Oracle, the syntax is `RTRIM(char[,set])`, while in DB2 the syntax is `RTRIM(char)`. If the optional Oracle parameter ′SET′ is not used, then the two functions are the same.

- SUBSTR

  The operation, syntax and output are the same in both environments. The difference is in the allowable parameters. Given the syntax SUBSTR(char,m,[n]), Oracle allows m to be a positive or negative number, while in DB2, the value must be positive.

- SUBSTRB

  This Oracle function is performed by SUBSTR in DB2.

- TRANSLATE

  The function and output are the same, but the syntax is different. Oracle uses the syntax TRANSLATE(char,from,to), while the syntax used by DB2 is TRANSLATE(char,to,from[,pad]) . The pad parameter in DB2 will make sure the to parameter is added to the same length as the from parameter. Oracle does not have this capability.

- TRUNC

  The function is the same in both environments and the output requires the same considerations previously discussed for arithmetic formatting of output. Given the syntax TRUNC(n[,M]), the parameter m is optional in Oracle, but it is mandatory for DB2.

7.1.2 Incompatible Functions

A lot of the compatibility issues come from data type conversion. Oracle supports both implicit and explicit conversions. Implicit data type conversion is when the database manager automatically converts one data type into another so that it is compatible for a particular operation. Explicit data type conversion is when the user must convert the data type using a function, before the values are used. Automatic conversion of values from one data type to another is not part of standard SQL. In the *Oracle 7 Server SQL language Reference Manual*, Oracle recommends that you use explicit data type conversion, rather than implicit conversions, for the following reasons:

- SQL statements are easier to understand when you use explicit data type conversion functions.
- Automatic data type conversion can have a negative impact on performance, especially if the data type of a column value is converted to that of a constant rather than the other way around.
- Implicit conversion depends on the context in which it occurs and may not work the same way in every case.
- Algorithms for implicit conversion are subject to change across software releases and among Oracle products. Behavior of explicit conversions is more predictable.

DB2 uses explicit conversion in all cases, except when a column of the data type 'DATE' is assigned to a character string variable or string column. When this occurs, the conversion to a string representation is done automatically.

For comparisons or concatenation operations in DB2, the data types must be compatible. There are a number of DB2 functions that can be used to convert data types. Some of these include 'CAST', 'CHAR', 'DECIMAL', 'DIGITS', and 'HEX'. These would then enable two different data types to be compared or concatenated.

The following Oracle conversion functions do not directly map to an equivalent DB2 conversion function. However, a similar function may exist, or could be created, using user-defined types and user-defined functions.

- CHARTOROWID and ROWIDTOCHAR

  CHARTOROWID converts a value from the 'CHAR' or 'VARCHAR2' data type to the Oracle data type ROWID. As discussed in the data types chapter, ROWID is specific to Oracle. When the rows are unloaded from Oracle, columns of this data type are converted to the DB2 type, 'CHAR(18)'.

  The following example shows how we can create a UDT called ROWID and a UDF called CHARTOROWID. This will allow applications that use the data type ROWID and the functions CHARTOROWID and ROWIDTOCHAR to be migrated without change.

```
CREATE DISTINCT TYPE rowid as CHAR(18) WITH COMPARISONS;
CREATE FUNCTION CHARTOROWID (VARCHAR(18)) returns rowid
              source rowid (VARCHAR(18));

CREATE TABLE tab1 (c1 rowid);
INSERT INTO tab1 VALUES('aa');
SELECT * FROM tab1 WHERE c1=CHARTOROWID('aa');
```

- HEXTORAW and RAWTOHEX

  To map these function to DB2, you would need to define a column as 'FOR BIT DATA', and then you may use the HEX or X functions. For example:

  1). The following could be done:

```
X'FFFF' representing the bit pattern '1111111111111111'
X'4672616E6B' representing the VARCHAR pattern of the
              ASCII string 'Frank' in the SQL.
```

  2). Either of the following insert statements could be used given that C1 and C2 are defined as:

```
CREATE TABLE tab1 (
                   C1 CHAR(2),
                   C2 CHAR(2) FOR BIT DATA
                  )

INSERT INTO tab1 VALUES('aa',x'6161')
INSERT INTO tab1 VALUES('bb','bb')
```

  3). The following four select statements produce the same result given the above table definition:

```
SELECT * FROM tab1 WHERE x'6161' = c2
SELECT * FROM tab1 WHERE 'aa' = c2
SELECT * FROM tab1 WHERE c1 = c2
SELECT * FROM tab1 WHERE HEX(c1) = HEX(c2)
```

  Again, it is possible to create a UDT, called HEXTORAW, to enable applications to be converted to DB2 unchanged. Also, there is the possibility of using the HEX function available in DB2 which returns the hexadecimal representation of a value.

- TO_CHAR

  TO_CHAR has many functions associated with it. Some of which are no longer relevant in DB2, while others can be accomplished using other DB2 functions, such as DIGITS. DIGITS will convert a number into a character string. The DATE data type is implicitly converted by DB2 into a character

string. To use the formatting capabilities of the TO_CHAR function, you would need to write a UDF or modify the application code to perform the formatting.

- TO_DATE

  The TO_DATE function has many functions associated with it. Again, some of them would no longer be required in DB2. To convert a number data type in to the date format, the DB2 DATE function could be used. If a character string was to be converted, you may be able to use the CASE function.

- TO_NUMBER

  This function is performed in DB2 by either the DECIMAL function or by the CAST function. A UDT called TO_NUMBER could be created based on these functions. This would allow the conversion of the SQL without changes.

As discussed in data types chapter, the DATE data type in Oracle is not like the DATE data type in DB2. The Oracle DATE data type indicates the year through to the second. This is similar to the DB2 TIMESTAMP datatype. The DB2 DATE data type contains only the year, month and day. The mapping of the DATE data type depends on the amount of information that is going to be required in the new data type. To extract the DATE of Oracle into a DB2 format the Oracle function TO__CHAR is used when unloading the rows.

When migrating to DATABASE 2 for AIX Version 2, a decision needs to be made on the TO_CHAR values required so the data that will be loaded into DB2. The applications could be updated to use the DB2 DATE functions or a UDF called TO_CHAR could be created which performs the required function.

Functions such as ADD_MONTHS, LAST_DAY, MONTHS_BETWEEN and NEXT_DAY could have UDF′s built, based on existing DB2 functions, so that SQL would not have to be rewritten.

Date arithmetic in DB2 is also calculated differently. The following date functions exist in DB2, but do not have an equivalent in Oracle. These functions are:

- DATE
- DAY
- DAYNAME
- DAYOFWEEK
- DAYOFYEAR
- DAYS
- HOUR
- MICROSECOND
- MINUTE
- MONTH
- MONTHNAME
- SECOND
- TIME
- TIMESTAMP
- TIMESTAMP_ISO
- TIMESTAMPDIFF
- WEEK
- YEAR

Addition and subtraction of the DATE data type is simple in DB2. For example:

```
CREATE TABLE tab1 (col1 DATE)
INSERT into tabl1 values (DATE(CURRENT TIMESTAMP))

SELECT * from tab1
```

If this select statement returned the value 04/2/1995, we could perform the following update statement followed by another select

```
UPDATE tab1 set col1 = col1 + 14 DAYS

SELECT * from tab1
```

This select statement would now return 05/10/1995.

Further information and examples can be found in the *Database 2 SQL Reference for common server Version 2*

The remaining functions that have not already been covered and have no direct translation to a DB2 function are listed below. However, a UDF or UDT could be created in DB2 which would allow the SQL to be converted unchanged. An example of creating a User-Defined Function (UDF) is shown in Appendix E, "User-Defined Functions" on page 169.

The functions that may require a UDF are:

- COSH
- DUMP
- GLB
- GREATEST
- GREATEST_LB
- INITCAT
- LEAST
- LEAST_UB
- LPAD
- LUB
- NEW_TIME
- NEXT_DAY
- RPAD
- SINH
- STDDEV
- TANH
- UID
- USERENV
- VARIENCE

7.1.3 Additional DATABASE 2 for AIX Version 2 Functions

There are a number of functions available in DATABASE 2 for AIX Version 2 that are not in Oracle. These functions are summarized in Table 18.

Table 18 (Page 1 of 4). Functions Available in DATABASE 2 for AIX Version 2

Function name	Description
ACOS	Returns the arccosine of the argument as an angle expressed in radians
ASIN	Returns the arcsine of the argument as an angle expressed in radians

Table 18 (Page 2 of 4). Functions Available in DATABASE 2 for AIX Version 2	
Function name	**Description**
ATAN	Returns the arctangent of the argument as an angle expressed in radians
ATAN2	Returns the arctangent of x and y coordinates, specified by the first and second arguments respectively, as an angle expressed in radians
BLOB	Casts from source type to BLOB with optional length
CHAR	Returns a string representation of the source type
CLOB	Casts from a source type to CLOB with optional length
COALESCR or VALUE	Returns the first non-null argument in the set of arguments
COT	Returns the cotangent of the argument where the argument is an angle expressed in radians
DATE	Returns a date from a single input value
DAY	Returns the day part of a value
DAYNAME	Returns a mixed-case character string containing the name of the day, for the day portion of the argument based on what the locale was when DB2 start was issued
DAYOFWEEK	Returns the day of the week in the argument as an integer value in the range 1-7, where 1 represents Sunday
DAYOFYEAR	Returns the day of the year in the argument as an integer value in the range 1-366
DAYS	Returns an integer representation of a date
DBCLOB	Casts from a source type to DBCLOB with optional length
DECIMAL or DEC	Returns decimal representation of a number, with optional precision and scale
DEGREES	Returns the number of degrees converted from the argument expressed in radians
DIFFERENCE	Returns the difference between the sounds of the words in the two argument strings as determined using the SOUNDEX function. A value of zero means the strings sound the same
DIGITS	Returns the character string representation of a number
DOUBLE or DOUBLE_PRECISION	Returns the floating-point representation of a number
EVENT_MON_STATE	Returns the operational state of a particular event monitor
FLOAT	Same as DOUBLE
GRAPHIC	Casts from source type to GRAPHIC with optional length
HEX	Returns the hexadecimal representation of a value
HOUR	Returns the hour part of a value
INSERT	Returns a string where arg3 bytes have been deleted from arg1 beginning at arg2 and where arg4 has been inserted into argument beginning at arg2
INTEGER or INT	Returns the integer representation of a number
LEFT	Returns a string consisting of the leftmost arg2 bytes in arg1

Table 18 (Page 3 of 4). Functions Available in DATABASE 2 for AIX Version 2	
Function name	**Description**
LOCATE	Returns the starting position of the first occurrence of arg1 within arg2. If the optional third argument is specified, it indicates the character position in arg2 at which the search is to begin. If arg1 is not found within arg2, the value 0 is returned
LONG_VARCHAR	Returns a long string with optional length
LONG_VARGRAPHIC	Casts from source type to LONG_VARGRAPHIC with optional length
MICROSECOND	Returns the microsecond(time-unit) part of a value
MINUTE	Returns the minute part of a value
MONTH	Returns the month part of a value
MONTHNAME	Returns a mixed case character string containing the name of month for the month portion of the argument that is a date or timestamp, based on what the locale was when db2start was issued
NULLIF	Returns NULL if the arguments are equal, else returns the first argument
QUARTER	Returns an integer value in the range 1 to 4 representing the quarter of the year for the date specified in the argument
RADIANS	Returns the number of radians converted from argument which is expressed in degrees
RAISE_ERROR	Raises an error in the SQLCA. The sqlstate returned is indicated by arg1. Second argument contains any text to be returned
RAND	Returns a random floating point value between 0 and 1 using the argument as the optional seed value
REPEAT	Returns a character string composed of arg1 repeated arg2 times
RIGHT	Returns a string consisting of the rightmost arg2 bytes in arg1
SECOND	Returns the second (time-unit) part of the value
SMALLINT	Returns the small integer representation of a number
SPACE	Returns a character string consisting of arg1 blanks
TABLE_NAME	Returns an unqualified name of a table or view based on the object name given in arg1 and the optional schema name given in arg2. It is used to resolve aliases
TABLE_SCHEMA	Returns the schema name portion of the two part table or view name given by the object name in arg1 and the optional schema name given in arg2. It is used to resolve aliases
TIME	Returns the time from a value
TIMESTAMP	Returns a timestamp from a value or a pair of values
TIMESTAMP_ISO	Returns a timestamp in the ISO format converted from the IBM format. If the argument is a date, it inserts zero for all the time elements. If the argument is a time, it inserts the value of CURRENT DATE for the date elements and zero for the fractional time element
TIMESTAMPDIFF	Returns an estimated number of intervals of type arg1 based on the difference between two timestamps
VARCHAR	Returns VARCHAR representation of the first argument

Table 18 (Page 4 of 4). Functions Available in DATABASE 2 for AIX Version 2	
Function name	**Description**
VARGRAPHIC	Returns a VARGRAPHIC representation of the first argument. If a second argument is present, it specifies the length of the result
WEEK	Returns the week of the year in the argument as an integer value in the range of 1 to 53.
YEAR	Returns the year part of a value

7.2 SQL Comparison

This section will compare the Data Manipulation Language (DML) and Data Definition Language (DDL) of Oracle 7 and DATABASE 2 for AIX Version 2. You should refer to the *Database 2 SQL Reference for common server Version 2* and the *Oracle 7 SQL Language Reference Manual* for further reference on the material covered in this chapter.

The following areas of discussion are covered:

- Some general considerations
- Optional parameters used by Oracle commands
- Operators, Expressions and Conditions
- The SQLCA, SQLDA and ORACA structures
- Oracle hints and recommendations on what to migrate over to DB2
- Comparison of the DML and DDL of Oracle and DB2
- Joins
- Dynamic SQL and Cursors

There are many similarities between the DML and DDL of Oracle and DB2; however, there are also some differences. Many of the difference are due to Oracle extensions to standard SQL and Oracle optional parameters. If these optional parameters and extensions are removed, the DML and DDL would map more closely to DB2.

There are some precompiler parameters in Oracle that can be used to flag the Oracle extensions. These include:

- FIPS FLAGGER

  If this is set to FIPS=YES, the precompiler will flag the Oracle extensions to standard SQL in the embedded SQL programs.

- MODE

  This specifies whether the program observes Oracle extensions or ANSI standards. The default is ORACLE. If this is set to MODE=ANSI or MODE=ANSI14, the embedded SQL will map more closely to the SQL used by DB2. For example, a 'no rows found' error will return a SQLCODE +1403 in Oracle if the MODE=ORACLE. If MODE=ANSI14, the sample error will return the SQLCODE of +100. The SQLCODE of +100 matches the code returned in DATABASE 2 for AIX Version 2.

Another consideration may be the use of host arrays in Oracle. If MODE=ANSI14 is specified, you cannot use host arrays in Oracle, and this will map to DB2. However DB2 does provide for host arrays when using CLI or compound SQL. If host arrays are used in Oracle, the application may need to be modified to remove them and to also remove the code which handles phantom 'no rows found' situations. That is, if you define an array for 100 and only 20 rows are returned, the SQLCA SQLCODE will be set to 'no rows found' return code.

- DBMS.

  If this is set to DBMS=V7, NULL processing maps closer to DB2 .

If the above options are already set to these values, the DML, DDL and the flow of the program will be close to the way it will work in DB2.

Another consideration when converting to DB2 is the length of identifiers. For example, table name, index name, view name, and so on. The identifier in Oracle has a maximum length of 30; in DB2, the maximum length is 18.

Some of the Oracle DML and DDL may use optional parameters that have a different meaning or are done a different way in DB2. These include schema, @dblink, alias, and synonym.

- SCHEMA

  In ORACLE V6, there was no distinction between a user and the collection of objects owned by the user. The name of an object could be qualified by the name of the user who owned it.

  In Oracle 7, the term schema now describes the collection of objects owned by a user. Every user owns a schema in which objects can be created. The name of that schema is the same as the name of the user. The name of an object can be qualified by the schema in which the object exists. For example, the table EMP in the schema of the user SCOTT can be defined by SCOTT.EMP.

  Schema in DB2 may be described as being similar to the schema concept in Oracle. However, there are no `CREATE SCHEMA` statements as there are in Oracle. DB2 schema may map to a user or may be defined when objects such as tables are created. In DB2, before doing any work in a database, the first action is to connect to the database we wish to work in; you can connect to the database by specifying a user ID and password. All activity is then done under this user ID in the database. Alternativly you may prefix the database object with the user ID or schema name. To perform actions in another database, you must connect to that database. A schema in DB2 is defined as a logical grouping for database objects. When a database object is created, it is assigned to one schema, which is determined by the name of the object. For example, the table EMP in the schema of the user PETE can be defined by PETE.EMP.

- @DBLINK

  @DBLINK is the complete or partial name of a database link to a remote database. If this is omitted, Oracle assumes that we are using a local database. You use this in conjunction with *Oracle SQL*NET* product. The optional '@DBLINK' parameter in Oracle is not required by DB2. DB2 automatically resolves the database location by using its database directories. This allows DB2 administrators to move the database without having to change any DDL, DML or SQL.

- SYNONYM

  A synonym in Oracle is referred to as either a synonym or an alias in DB2. In DB2, you can continue to use the term synonym if you wish, and you may either use the `create alias` or `create synonym` statements.

  In Oracle, a synonym can be used in SQL statements, such as `select, insert, update, delete, explain table, lock table, audit, noaudit, grant, revoke` and `comment`.

  In DB2, a synonym can be used anywhere a table or a view is allowed.

- ALIAS

  Some Oracle statements may specify an alias within an SQL statement; this is the same for DB2. However, in DB2, it is refer to as a "correlation name."

- Compound SQL

  Some statements in Oracle use PL/SQL which may need to be converted into compound SQL if you wish to keep the same functionality. An example of this might be:

```
EXEC SQL BEGIN COMPOUND NOT ATOMIC
SELECT c1, CASE c2,c3,c4
    WHEN c2 > :avail THEN RAISE_ERROR ('ZZ000','too much')
    WHEN c4 = NULL THEN RAISE_ERROR ('ZZ001','missing')
    END
  INTO :hv1,:hv2,:hv3,:hv4 from tab1 where....
END COMPOUND
```

For further details, please refer to the *Database 2 SQL Reference for common server Version 2*

7.2.1 Operators, Expressions and Conditions

All of the Oracle operators, expressions and conditions map to DB2 These include operators like NULL, +, -, *, /, LIKE, and between. There are two exceptions to this: the Oracle DECODE expression and the Oracle precompiler parameter 'DBMS' and the effect it has on processing NULL values.

The DECODE expression is comparable to the CASE function in DB2. In Oracle, an expression like,

```
DECODE (deptno, 10, 'SALES', 20, 'RESEARCH')
```

could map to the DATABASE 2 for AIX Version 2 statement:

```
CASE deptno
     WHEN 10 THEN 'SALES'
     WHEN 20 THEN 'RESEARCH'
END
```

This expression can be used in a number of places. This includes its use with select statement shown below:

```
SELECT empno, empname FROM emp
WHERE (CASE WHEN salary=0 THEN NULL ELSE comm/salary END) > 0.25
```

If the Oracle precompiler expression is DBMS=V7, both Oracle and DB2 will be compatible in the processing of NULL values. If the Oracle parameter is DBMS=V6, you can select a column which is null into a field which does not have a null indicator, without any error being set. This is not the way that

Oracle 7 or DATABASE 2 for AIX Version 2 process the NULL values, and it may cause you problems.

7.2.2 SQLCA Structure

The SQLCA and its use is mostly compatible between Oracle and DB2. Applications should be able to use the DB2 SQLCA without change. However, there are a few considerations that need to be taken into account.

- SQLERRD(5)

  In Oracle, this parameter holds the character position at which a parse error began. In DB2, it contains the number of rows updated or deleted as the result of a constraint enforcement.

- SQLWARN5

  DB2 does not use this parameter. In Oracle, it is used when a `create procedure, function, package,or package bodycommand` fails.

- INCLUDE SQLCA.

  This includes the SQLCA layout into both Oracle and DB2 applications. If you have the Oracle precompiler parameter 'MODE' set to ANSI13 or ORACLE, you must include this command; otherwise the include is optional. In DB2 applications, the inclusion of SQLCA is required.

- SQLSTATE

  In DB2, the SQLSTATE is included at the end of the SQLCA record. DB2 automatically provides information to both the SQLSTATE and the SQLCODE records. In Oracle, you are required to specify the precompiler parameter 'MODE' equal to ANSI or ANSI14, and then declare a field called SQLSTATE. In Oracle, to return values to SQLCODE and SQLSTATE you must also declare the field SQLCODE in the DECLARE section.

  SQLCODE stores error codes and "not found" conditions, while SQLSTATE stores error and warning codes. SQLSTATE uses a standard coding schema and is the preferred status variable under SQL92. Under SQL92, SQLCODE is retained for compatibility with SQL89 and is likely to be removed from future versions of the standard. For further information on Oracle SQLCODE and SQLSTATE, refer to the *Programmer's Guide to the Oracle Precompilers, Release 1.6*.

DB2 provides additional information in the SQLCA record.The additional fields are SQLERRP, SQLERRD(2), SQLERRD(4) and SQLWARNA. Further information on DB2 SQLCA can be found in *DATABASE 2 for AIX Version 2 SQL Reference*.

7.2.3 SQLDA Structure

SQLDA provides similar function in DB2 and Oracle, but the way it is used by the application is different. You will need to modify the application if you use the Oracle SQLDA. An example and further information can be found in 7.5, "CURSOR and DYNAMIC SQL" on page 102. In short, SQLCODE +236 in DB2 replaces the Oracle requirement to test the relationship between 'sqln' and 'sqld'. This is usually done to determine if the SQLDA contains enough SQLVARs for the `describe` statement. Applications coded for Oracle often use the check `"sqlcode<>0"` to check for warnings. You should bind your DB2 packages with SQLWARN NO and your applications should work as they would in Oracle.

7.2.4 ORACA Structure

The SQLCA handles standard SQL communications, while the ORACA handles ORACLE communications. When you need more information about run-time errors and status changes than SQLCA provides, the ORACA can be used. It contains an extended set of diagnostic tools. However, use of the ORACA adds to run-time overhead.

In DB2, you have the SQLCA structure. Any applications that reference ORACA will need to be converted to use the SQLCA. DB2 provides additional information in the SQLCA in fields SQLERRP, SQLERRD(2), SQLERRD(4) and SQLWARNA. Further information on DB2 SQLCA can be found in the *DATABASE 2 for AIX Version 2 SQL Reference*.

7.2.5 Oracle Hints

Hints are used in Oracle so that you can force the cost-based optimizer to use your chosen execution plan. The optimizer hints are hardcoded into the applications for specific `update`, `delete` and `select` statements. Hints allow you to specify use of the cost-based or rule-based optimizer, the access or join method and the goal of the cost-based optimizer. If hints are specified incorrectly, Oracle will ignore them, but will not issue an error. The RULE hint, along with the rules based optimizer, will not be available in future versions of Oracle.

If hints are not used, the cost-based optimizer looks at catalog statistics gathered by the ANALYZE utility to decide the access path. You can see what decision it has made by using the `explain plan` command.

If you use hints in your applications, they will need to be removed when converting to DB2.

DB2 also uses a cost-based optimizer. It makes a decision on the best access path to use by using catalog statistics gathered by the RUNSTATS utility. You can see what path the optimizer has chosen by using the VISUAL explain tool.

It is possible to perform “what if” scenarios by influencing the optimizer in its decision process, much like the hints of Oracle. This can be achieved by updating the statistics in the SYSSTAT catalogs. You can also influence the optimizer in the application by using ′FOR READ ONLY ′ or ′FOR UPDATE ONLY′ parameters in the cursor definitions. If you believe data returned for a `select` statement will not change, you can specify `optimize for x rows` to tell the optimizer how many rows will be returned by this statement.

The DATABASE 2 for AIX Version 2 Starburst optimizer has different levels of optimization, which are set via the `set current query optimization` statement for dynamic SQL. For static SQL, the `QUERYOPT` option is used during the preprocessing or binding stage. This tells the optimizer to perform very little optimization for simple SQL or to use the maximum amount of optimization for very complex SQL. The Starburst optimizer will even rewrite the query to achieve the best possible access path.

7.2.6 SQL Syntax Comparison

Table 19 is a list of common Oracle statements. It includes both the Data Manipulation Language (DML) and the Data Definition Language (DDL) found in Oracle. The table briefly describes the associated DATABASE 2 for AIX Version 2 command or statements that are equivalent in function to the Oracle command or statement. Many of the commands or statements may not have an equivalent due to differences in definitions or functionality of the database. Chapter 6, "Database Schema" on page 61 should be read before this chapter to obtain an understanding of where these differences lie.

The commands and statemets that require more detailed discussion are covered later in the chapter. References to these sections have been included in Table 19.

Table 19 (Page 1 of 4). ORACLE and DB2 DML/DDL Comparisons

Oracle 7 DML or DDL	**DATABASE 2 for AIX Version 2 DML or DDL**
alter cluster	Clustering is not used in DB2.
alter database	Much of this function is provided by the `update database configuration` command.
alter function	DB2 requires you to drop the function and then re-create it.
alter index	Much if this function is done with the `alter tablespace` statement.
alter package	Not available in DB2 as the concept of a package is different.
alter procedure	Procedures are bound to the database in DB2. To modify a procedure, it must be re-bound to the database.
alter profile	Profile information is altered using the operating system and the command, `update database configuration`.
alter resource cost	Resources are controlled by the operating system and the following DB2 commands: • `update database configuration` • `update database manager configuration`
alter role	Roles are not used in DB2. However, it is possible to grant and revoke permissions on groups to provide a similar function to roles.
alter rollback segment	To manage logs in DB2, you can use the command, `update database configuration`. Much of the log management in DB2 is automatic.
alter sequence	DB2 does not provide the the sequence capability. Refer to 7.2.6.10, "CREATE SEQUENCE" on page 93 further information.
alter session	Some of this function is not required in DB2; other functionality is found in various DB2 commands. For example: • `set sql_trace` is implemented in `update database manager configuration`. • `set optimizer goal` is implemented in `set current query optimization`. • `advise` is implemented in `list indoubt transactions`.
alter snapshot	This function is provided by the product, *Data Propagator Relational*.
alter snapshot log	This function is provided by the product *Data Propagator Relational*.
alter system	The function provided here is found in the operating system as well as the following DB2 commands: • `force application` • `update database configuration manager` • `connect to ... in exclusive mode`

Table 19 (Page 2 of 4). ORACLE and DB2 DML/DDL Comparisons

Oracle 7 DML or DDL	DATABASE 2 for AIX Version 2 DML or DDL
alter table	Refer to 7.2.6.1, “ALTER TABLE” on page 89.
alter tablespace	Refer to 7.2.6.2, “ALTER TABLESPACE” on page 90.
alter trigger	DB2 requires a trigger to be dropped and then re-created if changes are to be made.
alter user	Users in DB2 are at the operating system level and are modified using operating system commands. Access to a database and its tables may then be granted to the user.
alter view	DB2 requires that the view be dropped and re-created for any modifications.
audit/noaudit	Monitoring may be turned on using the `update database manager configuration`. Triggers may also be used to provide audit information.
close	Refer to 7.5, “CURSOR and DYNAMIC SQL” on page 102.
comment on	The statement syntax is similar. DB2 allows you to comment on an alias, column, constraint, distinct type, function, index, package, table, tablespace, trigger, or view.
commit	Commit will work in DB2 as it does in Oracle. However, the Oracle force parameter is not implemented. To commit or rollback in-doubt transactions, use the `list indoubt transactions` command.
connect	Refer to 7.2.6.3, “CONNECT” on page 90.
create cluster	Clustering is not used in DATABASE 2 for AIX Version 2.
create controlfile	Refer to 7.2.6.4, “CREATE CONTROLFILE” on page 90.
create database	Refer to 7.2.6.5, “CREATE DATABASE” on page 91.
create database link	DB2 resolves database location through its catalogs. To catalog a database, use the `catalog` commands.
create function	Refer to 7.2.6.6, “CREATE FUNCTION” on page 92.
create index	Refer to 7.2.6.7, “CREATE INDEX” on page 92.
create package	Refer to 7.2.6.8, “CREATE PACKAGE” on page 92.
create package body	This command is not required for the same reasons discussed in 7.2.6.8, “CREATE PACKAGE” on page 92.
create procedure	Refer to 7.2.6.9, “CREATE PROCEDURE” on page 93.
create profile	User profiles are at the operating system level and some information is in the database configuration. To update database configurations, use the `update database configuration` command.
create role	Roles are not used in DATABASE 2 for AIX Version 2. By using groups and privileges, it is possible to get the functionality of a role.
create rollback segment	This is not required in DB2 as the information is automatically managed by the database manager.
create schema	A schema is not used in DB2 as Oracle defines them. Refer to Chapter 6, “Database Schema” on page 61 for information on the schema concept.
create sequence	Refer to 7.2.6.10, “CREATE SEQUENCE” on page 93.
create snapshot	This function is provided by the product *Data Propagator Relational*.
create snapshot log	This function is provided by the product *Data Propagator Relational*.
create synonym	In DB2, a synonym and alias are equivalent. Chapter 6, “Database Schema” on page 61 describes this in more detail. Either `create synonym` or `create alias` may be used in DB2.

Table 19 (Page 3 of 4). ORACLE and DB2 DML/DDL Comparisons	
Oracle 7 DML or DDL	**DATABASE 2 for AIX Version 2 DML or DDL**
create table	Refer to 7.2.6.11, “CREATE TABLE” on page 93.
create tablespace	Refer to 7.2.6.12, “CREATE TABLESPACE” on page 94.
create trigger	Refer to 7.2.6.13, “CREATE TRIGGER” on page 94.
create user	Adding a user in DB2 is done via: • Add a user at the operating system level. • Modify profile to include the instance profile (db2profile). • Grant appropriate privilege on the database or databases.
create view	Views are similar in DB2. However the Oracle replace, constraint and force options are not available.
declare cursor	Refer to 7.5, “CURSOR and DYNAMIC SQL” on page 102.
declare database	This is replaced by the statement `connect to database`.
declare command	DB2 uses a `prepare` command that removes the requirement for this command.
declare table	DB2 performs this function during the precompile or bind time if the ′validate′ parameter is specified.
delete	Similar in DB2 The Oracle parameter ′alias′ may be replaced in DB2 using the ′as correlation-name′ capability.
describe	Refer to 7.5, “CURSOR and DYNAMIC SQL” on page 102.
drop cluster	Clustering not used in DB2.
drop database link	DB2 uses catalogs to resolve database locations. To uncatalog a database, use the `uncatalog` command.
drop function	Refer to 7.2.6.14, “DROP FUNCTION” on page 95.
drop index	DB2 also uses the `drop index` statement.
drop package	This statement exists in DB2. However, as the definition of a package is different care should be taken. Refer to 7.2.6.8, “CREATE PACKAGE” on page 92 and Chapter 6, “Database Schema” on page 61 for information on packages.
drop procedure	Not used in DB2.
drop profile	Not used in DB2. User profiles are at the operating system level.
drop role	Roles do not exist in DB2
drop rollback segment	Not used in DB2.
drop sequence	Not used in DB2.
drop snapshot	Not used in DB2.
drop snapshot log	Not used in DB2.
drop synonym	May use either `drop synonym` or `drop alias` in DB2.
drop table	The DB2 `drop table` statement is similar to Oracle if Oracle specified the ′cascade constraints′ parameter.
drop tablespace	Refer to 7.2.6.15, “DROP TABLESPACE” on page 96.
drop trigger	The `drop trigger` statement works the same as in Oracle.
drop user	Not required as users are at the operating system level.
drop view	The `drop view` statement works the same as in Oracle.
execute	Refer to 7.5, “CURSOR and DYNAMIC SQL” on page 102.
execute immediate	Refer to 7.5, “CURSOR and DYNAMIC SQL” on page 102.

Table 19 (Page 4 of 4). ORACLE and DB2 DML/DDL Comparisons	
Oracle 7 DML or DDL	**DATABASE 2 for AIX Version 2 DML or DDL**
explain plan	The explain facility is similar. You should issue the `set current explain snapshot` to enable the facility and then view with the Visual Explain tool. Alternatively you may use the `db2expln` or `dynexpln` tools.
fetch	Refer to 7.5, “CURSOR and DYNAMIC SQL” on page 102.
grant	Refer to 7.2.6.16, “GRANT” on page 96.
insert	Refer to 7.2.6.17, “INSERT” on page 97.
lock table	Refer to 7.2.6.18, “LOCK TABLE” on page 97.
open	Refer to 7.5, “CURSOR and DYNAMIC SQL” on page 102.
prepare	Refer to 7.5, “CURSOR and DYNAMIC SQL” on page 102.
rename	Not used in DB2. The same effect may be produced by the use of aliases.
revoke	Refer to 7.2.6.19, “REVOKE” on page 97.
rollback	Refer to 7.2.6.20, “ROLLBACK” on page 97.
savepoint	This is not used in DB2. DB2 will commit or rollback based on a Unit of Work (UOW).
select	Refer to 7.2.6.21, “SELECT” on page 98.
set role	Roles are not used in DB2. The concept of a role is discussed in Chapter 6, “Database Schema” on page 61.
set transaction	Refer to 7.2.6.22, “SET TRANSACTION” on page 99.
truncate	Not used in DB2.
type	Refer to 7.2.6.23, “TYPE” on page 99.
update	The DB2 `update` statement is no different to the Oracle command.
var	Refer to 7.2.6.24, “VAR” on page 100.
whenever	The DB2 `whenever` statement is similar to Oracle. However, the Oracle. optional parameters ’stop’ and ’do’ are not available. They could be replaced using the ’goto’ option.

7.2.6.1 ALTER TABLE

The syntax of the `alter table` statement is similar in both environments. However, there are some differences in the parameters used.

As space management is handled at the tablespace level, there is no use for the parameters ’PCTFREE’ and ’PCTUSED’ in the DB2 alter `alter table` statement. Also the parameters ’INITRANS’ and ’MAXTRANS’ are not used in the `alter table` statement because these types of values are set at the database or tablespace level. Storage allocation is handled at the tablespace level in DB2; so the parameters ’STORAGE’ and ’ALLOCATE EXTENT’ are also not required by the DB2 `alter table` statement.

Constraints on tables may be turned on and off in DB2 by using the `set constraints` statement. To enable and disable a trigger, you would need to use the `drop trigger` and `create trigger` statements. These statements replace the ’ENABLE’ and ’DISABLE parameters in the Oracle `alter table` statement.

Oracle defines the uniqueness of a column or group of columns at the table level. In DB2, this is done by assigning a unique index to the table. Because of this difference, the `alter table drop` statement in Oracle is not used in DB2

Finally, while Oracle allows a '(' to be added after the 'ADD' parameter in the `alter table` statement, this will cause error `SQL0104N` to be generated in DB2.

7.2.6.2 ALTER TABLESPACE

Tablespaces in Oracle and DB2 are very different in design and in the way that they are managed by the database manager. DB2 has two types of tablespaces, while Oracle only uses one type of tablespace.

The Oracle alter tablespace will perform functions that may be performed in DATABASE 2 for AIX Version 2 using different commands. To bring a tablespace online or take it offline, the DB2 `quiesce tablespace` command would be used. It is possible to make a backup of a tablespace in DB2 by using the `backup database` command. This will take a backup of the database or tablespace(s) while the system is either online or offline.

Finally, to increase the size of a tablespace, it is possible to add a container in DB2. If a container is added to a DB2 tablespace, the distribution of data will automatically be re-balanced across all the containers. Access to the tablespace is not restricted during this re-balancing process.

7.2.6.3 CONNECT

Connecting to a database is conceptually different in DATABASE 2 for AIX Version 2 and Oracle 7. An Oracle 7 server controls only one database; so a user cannot switch databases within the instance, but must connect to another server. DB2 may contain multiple databases or cataloged remote databases.

A user may connect to any database they have been granted access to without being concerned about its physical location. Also, a user may connect to another database using a different user ID if they know the correct password. An example of this is shown below:

```
CONNECT TO databaseA
SELECT .....
INSERT .....
CONNECT TO databaseB USER UserID USING password
SELECT .....
UPDATE .....
COMMIT
```

7.2.6.4 CREATE CONTROLFILE

This command is used in Oracle for a number of reasons. These include:

- Change the name of a database
- Create a new control file, if all copies have been destroyed
- Change the maximum number of logs, datafiles, and so on

Oracle recommends taking a full backup of all files in the database before and after you issue this command.

DATABASE 2 for AIX Version 2 does not used this command; however, similar information is contained in the database configuration file and database directories. DB2 does not allow you to rename a database, but you can catalog

an existing database under a different name. Both the new name and the old name would access the database. To change parameters, such as the maximum number of logs and so on, the command, update database configuration can be used. To look at the current settings of the parameters, use the get database configuration for <database-name> command.

7.2.6.5 CREATE DATABASE

The format the this command is similar in both environments. However, where the command is issued and the end result is different, as there are conceptual differences between an Oracle database and a DB2 database. In addition to the create database command, Oracle requires several other steps to create a database or instance. DB2 allows a single instance to contain multiple databases. Each database is created with the create database command, which is issued like any other command.

Due to these basic differences between databases in Oracle and DB2, you will need to look closely at the database mapping and logging before starting a conversion. Decisions, such as should I have multiple instances on a single machine and a single database in each instance or a single instance with multiple databases, should be considered. You may even choose multiple instances and multiple databases. The reason for this may include giving different people administrative authority on the different instances or using one instance as a production system while the other is for development.

When the create database command is issued in DB2, the database logs are created, the database directories are updated, and three tablespaces are created. If you wish, you can specify the name, location, size, and type (SMS or DMS) of these tablespaces, or leave them to default. If left to default, DB2 will create three SMS tablespaces: a catalog tablespace called SYSCATSPACE, a default user tablespace called USERSPACE1 and a temporary tablespace called TEMPSPACE1.

Many of the parameters used by Oracle in the create database command are set in the database configuration file of DATABASE 2 for AIX Version 2. To view these values for a database, you can use the command, get database configuration for database-name. To update the values, you would use the command update database configuration for <database> using <parameter> <value>.

Most of the information specified in the Oracle create database parameters may be viewed and modified using the above commands. This includes all the logging information, archiving details and character sets. Other parameters, such as 'DATAFILE', may be implemented in the DB2 create database command by specifying the tablespace containers to be used or by using the 'ON' value to specify where the database is to be created.

An example of the DB2 create database command would be:

```
CREATE DATABASE payroll
       ON /database/pay ALIAS pay
       NUMSEGS 30
```

In Oracle, if you use the create database command on an existing database, it will destroy the existing database. DB2 will return error SQL1005N, saying that a database of that name already exists. You will need to drop the database before trying to re-create it.

7.2.6.6 CREATE FUNCTION

The concept of a User-Defined Function is similar in both Oracle and DB2. The differences lie in the language used to create them and in the way the database manager stores them.

In Oracle, the function is defined as a PL/SQL block; in DB2 it is a C or C++ program or another User-Defined Function (UDF). An example of a user-defined function can be found in Appendix E, "User-Defined Functions" on page 169.

Due to the language differences, an Oracle function must be re-written using one of the supported DATABASE 2 for AIX Version 2 languages. Also, if the Oracle function contains SQL statements, further problems exist since DB2 Functions cannot contain SQL statements. In these cases, it would be necessary to convert the Oracle function into a DB2 stored procedure, compound SQL or incorporate it into the application.

7.2.6.7 CREATE INDEX

This syntax of the DDL below should work in both Oracle and DB2.

```
CREATE INDEX indexname ON
       { tablename (column [ASC|DESC] [, column [ASC|DESC]] ...)}
```

Oracle has the 'DESC' parameter for compatibility. This would normally cause the index to be a descending index. However, in Oracle, indexes are always ascending. In DB2, it is possible to create the index as an ascending (ASC) or descending (DESC) index.

While Oracle allows you to specify the tablespace that will contain the index being created, DB2 does this when the table is created. If this option is used in any `create index` statement, both the `create index` and `create table` statements will need to be modified.

In both Oracle and DB2, you can specify up to 16 columns in the index. DB2 allows for a maximum size of 255 total bytes for an index.

The optional Oracle parameter 'NOSORT' is compatible with the DB2 parameter 'INDEXSORT' in the database configuration file. Also, the optional Oracle parameter 'CLUSTER' is not applicable to DB2 because clustering is not available in DB2. Storage parameters are specified in the DB2 `create tablespace` statement and in the file, directory or logical volume specifications.

DB2 allows you to specify a unique index on a table. Oracle recommends that you do this via a 'UNIQUE' constraint in the `create table` statement. However, you can still use the `create unique index` statement in Oracle, but it is not documented in the *Oracle 7 SQL Language Reference Manual* .

7.2.6.8 CREATE PACKAGE

In Oracle, a package is an encapsulated collection of related procedures, functions, variables, constants, exceptions, and cursors. It is an alternative to creating procedures and functions as stand-alone schema objects. They are used to help you organize the application development process better by grouping together related functions and procedures.

The package specifies functions, procedures and their parameters. The package body specifies the PL/SQL to be executed when the function or procedure is executed by an application. Having a separate package body allows you to alter

the package body by using the create package or replace package statement and to then recompile it via the alter package compile body command. This eliminates the requirement to recompile the applications that use the function or procedure.

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 with the prep statement or by binding a precompiler-generated bind file using the bind command.

The create package command is not available in DB2 for the reason shown above. The Oracle packages should be changed to stand-alone functions or procedures. In this form, it is possible to re-bind a function or procedure without the need to recompile or modify any applications.

7.2.6.9 CREATE PROCEDURE

In Oracle, a procedure is created via this command and is defined using PL/SQL. It is invoked by the application via the via the execute statement.

DB2 stored procedures are an application written in one of the supported languages and stored on the database server but external to the database. The stored procedure is invoked by the client application using the SQL CALL and the SQLDA structure. When the application is compiled, it is bound to the database, and information about the procedure is stored within the database.

As stored procedures in Oracle are written in PL/SQL, they will need to be re-written using one of the DB2 supported languages.

7.2.6.10 CREATE SEQUENCE

This is not used in DB2; however, a before-insert trigger to generate sequential numbers in a column could be used to perform a similar task, for example:

```
CREATE TRIGGER trig1
   no cascade before insert on tabl1
   referencing new as n for each row mode db2sql
   set n.col1 = case
            when (select count(*) from tabl1) = 0 then 1
            else (select max(col1)+1 from tabl1)
            end;
```

This is only an example; if used as written above, it works when rows are inserted one at a time. If multiple rows are inserted in a single SQL statement, the above trigger would insert the same value for each row.

7.2.6.11 CREATE TABLE

The format of the create table statement is similar in both Oracle and DB2. However, as mentioned in the alter table statement, there are some significant differences in some of the clauses used. This is due to the way areas, such as storage allocation and constraints, are managed.

In DB2, space management and concurrency are not defined when the table is created. This means that the Oracle parameters ′PCTUSED′, ′PCTFREE′, ′INITRANS′, and ′MAXTRANS′ are not used in the DB2 form of the create table statement. Also, the ′ENABLE′ and ′DISABLE′ parameters are not used in DB2 as the constraints are managed via the set constraints statement.

It is possible in DB2 to specify that indexes and long columns, such as CLOB, BLOB and DBCLOB, be stored in separate tablespaces.

Other differences in the create table statement clause involve the definition of constraints and the different data types. For further information on these, you should refer to the documentation supplied with DATABASE 2 for AIX Version 2.

7.2.6.12 CREATE TABLESPACE

DATABASE 2 for AIX Version 2 supports two different types of tablespace. These are discussed in detail in Chapter 4, “Storage” on page 35. The create tablespace statement will be quite different because of these differences.

It is possible to specify a regular tablespace, long tablespace or temporary tablespace. The regular tablespace stores all data other than temporary tables. The temporary tables go into the temporary tablespace, of which at least one must exist in the database environment. The long tablespace is for the storage of long or LOB table columns. The long tablespace is optional, and if excluded, this data will be stored in a regular tablespace.

In DB2, you specify the parameter ′USING′ when defining a tablespace. This replaces the Oracle ′DATAFILE′ parameter. Finally, taking a tablespace online or offline is done via the quiesce tablespace command in DB2.

An example of the DB2 create tablespace statement would be:

```
CREATE TABLESPACE payroll MANAGED BY DATABASE
      USING ( DEVICE ′/dev/rhd6′ 1000, DEVICE ′/dev/rhd7′ 1000)
      EXTENTSIZE 4
```

7.2.6.13 CREATE TRIGGER

In Oracle, this command creates a database trigger. A database trigger is a PL/SQL block associated with a table, which is stored in the database. To use this, you must be using Oracle with the procedural option.

DB2 conforms to *Draft SQL3*. The syntax is similar to Oracle, but there are differences that may need to be addressed when migrating to DB2. An example of a DB2 trigger is:

```
CREATE TRIGGER new_hired
   AFTER INSERT ON employee
   FOR EACH ROW MODE DB2SQL
   UPDATE company_stats SET nbemp = nbemp + 1
```

The purpose, function and syntax are similar in Oracle and DB2. The areas that need consideration during a conversion to DB2 are as follows:

- In Oracle, you can specify ′OR REPLACE′ to update or change a trigger, while in DB2, you are required to drop and then create the trigger again.
- In Oracle, you may specify ′OR′ to link together the actions that cause the trigger to fire. In DB2, you need specify triggers one at a time.
- In Oracle, the trigger is specified with a PL/SQL block, which must not contain ddl, rollback or commit statements. In DB2, a trigger is an SQL statement or series of statements. When using the BEFORE trigger in DB2, the triggered SQL statment must be one of the following:
  - a full select
  - a set transition-variable SQL statement

- a signal sqlstate statement

  When the trigger is an AFTER trigger, the triggered SQL statement must be one of the following:

  - an insert sql
  - a searched update sql
  - a searched delete sql
  - a signal sqlstate
  - a full select

- 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.
- The mode of a trigger must be specified in DB2. Currently, DB2SQL is the only available mode.
- Oracle allows the 'FOR EACH STATEMENT' to be specified on BEFORE triggers. This is not allowed in DATABASE 2 for AIX Version 2.

The 'NO CASCADE BEFORE' parameter in DB2 specifies that the actions caused by the trigger will not cause other triggers to be activated. Also, the action of the trigger will be applied before the subject table is actually updated.

DB2 allows you to use the 'REFERENCING' clause to specify correlation names for the rows modified by the trigger. You can specify a correlation name for the row state prior to the triggered operation and the modified row. In addition to this, you can reference temporary tables that contain all the affected rows before the trigger modified them and another table that contains all the modified rows.

In DB2, you can specify the 'WHEN' condition for statement and row triggers, while Oracle only allows this to be used for row triggers. Finally, while Oracle limits the number of triggers allowed, DB2 does not.

7.2.6.14 DROP FUNCTION

The syntax in both Oracle and DB2 are similar. The differences lie in the effect on the database objects after a function has been dropped.

Oracle invalidates any local objects that depend on, or call, the dropped function. If you subsequently reference one of these objects, Oracle tries to recompile the object and returns an error message if the dropped function has not been recreated. In DB2, other objects, excluding packages, that depend on the function must be removed before the function can be dropped. An attempt to drop a function with such dependencies will result in an error (SQLSTATE 42893). However, when the function is dropped, any packages dependent on the function are marked as inoperative. Such a package will not be rebound implicitly. It must either be rebound by using the bind or rebind commands or re-prepared by using the prep command.

7.2.6.15 DROP TABLESPACE

DB2 has the same behavior as Oracle when the Oracle parameters 'INCLUDING CONTENTS' and 'CASCADE CONSTRAINTS' are specified.

In DB2, the tablespace will not be dropped (SQLSTATE 55024) if there is any table that stores at least one of its parts in this tablespace and has one or more parts in another tablespace. To drop this tablespace, the tables in the state mentioned above would need to be dropped first.

7.2.6.16 GRANT

There are two versions of GRANT in Oracle: `grant system privileges` and `grant object privileges`. When using the `grant` statement in DB2, the main consideration, in both cases, is with roles. In DB2, users are at the operating system level, and instead of using roles, DB2 uses the operating system groups. It is possible to grant privileges to groups as well as to users. To convert Oracle roles to DB2, you could create an operating system group for each role, and add the users who were granted that role to the newly created group that matches the old Oracle role.

Since DB2 allows multiple databases to exist in a single instance the system privileges are more complicated than in Oracle. This is because you may require system control over the entire instance or simply over a single database. To resolve this, there are four default system authority levels that provide different levels of control over the instance. These authorities are:

SYSADM System Administration Authority

SYSCTRL System Control Authority

SYSMAINT System Maintenance Authority

DBADM Database Administration Authority

A summary of the capabilities that these authorities allow is listed in Table 20 on page 110. To provide a user with SYSADM, SYSCTRL or SYSMAINT authority, you would need to decide which operating system group will hold the authority, and add the user to that group. Then update the database manager configuration to reflect the group name for the authority level in question. By default, the primary group of the instance owner becomes the SYSADM group. DBADM authority is given to the creator of a database and to anyone else granted the authority.

It is possible to grant the following authorities to users, groups or to the public. These authorities will apply to the entire database your are connected to when the grant is executed. The authorities include:

CONNECT Authority to connect or access the database

BINDADD Authority to create a package and bind it to the database

CREATETAB Authority to create base tables

CREATE_NOT_FENCED Authority to register functions that will execute as a database manager's process

DBADM Database Administration Authority over the entire database

The Oracle option 'WITH ADMIN OPTION' would map to 'GRANT DBADM' in DB2.

The format of the grant statement for granting privileges on objects is similar for both Oracle and DB2. The same considerations discussed for roles and groups, that was discussed previously, is also valid for object privileges.

In Oracle, it is possible to specify a 'WITH GRANT OPTION'. DB2 allows this capability, but it is specified by granting 'CONTROL' privilege to the user or group. The 'CONTROL' privilege in DB2 provides different privileges, depending on the object being referenced. For more information refer to the *SQL Reference Guide*.

7.2.6.17 INSERT

The following syntax of the DML works in both Oracle and DB2:

```
INSERT INTO {table | view} [(column [, column] ...)]
      {VALUES (expr [, expr] ...) | subquery}
```

In Oracle, each 'INSERT INTO .... VALUES (expr)' statement allows for a single row to be inserted into the table. The same statement in DB2 can insert multiple rows into the table. For example, the following statement will insert multiple rows into the table 'tab1':

```
INSERT INTO tab1 VALUES ('wood',CURRENT DATE),('miller',CURRENT DATE),...
```

7.2.6.18 LOCK TABLE

DB2 has the two lockmodes, SHARE and EXCLUSIVE, which are at the table level. Oracle has the additional lockmodes of ROW SHARE, ROW EXCLUSIVE, SHARE UPDATE, and SHARE ROW EXCLUSIVE which are taken at row or table level. If these are used, you will need to consider how to migrate them to table-level locks.

Oracle allows you to specify a view name in the lock table statement. This is not allowed in DB2; the name must be a table or an alias. Also in DB2, it is possible to specify a single table/alias name in each lock statement, while Oracle permits multiple names in a single statement. .

The behavior of the DB2 lock statement will be the same as the behavior in Oracle if the Oracle parameter 'NOWAIT' was specified.

7.2.6.19 REVOKE

The considerations that hold for the grant statement also hold true for the revoke statement. There are also two versions of the revoke statement in Oracle: one for system privileges and the other for object privileges. You should read 7.2.6.16, "GRANT" on page 96 and then refer to the SQL Reference guide for DB2 for the different syntax of the revoke statement or statement.

7.2.6.20 ROLLBACK

The format of the rollback statement is similar in both DB2 and Oracle. The Oracle optional parameters, 'SAVEPOINT' and 'FORCE' are not used in DB2 commands. The Oracle optional parameter, 'FORCE', manually commits an in-doubt distributed transaction. In DB2, you use an interactive command, list indoubt transactions, which prompts you to commit or rollback any in-doubt transactions.

7.2.6.21 SELECT

The format of the select statement in both Oracle and DB2 is almost the same. There are, however, differences in some of the options used within the select statement. These include:

START WITH, CONNECTED BY
: This enables you to retrieve rows in a hierarchical order in Oracle, but is not available in DB2. However, the same effect may be achieved using recursive SQL statements.

GROUP BY
: In Oracle, you are allowed to specify columns that are not in the select list. In DB2, a column used in the GROUP BY clause must also appear in the select list for the query. If it is not, then DB2 will return the error, SQL0119N SQLSTATE=42803. The query may however be re-written using a nested table expression.

MINUS
: In DB2, this is done via the EXCEPT keyword. It is also possible to specify EXCEPT ALL and INTERSECT ALL in DB2.

ORDER BY
: In Oracle, it is possible to specify any column in the order by clause, provided that the DISTINCT keyword is not used. DB2 requires that the columns used in the order by clause to always appear in the select list. If they do not, then the error SQLSTATE=42707 will be returned.

FOR UPDATE OF
: In DB2, the column must be unqualified and identify columns of the table or views identified in the first FROM clause of the full select.

NOWAIT
: DB2 will have the same behavior as Oracle if this parameter is not specified; however, DB2 will only wait until the time-out value.

The comments on the full select also apply to subqueries and correlated subqueries, which are similar in Oracle and DB2. For the embedded select into statement, Oracle has the optional parameters 'INDICATOR' and 'INDICATOR-VARIABLE'. These are not used in DB2.

The Oracle expression DECODE is compatible with the CASE expression that is provided by DB2. However, the format of the CASE expression is different, and the SQL statement would need to be reworked. The predicates and comparison operators, such as BETWEEN, LIKE, IN, and =, are compatible between Oracle and DB2. For further information on these operators and expressions, refer to 7.2.1, "Operators, Expressions and Conditions" on page 83.

There are some capabilities provided by DB2 that are not available in Oracle. These include:

- FOR READ ONLY and FOR FETCH ONLY

  This clause indicates that the resulting table is read-only; therefore the cursor cannot be referred to by positioned update and delete statements. For result tables in which updates and deletes are allowed, specifying FOR READ ONLY (or FOR FETCH ONLY) can possibly improve the performance of FETCH operations by allowing the database manager to do blocking and avoid exclusive locks. Some result tables are read-only by nature. An example would be a table based on a read only view. The FOR READ ONLY clause can still be specified on such tables.

- OPTIMIZE FOR

  The OPTIMIZE FOR clause indicates to the optimizer how many rows may be returned from this command. The optimizer can then use this as a factor when deciding on the optimum path for the command.

- Common table expression

  A common table expression is like a temporary view within a complex query that can be referenced in other places within the query. For example, we could avoid creating a view and use a common table expression:

  ```
  SELECT....WITH com-tab-name (col1,col2,..) AS fullselect.....
  ```

  Common table expressions can be used in `select`, `create view` and `insert` statements. It can be used in the following circumstances:

  - In place of a view to avoid creating the view
  - To enable grouping by a column that is derived from an expression
  - When the desired result table is based on host variables
  - When the same result table needs to be shared in a full `select`
  - When the result needs to be derived using recursion

- Nested table expression

  This allows the creation of intermediate results without creating a temporary table. The same result could be achieved by using a view; however, no host variables or grouping by a derived column would be allowed. It can be used:

  - In place of a view to avoid creating the view
  - To enable grouping by a column that is derived from an expression
  - When the desired result table is based on host variables

7.2.6.22 SET TRANSACTION

DB2 does use the `set transaction` statement. Through the command line processor wou may use the `change isolation to` command to set the required isolation level. The supported isolation levels are Repeatable Read (RR), Read Stability (RS), Cursor Stability (CS) and Uncommitted Read (UR). You may also specify 'FOR UPDATE OF', 'FOR READ ONLY' or 'FOR FETCH ONLY' to indicate the command will be doing reads or updates.

7.2.6.23 TYPE

Oracle allows you to use an embedded command, `type`. The syntax of this command is as follows:

```
EXEC SQL TYPE type IS data-type
```

This is used in Oracle applications to perform user-defined type equivalencing. It can only be used with PRO*C or PRO*PASCAL Precompilers.

In DB2, there are User-Defined Types (UDTs) that provide more functionality than the Oracle `type` statement. The DB2 UDT is stored in the database and is accessible to users and applications that are bound to the database containing the UDF definition.

An example of the DB2 UDT for the type `inch` would appear as follows:

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

You can then use inch as a unique type. For example:

```
CREATE TABLE US_BOX (udesc  char(20), ulength inch not null)

SELECT * from US_BOX where ulength > inch(10)
```

Note that the integer value 10 in the above example must be type cast to the type inch. This is because *inch* and *integer* are defined as two different types. The casting functions and comparison operators are automatically created when the 'WITH COMPARISONS' parameter is included in the create distinct type statement.

The DB2 CAST function may also be of use when converting type statements. Refer to 7.2.6.24, "VAR" for further information on the CAST function.

7.2.6.24 VAR

This is not available in DB2. Once a host variable has been defined in DB2, it is not possible to redefine it at a later time. However, in some cases, you may be able to use the CAST expression. CAST may be used anywhere an expression is allowed. The syntax of the CAST function is:

```
CAST ( {{ expression | NULL | parameter-marker } AS datatype } )
```

It is possible to use the CAST function in a select statement as follows:

```
SELECT empno, CAST(salary AS integer) from emp
       WHERE age = CAST(? AS T_AGE)
```

7.3 Constraints

Constraints are found in both Oracle and DB2. In both environments, there exist table constraints and column constraints. Both of these types of constraints are similar in the different environments.

In Oracle, a unique constraint may be placed on one or more columns of a table to ensure uniqueness. DB2 performs this function by creating a unique index on the columns. Oracle also allows you to specify a column that allows NULL values as the 'PRIMARY KEY'. This is not allowed in DB2. Any column used as a primary key must be defined as 'NOT NULL'.

Oracle allows you to specify the 'EXCEPTIONS INTO' parameter. This allows you to identify a table into which Oracle places information about rows that violate an integrity constraint. DB2 provides this function through the statement, set constraints for table2 immediate checked for exception in table2. Oracle allows you to disable integrity constraints though the DISABLE statement. To disable integrity constraints in DB2, the statement, set constraints for table off should be used.

The Oracle option 'ON DELETE CASCASE' is available in DB2; however, there are more alternatives in DB2 that provide greater functionality. These include:

- ON DELETE RESTRICT

  Enforced before other constraints and prevents the row from being deleted if it has dependent rows.

- ON DELETE NO ACTION

Enforced after other constraints and prevents the row from being deleted if it has dependent rows.

- ON DELETE CASCASE

  Deletes any rows that are dependent on the row being deleted.

- ON DELETE SET NULL

  Dependent rows have the foreign key set to NULL.

- ON UPDATE RESTRICT

  Enforced before other constraints and prevents the row from being updated if it violates dependent rows.

- ON UPDATE NO ACTION

  Enforced after other constraints and prevents the row from being updated if it violates dependent rows.

An additional difference for the column constraints is that, in Oracle, you can specify a column as 'NULL' or 'NOT NULL'. In DB2, you only need to specify a column as 'NOT NULL' because the default will be to allow NULL values. If you specify 'NULL' in DB2, you will receive the error, SQLSTATE=42601.

7.4 Joins

Joins in Oracle and DB2 are very similar in the syntax used and in the way they work. DB2 uses the optimizer to determine the best join method, based on the catalog statistics for the tables involved in the join. To allow you to influence the optimizer, some of the statistical system tables may be modified by a user. This also allows you to simulate what would happen in different environments. Oracle lets you use the optimizer to decide, or you can specify join instructions by using HINTS in the application.

The two main types of joins are inner joins and outer joins. Inner joins combine the data, but if a row exists in one table and not the other, the information is not included in the result table. Outer joins preserve these unmatched rows.

All joins in Oracle and DB2 are the same, except for the specification of the outer join. The function of the outer join is the same in Oracle and DB2; however, the specification of the join is different.

In Oracle, a left outer join is characterized by a join condition that uses the outer join operator (+). All the rows that meet the join condition are returned. Also returned are all the rows from the table without the outer join operator for which there are no matching rows in the table with the outer join operator. Oracle joins the row to a row with all the columns from the other tables containing NULL values.

In DB2, there are three types of outer join. These are:

- Left outer join, which includes rows from the left table that were not matched to rows in the right table.
- Right outer join, which includes rows from the right table that were not matched to rows in the left table.
- Full outer join, which includes rows from both the left and right tables that did not have matches in the other table.

control of which of the outer join methods is to be used is done by controlling which of the common table expressions are included in the query. An example of performing an outer join can be found in the *DATABASE 2 SQL Reference for common servers Version 2* manual.

7.5 CURSOR and DYNAMIC SQL

This section will compare cursors and Dynamic SQL use in Oracle 7 and DATABASE 2 for AIX Version 2. The topics covered include:

- General considerations in migrating to DB2
- The Dynamic SQL methods available
- The statements involved in using cursors and dynamic SQL

You will find that most of the cursor statements in Oracle and DB2 will be very similar. In DB2, you can specify a cursor 'FOR READ', 'FOR FETCH' or 'FOR UPDATE'. If not specified, an ambiguous cursor results, and DB2 will decide what type it is via the context of the cursor statement. For example, a cursor defined as `select ..... order by` would be determined to be a read-only cursor.

Oracle only supports embedded cursors. DB2 also supports embedded cursors; however, the `DECLARE, OPEN, FETCH` and `close` statements may also be issued interactively. This could be done through the command line processor.

The Oracle optional parameter 'AT dbname' is not used in DB2. This is because of the different concepts in databases and instances. The first SQL statement issued would be a `connect to database`; otherwise an implicit connect to the default database would occur.

Oracle allows you to specify a PL/SQL block in some SQL statements. Since PL/SQL is not found outside Oracle, these statements would need to be converted into part of the application, into a DB2 compound SQL or into a stored procedure.

The Oracle term, "placeholders" are known as parameter markers in DB2. The function that they provide is identical.

There are four different methods for using dynamic SQL. These are:

1. Nonquery without host variables. This refers to the `execute immediate` statement, as shown in the following example.

```
SET I_STMNT ='INSERT INTO WORKTAB
     SELECT * FROM EMP WHERE ACTNO < 100'

EXECUTE IMMEDIATE :I_STMNT
```

2. Nonquery, where the number of host variables, parameter markers and their data types are known at precompile time. This refers to the prepare and execute statements, as shown in the following example.

```
I_STMNT='INSERT INTO TDEPT VALUES (?,?,?,?)'

PEPARE DEPT_INSERT FROM I_STMNT

(Ask for the host variable values from the user)

EXECUTE DEPT_INSERT USING :HV1, :HV2, :HV3, :HV4
```

3. Query, where the number of host variables, parameter markers, select list items, and their data types are known at precompile time. This refers to the prepare, declare, open, fetch and close cursor statements. For example:

```
'SELECT ENAME,EJOB,ELOCATION FROM EMP WHERE EMPNO = ?'
```

   This method is the same as method two, but allows select statements.

7.5.1 CLOSE

The format of the close statement shown below will work in both the Oracle and DB2 environments.

```
CLOSE cursor
```

There are no differences in behavior between Oracle and DB2 for this statement.

7.5.2 DECLARE CURSOR

When using a command name or block name in Oracle, you must first define it using the declare statement statement or the prepare statement.

DB2 also provides the 'WITH HOLD' option. This maintains resources across multiple units of work. For example, with a unit of work that ends with a commit. Any locks the cursor specified 'WITH HOLD' are kept, and the cursor remains open and positioned before the next logical row. It is then possible to issue a fetch for the next row.

7.5.3 DESCRIBE

The format of the describe statement shown below will work in both the Oracle and the DB2 environments.

```
DESCRIBE statement-name INTO descriptor-name
```

The DB2 statement works like the Oracle statement where the default option 'SELECT LIST FOR' is specified. Describe input, which is specified in Oracle by using bind variable for, is not available in DATABASE 2 for AIX Version 2.

7.5.4 EXECUTE

The following format of the execute statement will work in both Oracle and DB2 environments.

```
EXECUTE statement_name
       [USING { :host_variable [[INDICATOR] :indicator_variable]
             [, :host_variable [[INDICATOR] :indicator_variable] ...
              |  DESCRIPTOR descriptor } ]
```

In Oracle, the SQL statement may only be delete, insert or update statements.. DB2 does not allow the the prepared command to be use a select statement.

It is possible in Oracle to specify a PL/SQL block with known host variables; however, the PL/SQL may not contain a fetch statement. Because PL/SQL is not found outside Oracle, any PL/SQL code would need to be rewritten as a part of the application, converted to compound SQL or a stored procedure. Another alternative would be to rewrite the execute statement using some of the other SQL statements available in DATABASE 2 for AIX Version 2. DATABASE 2 for AIX Version 2

7.5.5 EXECUTE IMMEDIATE

The syntax, execute immediate host-variable, will work in both the Oracle and DB2 environments.

Oracle will only support the delete, insert or update statements in the execute immediate statement. The DB2 execute immediately statement will support the following:

ALTER TABLE	ALTER TABLESPACE
COMMENT ON	COMMIT
CREATE ALIAS	CREATE DISTINCT TYPE
CREATE EVENT MONITOR	CREATE FUNCTION
CREATE INDEX	CREATE TABLE
CREATE TABLESPACE	CREATE TRIGGER
CREATE VIEW	DELETE
DROP	GRANT
INSERT	LOCK TABLE
REVOKE	ROLLBACK
SET CONSTRAINTS	SET CURRENT EXPLAIN SNAPSHOT
SET CURRENT FUNCTION PATH	SET CURRENT QUERY OPTIMIZATION
SET EVENT MONITOR STATE	UPDATE

In Oracle, you may use a host-variable or a text literal containing the SQL statement, while in DB2 you must use a host variable. Oracle also allows you to specify a PL/SQL block as long as it contains no host variables and does not contain a fetch. To move this to DB2, it would need to be converted into part of the application, into compound SQL or into a stored procedure.

7.5.6 PREPARE

The Oracle format of the prepare statement will work in DB2 without any changes. However, DB2 allows you to specify an SQLDA record into which prepare statement information will be placed and so provides some extra parameters not found in Oracle.

In Oracle, any variables in the host string are placeholders. The actual host variable names are assigned in the open using, execute using or fetch into statements. In DB2, these placeholders are known as *parameter markers* and are specified by using a question mark (?). They are assigned in the open using and execute using statements.

Oracle allows you to specify a text string or a host string for the command being prepared; however, in DB2 a host string must be specified.

If the command name identifies an existing prepared statement, Oracle considers them to be the same and will use the most recently defined one. In DB2, the previous prepared command is destroyed and the new one used; so the actual effect is the same.

7.6 Reserved Words

The following schema names or user names are reserved in DB2:

- SYSCAT
- SYSIBM
- SYSSTAT

In addition, it is strongly recommended that schema names never begin with the SYS prefix. SYS is, by convention, used to indicate an area reserved by the system.

DATABASE 2 for AIX Version 2 does not enforce reserved words. This allows SQL to be converted to DB2 with checking not being required. Keywords can be used as ordinary identifiers, except in the context where they could also be interpreted as SQL keywords. In such cases, the word must be specified as a delimited identifier.

IBM SQL and ISO/ANSI SQL92 reserved words are documented in Appendix B, "IBM SQL Reserved Words" on page 159.

7.7 Special Registers

A special register is a storage area that is defined for an application processed by the database manager. It is used to store information that can be referenced by SQL statements. Special registers are in the database code page. These registers include:

- CURRENT DATE, CURRENT TIME and CURRENT TIMESTAMP

  These special registers specify information that is based on a reading of the time-of-day clock. The clock is read when the SQL statement is executed at the application server. If this special register is used more than once within a single SQL statement, or used with another special register within a single statement, all values are based on a single clock reading. For example:

  ```
  UPDATE project SET startdate = CURRENT DATE where projno='MA2111'.

  SELECT class FROM tab1 WHERE start > CURRENT TIME and day=3

  INSERT INTO tab1 VALUES (CURRENT TIMESTAMP, :src, :sub, 'Pete', 12.6)
  ```

- CURRENT TIMEZONE

  The CURRENT TIMEZONE special register specifies the difference between Coordinated Universal Time (UTC) or Greenwich Mean Time (GMT) and local time at the application server. The difference is represented by a DECIMAL(6,0) data type where the first two digits are the number of hours; the next two digits are minutes, and the last two digits are seconds.

- CURRENT SERVER

  The CURRENT SERVER special register specifies a VARCHAR(18) value that identifies the current application server.

- CURRENT QUERY OPTIMIZATION

  The CURRENT QUERY OPTIMIZATION special register specifies an integer value that controls the class of query optimization performed by the database manager when binding statements.

- CURRENT EXPLAIN SNAPSHOT

  The CURRENT EXPLAIN SNAPSHOT special register specifies a CHAR(8) value. It contains a value controlling the behavior of the Explain Snapshot facility for dynamically prepared statements.

- CURRENT FUNCTION PATH

The CURRENT FUNCTION PATH special register specifies a VARCHAR(254) value that identifies the function path to be used to resolve function references and data type references that are used in dynamically prepared SQL statements.

Chapter 8. Database Security

This chapter discusses the security features found in DATABASE 2 for AIX Version 2 and how best to convert from the security features of Oracle 7.

The topics covered include:

- Instance and Database Security
- Users, Groups and Privileges
- DCE Directory and Security
- Applications

8.1 Instance and Database-Level Security

As discussed in Chapter 6, “Database Schema” on page 61, DB2 allows for multiple databases to exist within a single instance. Oracle, however maintains a one-to-one mapping of a database to an instance. This basic difference in the structure of the database instance leads to differences in the way security is managed. In both environments, we need to control who has access to the database environment or instance and the level of access within the database. DB2 also needs to consider who has access to the different databases within the instance.

8.1.1 Oracle Instance and Database Security

Oracle provides two choices of authorization methods before a user may access the database.

The first choice is a single authentication level. This level leaves user authentication to the operating system. The user will usually supply a user ID and password t o the operating system. Once the operating system has verified the user, access to the database is granted.

The second choice is double authentication. In this situation the user must supply a user ID and password to Oracle after they have logged in at the operating-system level. Only after the Oracle authentication is completed is the user granted access to the database.

The single authentication method is similar to DATABASE 2 for AIX Version 2. However, as a DB2 instance may contain multiple databases, there are additional DB2 authority requirements before the user may access any individual database.

8.1.2 DB2 Instance and Database Security

DB2 does not maintain a list of users and passwords for accessing the instance or database as is done in Oracle. DB2 authentication of users is done by the operating system.

If a user has a valid operating system user ID and password, they should be able to access the database instance. When an instance is created, the ownership of that instance is associated with a user and their primary group. This user is known as the instance owner and inherits system administration authority for the database instance.

By setting the file or directory permission at the operating-system level, it would be possible to stop other users from accessing the instance. This can be done, for example, by removing read and execute permission from the directories. Under normal circumstances, this is not a problem since even though a user may have access to the instance, access to the individual databases must be granted to that user or to a group the user belongs to. The users and groups are defined at the operating system level. Granting access to a database may be done by adding a user to the group that has been granted access.

Since DATABASE 2 for AIX Version 2 allows you to catalog a remote database within your instance, the security of connecting to another machine needs to be considered. There are three authentication methods for connecting to remote databases. They are:

SERVER Server authentication requires that the node containing the actual database authenticates all connections. This requires the connecting user to have a valid user ID and password on the server machine. This user ID and password is at the operating-system level. The user ID on the server machine does not need to be defined as a logon user, but must exist on the machine with a valid password.

CLIENT Client authentication means that the server will assume authentication has been done at the client. This should only be used in an environment with a secure network.

DCS DCS authentication is similar to SERVER authentication, but it is used when connecting to a DRDA application server.

The type of authentication of a database is defined at the instance level on the server node. When cataloging a database on a client, you are required to specify the authentication level as well. The authentication level on both the client and server must match.

8.2 Users, Groups and Roles

Once authenticated, a user will then be able to perform certain operations on the database or instance. The authorization or privilege of the user will initially depend on the user′s ID and group. These users or groups may then be granted privileges. DB2 does not use the concept of a “role” as found in Oracle. Oracle roles are covered later in this section.

The definition of users, groups and roles is quite different in DB2 and Oracle. However, once these have been established, the granting and revoking of privileges on them is similar for both environments.

Oracle 7 Users and Groups

Oracle may maintain a list of users and passwords within the Oracle instance which are separate to the operating system. If double authentication is being used, the user must know a valid Oracle user ID and password before they will be granted access to the database. The level of access the user is granted depends on the user ID and the roles granted to that user.

Oracle creates a default set of users and privileges that cannot be changed. These users and privileges provide different levels of access to the database. The default users that are created are:

SYS SYS is the owner of all the tables and views that make up the data dictionary.

SYSTEM SYSTEM is the owner of all the additional tables and views that contain administrative information and the internal tables and views used by Oracle tools.

The default privileges include:

INTERNAL This privilege level is usually only granted to the database administrator because it enables operations such as starting and stopping the database.

OSOPER A user with OSOPER privilege is able to perform operations such as:

- startup and shutdown
- alter database open/mount
- alter database backup
- archive log
- recover
- use a restricted session

OSDBA A user with OSDBA privilege is able to perform all the operations of a user with OSOPER privilege, plus all the system privileges with ′ADMIN OPTION′ and the ability to create a database.

DBA The DBA privilege allows the user to perform all the database system operations. By default the users SYS and SYSTEM are automatically granted this level of privilege.

Oracle has a default group called PUBLIC. If a privilege is granted to PUBLIC, anyone who is authenticated by the database will have the privilege.

Oracle also allows you to group privileges into roles. These groups of privileges may then be granted to a user. By doing this, you are able to minimize the amount of overhead in controlling what privileges the different database users have.

DATABASE 2 for AIX Version 2 Users and Groups

DB2 also creates a default set of users and groups. When a DATABASE 2 for AIX Version 2 instance is created, it is associated to an operating system user ID. The user is then known as the instance owner. The instance owner will inherit system administration authority over the entire instance. DB2 provides four levels of administrative authority. These levels provide different access and privileges on the database instance and to the databases it contains.

The four authority levels are:

SYSADM System Administration Authority

SYSCTRL System Control Authority

SYSMAINT System Maintenance Authority

DBADM Database Administration Authority

Table 20 on page 110 provides a full listing of the authorities and capabilities provided with each of these authority levels.

To assign the authority levels for SYSADM, SYSCTRL and SYSMAINT, they need to be associated to an operating system group. SYSADM is automatically assigned to the primary group of the instance owner. The other two authority levels are not assigned a group by default. To assign an operating system group to the authority levels, the database manager configuration needs to be modified using the following commands:

```
UPDATE DATABASE MANAGER CONFIGURATION USING SYSCTRL_GROUP  group-name
UPDATE DATABASE MANAGER CONFIGURATION USING SYSMAINT_GROUP group-name
UPDATE DATABASE MANAGER CONFIGURATION USING SYSADM_GROUP   group-name
```

Once the authorities have been associated to an operating-system group, users may be granted the authority by adding them to the group at the operating system level.

Table 20. DB2 System Group Authorizations

Function	SYSADM	SYSCTRL	SYSMAINT	DBADM
CATALOG / UNCATALOG DATABASE	yes	yes		
CATALOG / UNCATALOG NODE	yes	yes		
CATALOG / UNCATALOG DCS	yes	yes		
MIGRATE DATABASE	yes			
UPDATE DBM CFG	yes			
GRANT / REVOKE DBADM	yes			
GRANT / REVOKE SYSCTRL	yes			
REVOKE / GRANT SYSMAINT	yes			
FORCE USERS	yes	yes		
CREATE / DROP DATABASE	yes	yes		
QUIESCE INSTANCE or DATABASE	yes	yes		
CREATE / DROP / ALTER TABLESPACE	yes	yes		
RESTORE TO NEW DATABASE	yes	yes		
UPDATE DB CFG	yes	yes	yes	
BACKUP DATABASE or TABLESPACE	yes	yes	yes	
RESTORE TO EXISTING DATABASE	yes	yes	yes	
PERFORM ROLL FORWARD RECOVERY	yes	yes	yes	
START / STOP DATABASE INSTANCE	yes	yes	yes	
RESTORE TABLESPACE	yes	yes	yes	
RUN TRACE	yes	yes	yes	
TAKE DBM or DB SNAPSHOTS	yes	yes	yes	
QUERY TABLESPACE STATE	yes	yes	yes	yes
UPDATE LOG HISTORY FILES	yes	yes	yes	yes
QUIESCE TABLESPACE	yes	yes	yes	yes
LOAD TABLES	yes	yes		yes
SET / UNSET CHECK PENDING STATUS	yes	yes		yes
READ LOG FILES	yes	yes		yes
CREATE / ACTIVATE / DROP EVENT MONITORS	yes	yes		yes
RUN LOAD UTILITY	yes	yes		yes

The authority of DBADM is automatically assigned to the creator of individual databases. The privilege may then be granted to an individual user or to a group, excluding the special group PUBLIC. Unlike the SYSADM, SYSMAINT and SYSCTRL privileges, DBADM is only applicable to a single database within the instance.

8.2.1 Privileges

In both DB2 and Oracle, user privileges are granted by using the grant statement and revoked with the revoke statement. The concept of granting and revoking privileges is similar in both the DB2 and Oracle environments.

Oracle 7 automatically creates a special group called PUBLIC. Users are automatically added to this group. As a member of the PUBLIC group, a user is able to select from certain tables in the data dictionary, grant or revoke privileges on the PUBLIC group, or create links and synonyms to be assigned to the PUBLIC group.

The PUBLIC group is the only group used in Oracle, and it does not relate to any of the operating-system groups. Rather than making use of the operating-system groups like DB2, Oracle allows you to create roles. A role is a group of privileges that may then be granted to users or other roles. As mentioned above, DB2 makes use of the operating-system groups to perform a function similar to the Oracle roles. Privileges may be granted to a group, which is similar in concept to grouping them into a role. Then,to grant users the privileges, you simply add the user to the group at the operating-system level.

Oracle divides the granting of privileges into two forms.

1. Granting system privileges and roles to users or roles
2. Granting privileges on objects to users or roles

DATABASE 2 for AIX Version 2 also divides privileges into separate groups. These are

1. Granting database authorities to users or groups
2. Granting index privileges to users or groups
3. Granting package privileges to users or groups
4. Granting table or view privileges to users or groups

Privileges are grouped differently in DB2, but remain similar in syntax and function to the privileges in Oracle. The different groupings in DB2 are based on the different types of privileges that may be granted to the different database objects. Oracle contains a large selection of system privileges that may be granted to individual users. For example, you may grant one user the ability to drop any table in the database, while another user may be able to drop users. This level of granularity is not found in DB2 at the system or database level. Many of the system-level privileges are associated with the four authority levels discussed previously. The database-level authorities available in DATABASE 2 for AIX Version 2 are:

CONNECT	Allows a user to connect or access the database.
BINDADD	Allows a user to bind a package (stored procedure or application) to the database.
CREATE_NOT_FENCED	Allows a user to register functions that will be executed in the database manager process area. Normally functions are fenced from the database to minimize the risk of adverse side effects.
CREATETAB	Allows a user to create their own tables in the database.

DBADM Grants the user database administration authority. They inherit all the above privileges and have all privileges on all objects within the database. They may also grant or revoke privileges from other users.

The above authorities may be granted to users, group or to the special group, PUBLIC. With the single exception that DBADM authority may not be granted to PUBLIC. The DB2 PUBLIC group is the same as the PUBLIC group used in Oracle.

At the table and view level, the privileges in Oracle and DB2 are almost identical. The privileges found in DB2 for tables and views are:

ALL Grants the user all of the privileges listed below.

ALTER Allows the user to alter the specified table by adding columns, and by creating or dropping primary keys, foreign keys, triggers, or constraints. This privilege is not valid for views.

CONTROL The control privilege grants the user all the privileges listed, including the capability to drop the table or view, execute the runstats utility on the table or view or grant any of the privileges (except CONTROL) to another user. Granting a user the CONTROL privilege is similar to using the Oracle parameter, 'WITH GRANT OPTION'.

DELETE Allows the user to delete rows from the table or an updatable view.

INDEX Permits the user to create an index on the table. The creator of an index automatically has control privilege on that index. This privilege is not valid for views.

INSERT Allows the user to insert rows into the table or updatable view. This also allows the user to run the import utility.

REFERENCES Allows the user to create or drop a foreign key that references the table as a parent. This privilege is not valid for views.

SELECT Allows the user to retrieve rows from the table and run the export utility.

UPDATE Allows the user to update existing rows in the table or updatable view.

DB2 also has a privileges associated with packages (stored procedures or applications). These packages are not the same as an Oracle package. For further information on DATABASE 2 for AIX Version 2 packages, refer to Chapter 6, "Database Schema" on page 61 or to Chapter 9, "Applications" on page 117. The privileges associated with packages are:

CONTROL Allows the user to grant the package privileges (except CONTROL) to another user. Both BIND and EXECUTE privilege are granted to a user that is granted the CONTROL privilege.

BIND Allows the user to rebind a package to the database. The package must already have been bound by someone with BINDADD authority.

EXECUTE Execute privilege allows the user to run the package that has been bound.

Finally, there are the privileges associated to an index. This should not be confused with the index privilege on a table. If you have been granted the index privilege for a table, you are able to create indexes and drop them for that table. If you create an index on a table and you wish to allow other users to drop them, you will need to grant them the following index privilege:

CONTROL Allows a user to drop the index for which the control privilege has been granted.

8.3 DCE Directory and Security

DCE services are not found in Oracle, but may be incorporated into the DATABASE 2 for AIX Version 2 environment. Using DCE Directory Services, you are able to store information about the target databases within the DCE directory objects. The DCE information on the clients may be stored in the database manager configuration or as environment variables.

Currently, DATABASE 2 for AIX Version 2 supports DCE Cell Directory Services (CDS) for a single cell environment. There also exists DCE Global Directory Services (GDS) for multiple cells or global cell naming environments.

The *Database 2 Administration Guide, Version 2* provides information on the tasks required to configure DCE CDS for DB2. The type of information stored in the DCE database object contains information such as the database name, location, protocol, authentication and well as information about the database products.

Authentication of users may be performed at the client, server or DDCS Client/Server gateway. This is the same as in DB2 without DCE. However, it is important to correctly specify the type of authentication being used in the database object at the DCE server because this is where a client first requests that information.

8.4 Applications

Application development in Oracle is different to application development in DB2. One of the main differences is the way in which applications are compiled and bound to a database. If you are not familiar with application development in the DATABASE 2 for AIX Version 2 environment, you should refer to Chapter 9, “Applications” on page 117.

Most of the applications written for Oracle will need to be modified and re-compiled for DATABASE 2 for AIX Version 2. This process will require application development on the new DB2 database. In DATABASE 2 for AIX Version 2, there are several methods that you could use to allow application development without placing the database at risk.

The first method would be to create a second database instance. This second instance would be a complete copy of the primary instance. It would be possible to grant system administration authority to the application developer on the development instance. By doing this, you provide the application developer full

access to the database environment without the risk of corruption to the live or primary instance.

Another method would be to create a second database in the existing instance. By granting the application developer database administration authority on this development database, you also provide full access to the development database, but not to the primary database. However, this would require the application developer to ask the system or instance owner to make any required changes to the database instance.

If you have space or resource limitations, then you may decide to let the application developer access the primary database. To minimize the risk of corruption in this environment, you would grant the developer only a minimum set of privileges. The main privileges required by a developer include:

- CONNECT authority on the database(s) the applications access.
- BINDADD authority if the developer is to create new applications.
- BIND authority if modifications to applications that have already been bound to the database are to be made.
- CREATETAB authority if new tables are to be made within the database.
- The developer may also require specific privileges on existing database objects, such as tables or indexes.

Static SQL and Dynamic SQL

An application that contains static SQL must be prepared or bound to the database before it may be executed. When a static SQL application is prepared the SQL statements are turned into executable form and stored within the database as packages.

Once an application has been bound to the database, the end user will only require execute privilege and does not require privileges on the tables the package/application accesses. This is because the privileges of the user who binds the application will be inherited by the user running the application.

The person binding the application must have sufficient access to all the database objects accessed by the application.

This is not the case with dynamic SQL. In the case of applications that contain dynamic SQL, the authorizations are determined at the time the statement is executed and not at the time the application is bound to the database. The individual users running the application need to have the appropriate privileges on the database objects that the application accesses.

By choosing static SQL over dynamic SQL, you are able to control what a user can modify through the application program because outside the application, the user will not have access to the database objects. Also, since dynamic SQL statements must be prepared at run-time, they will be slower in execution than the same statement would be using static SQL. In short, dynamic SQL should only be used when the added flexibility of dynamic statements is required.

Application Programming Interface

Most of the Application Programming Interface (API) calls for DATABASE 2 for AIX Version 2 do not require any special privilege to be called. However,

depending on the function the API performs, you may require some levels or authorization or privilege. The *Application Programming Interface Reference* lists all the DB2 API's and the privileges required to execute them.

Chapter 9. Applications

This chapter covers the different topics that require consideration when converting an application from an Oracle 7 environment to a DATABASE 2 for AIX Version 2 environment. These topics include:

Programming Languages

Pre-Compilers

Stored Procedures

9.1 Pre-Compilers

Embedded SQL is available in both Oracle 7 and DATABASE 2 for AIX Version 2. The differences in the actual SQL syntax has been covered in Chapter 7, “SQL Language Elements” on page 71. This chapter focuses on the compilation and execution of applications in the DB2 environment.

9.1.1 Embedded SQL in Oracle 7

The process involved in compiling an Oracle application is similar to that of compiling any application under the operating system. Figure 22 on page 118 indicates the basic process involved in the compilation of an Oracle embedded SQL program. These basic steps are:

1. **Precompilation**

   During the precompilation phase or step, the embedded SQL statements are replaced with data structures and library calls for the host language.

2. **Compilation**

   The compilation phase turns the precompiled files, and any other host language source files used by the application, into object modules.

3. **Linking**

   Linking is the final phase for an Oracle compilation. This will link the object modules and libraries to produce an executable application.

Once the application has been compiled, it will be ready for users to execute.

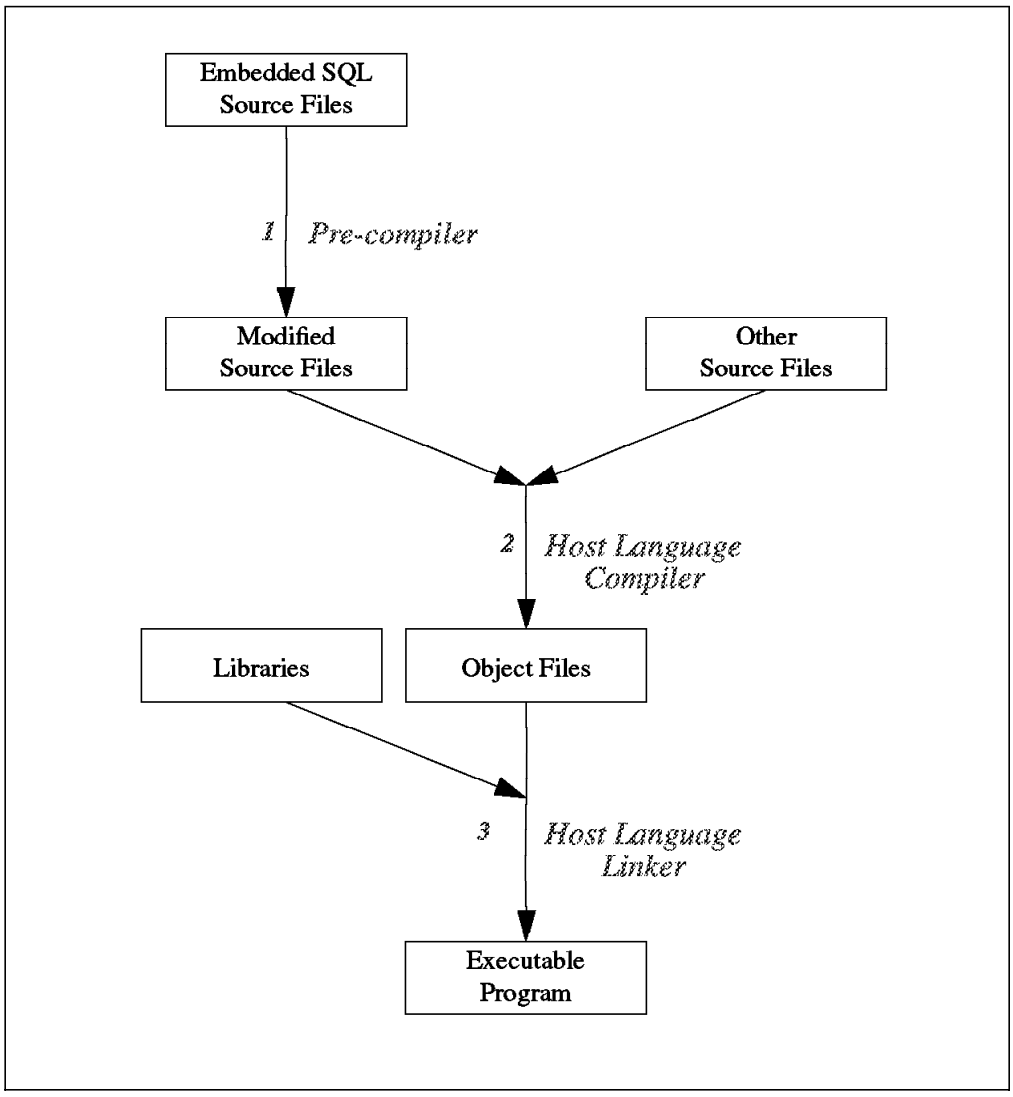


Figure 22. Compiling Oracle Embedded SQL Applications

9.1.2 Embedded SQL in DB2

The compilation process of an embedded SQL application in DATABASE 2 for AIX Version 2 is similar to the compilation process for Oracle. However, an additional step is required before the application may be executed successfully against the database. Figure 23 on page 119 shows the compilation process for DB2, including the additional step.

The basic steps for compiling an application with embedded SQL in a DB2 environment are:

1. **Precompilation**

   During the precompilation phase or step, the embedded SQL statements are replaced with data structures and library calls for the host language. This phase will also produce a bind file that will be used in the last phase.

2. **Compilation**

   The compilation phase turns the precompiled files, and any other host language source files used by the application, into object modules.

3. **Linking**

   Linking will link all the object modules and libraries into an executable application.

4. **Binding**

   Binding will use the file produced by phase one and store the information in the database as a package. This package includes all the necessary information required to execute the static SQL statements found in the application.

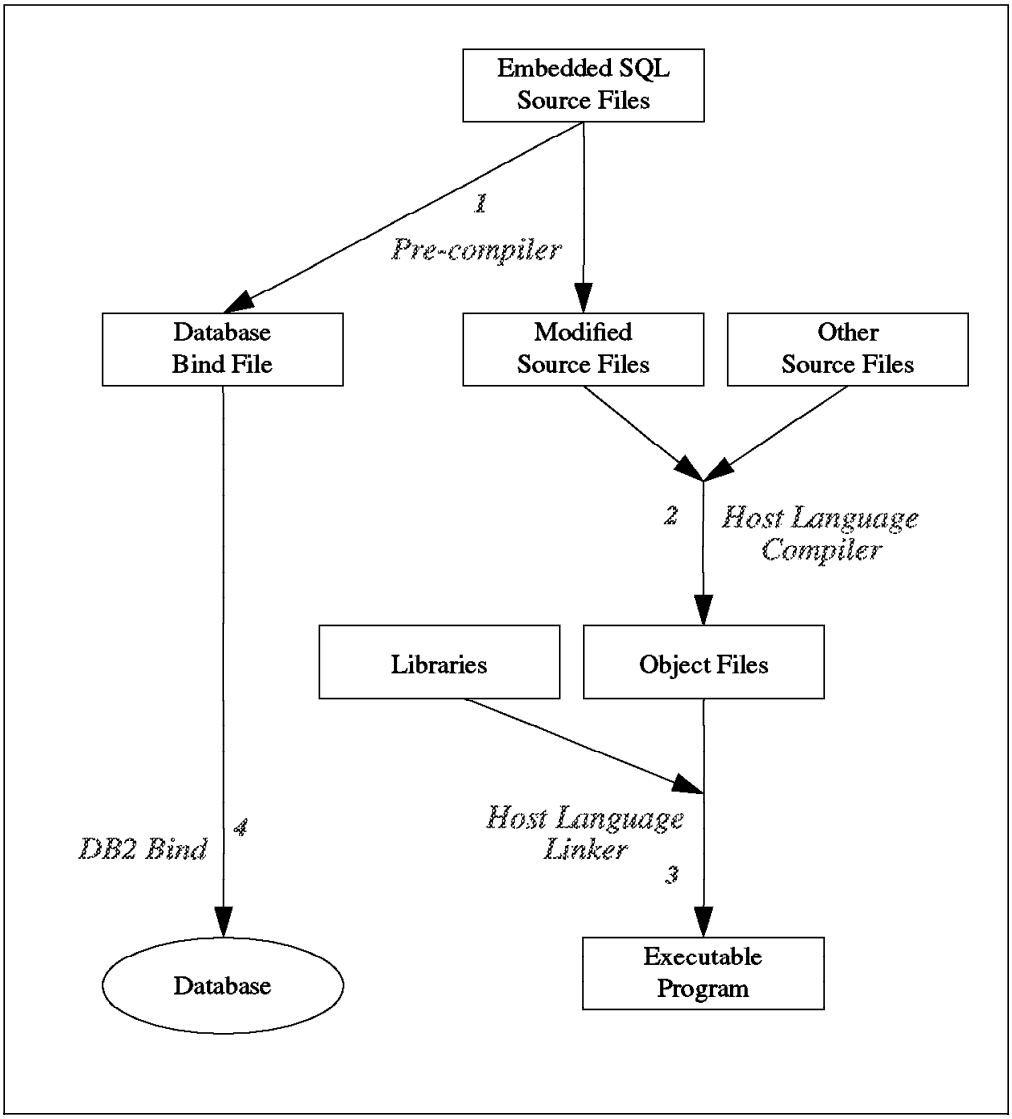


Figure 23. Compiling DB2 Embedded SQL Applications

Since DB2 binds applications to a database, you will need to be connected to the target database during the precompilation (prep) phase. This is because some SQL validation is performed during this phase of the compilation.

During the precompilation phase, a bind file will be produced for each source module and may automatically be bound to the database. It is possible to delay this binding by using the 'BINDFILE' option with the preprocessor. The binding process produces a package on the database server. These packages are parsed forms of the SQL statements. By performing this operation at the

compilation phase, it is not necessary to perform it at run time. This may have significant performance benefits when the applications are executed.

Figure 24 is an example of the statement used to precompile and bind an application to the database.

```
$ db2 CONNECT TO sample

$ db2 PREF source.sqc BINDFILE

$ xlc -I/usr/lpp/db2_02_01/include -c source.c

$ xlc -o source source.o -L/usr/lpp/db2_02_01/lib -l db2

$ db2 BIND source.bnd
```

Figure 24. Commands Used to Create a Source Executable

9.2 Programming Languages

An application in DB2 is made up of several phases. These include setting up the application, connecting to the database, performing one or more units of work, disconnecting from the database, and ending the application.

During the setting up of the application, you need to declare all variables and data structures that will be used to interact with the database manager. For example, setting up the host variables to contain any table information returned by an SQL query.

The following sections discuss the host variables, the SQL Communication Area (SQLCA), the SQL Descriptor Area (SQLDA) structures, and some of the other important areas in application programming.

9.2.1 Host Variables

In DB2, host variables are declared within an SQL declare section. Oracle allows you to declare variables outside this declare section, but the FIPS (Federal Information Processing Standard) flagger will issue a warning message because the declare section is a part of the SQL standard.

The SQL declare section is a group of host program variable declarations that are preceded by the SQL statements, `begin declare section`, and ended with the `end declare section` statement. The definition and attributes of each host variable will depend upon how the variable is to be used within the SQL statements. Variables that receive and store data from the DB2 tables need to match the data type and length attributes of the tables column. An example of the different data type mappings for the C programming language is shown in Table 21 on page 121. For further information on the different data types, refer to Chapter 5, “Data Types” on page 53.

Table 21. Data Types in C / C++	
DB2 Column type	**C/C++ Data Type**
SMALLINT	short, short int
INTEGER	long, long int
FLOAT	double
DECIMAL(p,s)	No exact equivalent. Use the type double
CHAR(1)	char
CHAR(n)	No exact equivalent. Use the type char[n], where 1 <= n <= 254
VARCHAR(n)	struct tag { short int; char[n]; }, where 1 <= n <= 4000
LONG VARCHAR	struct tag { short int; char[n]; } 1 <= n <= 32700
CLOB	sql type is clob[n] 1 <= n <= 2147483647
BLOB	sql type is blob[n] 1 <= n <= 2147483647
DATE	char[10]
TIME	char[8]
TIMESTAMP	char[28]

A more complete listing of the supported data types is available in the *DB2 Application Programming Guide for common servers, Version 2*. The SQLDA record contains additional information about the data types in the SQLTYPE field, including information indicating if the column is a nullable or non-nullable column type.

An example of the declare section can be seen in Figure 25. The values shown as comments are examples of the values that would be found in the SQLTYPE field of the SQLDA record.

```
EXEC SQL BEGIN DECLARE SECTION;
    short      age=26;                  /* SQL type 500 */
    long       salary;                  /* SQL type 496 */
    double     wage;                    /* SQL type 480 */
    char       mi;                      /* SQL type 452 */
    char       name[6];                 /* SQL type 460 */
    struct     {
                short int len;
                char arr[24];
               } address;               /* SQL type 448 */
    sql type is clob(1m) chapter;       /* SQL type 408 */
    char       timestamp[26];           /* SQL type 392 */
    short      wage_ind;                /* Null indicator */
    char       (*arr)[10];              /* Ptr to a char of 10
                                           bytes must be referenced
                                           as :*mychar */
EXEC SQL END DECLARE SECTION;
```

Figure 25. Sample DECLARE SECTION in DATABASE 2 for AIX Version 2

Within the declare section of Oracle, you are able to include statements, such as exec sql include, exec sql type and exec sql var. DB2 does not permit the use of any exec sql statement within the declare section.

9.2.2 SQL Communication Area (SQLCA)

The SQLCA structure in DATABASE 2 for AIX Version 2 is similar to the same structure found in Oracle 7. The SQLCA data structure provides information for diagnostic checking and event handling. To declare the SQLCA structure in your application, you use the statement, `exec sql include sqlca`.

During the preprocessing of your application, the host language variable declarations replace the `include sqlca` statement. As mentioned in the previous section, the `include sqlca` statement must be outside the declare section of your application. The C/C++ representation of the SQLCA structure can be seen in Figure 26 for Oracle 7 and in Figure 27 for DATABASE 2 for AIX Version 2.

```
struct   sqlca
{
   char    sqlcaid[8];
   long    sqlabc;
   long    sqlcode;
   struct
   {
      unsigned short sqlerrml;
      char           sqlerrmc[70];
   } sqlerrm;
   char    sqlerrp[8];
   long    sqlerrd[6];
   char    sqlwarn[8];
   char    sqlext[8];
};
```

Figure 26. The Oracle 7 SQLCA Structure

```
SQL_STRUCTURE  sqlca
{
   char           sqlcaid[8];       /* Eyecatcher = 'SQLCA   '        */
   long           sqlcabc;          /* SQLCA size in bytes = 136      */
   long           sqlcode;          /* SQL return code                */
   short          sqlerrml;         /* Length for SQLERRMC            */
   _SQLOLDCHAR    sqlerrmc[70];     /* Error message tokens           */
   _SQLOLDCHAR    sqlerrp[8];       /* Diagnostic information         */
   long           sqlerrd[6];       /* Diagnostic information         */
   _SQLOLDCHAR    sqlwarn[11];      /* Warning flags                  */
   _SQLOLDCHAR    sqlstate[5];      /* State corresponding to SQLCODE */
}
```

Figure 27. The DATABASE 2 for AIX Version 2 SQLCA Structure

After an SQL statement is executed, the system will place a return code in both the `sqlcode` and `sqlstate` fields of the SQLCA structure. The `sqlcode` is an integer value that summarizes the execution of the statement, while the `sqlstate` is a character field that provides common error codes across the all the IBM relational database products. The `sqlstate` also conforms to the ANSI standards. The *DB2 Messages Reference Version 2* manual provides a listing of all the error codes and conditions that might be found in the SQLCA structure.

It is possible to let the system provide some control over checking for errors or warnings from the execution of SQL statements. This is provided by the whenever statement. The whenever statement provides handling for not-found conditions, warnings and errors. You may specify that upon the given condition an appropriate action is to be taken or simply continue execution of the application.

Note

DATABASE 2 for AIX Version 2 does not allow parameters 'DO' and 'STOP' with the whenever statement. For further details, refer to Chapter 7, "SQL Language Elements" on page 71.

9.2.3 SQL Descriptor Area (SQLDA)

The SQLDA structure provides a mechanism for applications to transfer data to the database manager as well as for receiving information from the database manager. The C/C++ representation of this structure is shown in Figure 28 for Oracle, and the DB2 structure is shown in Figure 29 on page 124.

```
struct SQLDA {
   long    N;         /* Descriptor size in number of entries */
   char  **V;         /* Ptr to Arr of addresses of main variables */
   long   *L;         /* Ptr to Arr of lengths of buffers */
   short  *T;         /* Ptr to Arr of types of buffers */
   short **I;         /* Ptr to Arr of addresses of indicator vars */
   long    F;         /* Number of variables found by DESCRIBE */
   char  **S;         /* Ptr to Arr of variable name pointers */
   short  *M;         /* Ptr to Arr of max lengths of var. names */
   short  *C;         /* Ptr to Arr of current lengths of var. names */
   char  **X;         /* Ptr to Arr of ind. var. name pointers */
   short  *Y;         /* Ptr to Arr of max lengths of ind. var. names */
   short  *Z;         /* Ptr to Arr of cur lengths of ind. var. names */
   };
```

Figure 28. The Oracle 7 SQLDA Structure

```
SQL_STRUCTURE  sqlname           /* Variable Name                  */
{
   short          length;        /* Name length                    */
   _SQLOLDCHAR    data[30];
                                 /* Variable or Column name        */
};

SQL_STRUCTURE  sqlvar            /* Variable Description           */
{
   short          sqltype;       /* Variable data type             */
   short          sqllen;        /* Variable data length           */
   _SQLOLDCHAR   *SQL_POINTER sqldata;
                                 /* Pointer to variable data value */
   short         *SQL_POINTER sqlind;
                                 /* Pointer to Null indicator      */
   struct sqlname sqlname;       /* Variable name                  */
};

SQL_STRUCTURE  sqlda
{
   char           sqldaid[8];
                                 /* Eye catcher = 'SQLDA   '       */
   long           sqldabc;       /* SQLDA size in bytes=16+44*SQLN */
   short          sqln;          /* Number of SQLVAR elements      */
   short          sqld;          /* # of columns or host vars.     */
   struct sqlvar  sqlvar[1];
                                 /* first SQLVAR element           */
};
```

Figure 29. The DATABASE 2 for AIX Version 2 SQLDA Structure

A mapping between the different SQLDA records can be found in Table 22.

Table 22. Mapping SQLDA in Oracle and DB2

Variable in Oracle 7	Variable in DATABASE 2 for AIX Version 2	Comment
sqlda->N	sqlda->sqln	Specifies the total number of variables currently allocated for the SQLDA.
sqlda->V[i]	sqlda->sqlvar[i].sqldata	Specifies a pointer to the data item.
sqlda->L[i]	sqlda->sqlvar[i].sqllen	Specifies the actual length of the data item.
sqlda->T[i]	sqlda->sqlvar[i].sqltype	Specifies the numeric value representing the data type of an item (see Table 21 on page 121).
sqlda->I[i]	sqlda->sqlvar[i].sqlind	Specifies a pointer to a short integer that is used to indicate whether or not the data item pointed to by the sqldata pointer is null or truncated.
sqlda->F	sqlda->sqld	Specifies the number of variables currently being used.
sqlda->S[i]	sqlda->sqlvar[i].sqlname.data	Specifies the fully qualified distinct type name for columns of distinct type when used by the describe or prepare statements.

To include the SQLDA structure using the SQL preprocessor directive, you would use the statement, exec sql include sqlda. It is also possible to include the header file with the C/C++ precompiler. This would appear as the statement, #include <sqlca.h>.

Once the definition of the SQDA structure has been included in the application, you will need to allocate space for the data structure. This is because the size of the structure is not fixed and will depend upon the number of distinct data items being passed. Oracle provides the following function to perform the allocation of space;

```
SQLDA *sqlald(int max_vars, size_t max_name, size_t max_ind_name);
```

DB2 defines a macro, called SQLDASIZE. The form of the definition is:

```
#define SQLDASIZE(n) (sizeof(struct sqlda) + (n-1) * sizeof(struct sqlvar))
```

This definition can then be used in the definition of the SQLDA structure, as follows:

```
struct sqlda *outda = (struct sqlda *) malloc(SQLDASIZE(1));
```

To maintain compatability with Oracle, the code example shown in Figure 30 allows you to allocate space in the same manner as it would in Oracle.

```
struct sqlda *sqlald(int p1,int p2,int p3)   /* Only p1 is used */
{
   struct sqlda *da;
   da=(struct sqlda *) malloc (SQLDASIZE(p1));
   if (da==NULL) {
        fprintf(stderr,"\nsqlald: out of memory error.\n");
        exit(-1);
   }
   strncpy(da->sqldaid,"SQLDA   ",8);
   da->sqldabc=(long)SQLDASIZE(p1);
   da->sqln=p1;
   da->sqld=0;
   return(da);
}
```

Figure 30. Sample sqlald

9.2.4 Static and Dynamic SQL

When the syntax of the embedded SQL statements are fully known at precompile time, the statements are referred to as static SQL. This is in contrast to dynamic SQL statements, which can have part or all of the statements specified at run time, instead of at precompile time.

Both Oracle 7 and DATABASE 2 for AIX Version 2 use static SQL as well as dynamic SQL. The important difference to note is that DB2 compiles the SQL statements and calculates the access plans to the data when the application is compiled. Oracle does this at run time. Because DB2 does not require the access plans do to be calculated at run time, you may see an increase in the performance. In general, static SQL statements will execute faster than dynamic SQL for this reason.

Figure 31 on page 126 is an example of a Oracle C program modified to run with DB2. The code enclosed in comments shows the old Oracle code, before the modification was made.

```
#include <stdio.h>
#include <sqlenv.h>

EXEC SQL INCLUDE sqlca;
EXEC SQL INCLUDE sqlda;

struct sqlda *sqlda; /* SQLDA *sqlda;*/

/* extern SQLDA *sqlald(); */
struct sqlda *sqlald(int p1,int p2,int p3) {
   struct sqlda *da;
   da=(struct sqlda *) malloc (SQLDASIZE(p1));
   if (da==NULL) {
      fprintf(stderr,"\nsqlald: out of memory error.\n");
      exit(-1); }
   strncpy(da->sqldaid,"SQLDA   ",8); da->sqldabc=(long)SQLDASIZE(p1);
   da->sqln=p1; da->sqld=0;
   return(da);
}

main() {
   EXEC SQL BEGIN DECLARE SECTION;
   /* VARVHAR username[20]; */
   struct { short int len; char arr[20]; } username;
   /* VARCHAR passwd[22]; */
   struct { short int len; char arr[22]; } passwd;
   char *stmt="select * from dept";
   EXEC SQL END DECLARE SECTION;

   int i,null_ok;

   strcpy(username.arr,"scott");  username.len=strlen("scott");
   strcpy(passwd.arr,"tiger");    passwd.len=strlen("tiger");

   EXEC SQL WHENEVER SQLERROR GOTO Error;

   EXEC SQL CONNECT to A user :username using :passwd;
   /*EXEC SQL CONNECT :username IDENTIFIED BY :passwd;*/

   sqlda=sqlald(5,5,5);

   /* for (i=1;i<sqlda->N;i++)
   sqlda->I[i]=(short *) malloc(sizeof(short *)); */

   EXEC SQL PREPARE s FROM :stmt;
   /* EXEC SQL DESCRIBE SELECT LIST FOR s INTO sqlda; */
   EXEC SQL DESCRIBE s INTO :*sqlda;

   sqlda->sqln=sqlda->sqld; /* sqlda->N=sqlda->F; */

   /* for (i=0;i<sqlda->F;i++) {
         printf("%d ==> \n\tcol. name(S) : %s\n\ttype(T):
         %d\n\tLong(C) : %d\n",i,sqlda->S[i],
         sqlda->T[i],sqlda->C[i]); } */

   for (i=0;i<sqlda->sqld;i++) {
      printf("%d ==> \n\tcol. name(S) : %s\n\ttype(T) :
      %d\n\tLong(C) : %d\n",i,
      sqlda->sqlvar[i].sqlname.data,
      sqlda->sqlvar[i].sqltype, sqlda->sqlvar[i].sqlname.length);
   }

   EXEC SQL CONNECT RESET; /* EXEC SQL COMMIT WORK RELEASE; */
   exit(0);

   Error :
      printf("Error detectee\n"); exit(1);
}
```

Figure 31. Sample C Program

9.3 Stored Procedures

In Oracle, a procedure or function consists of SQL statements and/or PL/SQL programming language statements that are written to perform a specific task. Procedures or functions are created in a user's schema and stored in the database. Oracle groups related procedures, functions, cursors and variables together as a unit which is referred to as a package. This package is stored in the database, and the procedures or functions may be explicitly called using the execute statement. For example,

```
EXECUTE pkgname.procname ('TITI',1035,NULL);
```

In DATABASE 2 for AIX Version 2, stored procedures can be used for database manager applications running in a client/server environment. This technique allows an application running on a client to call a procedure stored on at the database server. The server procedure executes and accesses the database locally and returns information to the client application.

To use this technique, the application is written in two separate parts. The calling procedure is contained in a client application and executes on the client. The server procedure executes at the location of the database that is on the database server. However, unlike Oracle, the procedure is located outside the database. Applications using this technique have the following advantages:

- Reduced network traffic
- Improved performance of server-intensive work
- Access to features that exist only on the database server

As mentioned above, the stored procedures are written in two separate parts. These are:

- The client application, which performs the following functions;
  1. Declares, allocates and initializes storage for the optional data structures and host variable.
  2. Connects to a database (`connect to dbname`).
  3. Invokes the server procedure through the `sql call` statement and passes the SQLDA structure to the server.
  4. Receives information back from the server procedure.
  5. Disconnects from the database (`connect reset`).
- The server procedure, which performs the following functions:
  1. Accepts the SQLDA data structure from the client application.
  2. Executes on the database server under the same transaction as the client application.
  3. Returns SQLCA information and optional output data to the client application.

Chapter 10. Backup and Restore

This chapter discusses how to back up your existing database and how to establish a backup and recovery methodology for your new DB2 environment.

Topics covered include:

Backing up you existing Oracle database

Restoring you Oracle database

Backup and Recovery strategies for DB2

Logging

10.1 Oracle 7 Database Backup

A major responsibility of the database administrator is to prepare for the possibility of hardware, software, network, process, or system failures. If such a failure effects the operation of a database system, it is usually desirable to recover the database and return to normal operations as quickly as possible.

Several problems can halt the normal operation of an Oracle database or affect the writing of database information to disk. These problems may include:

- User error
- Command failure
- Process failure
- Network failure
- Database instance failure
- Media (Disk) failure

It is always possible that a failure may occur during the conversion process. To safeguard against the possibility of a failure and to minimize the effect if one occurs, you should understand the following Oracle concepts and objects.

- Database Backups

  A database backup consists of operating-system-level backups of the physical files that constitute an Oracle database. To begin database recovery from a media failure, the backup files are used to restore the damaged data files or control files.

- The Redo Log

  The Redo Log, which is present for every Oracle database instance, records all changes made to the Oracle database. A database redo log is comprised of following two parts.

  – The online redo log

    This log works in conjunction with the Oracle background log writer process (LGWR) to immediately record all changes made through the associated instance. The online redo log consists of two or more pre-allocated files that are reused in a circular fashion to record ongoing database changes.

  – The archived (offline) redo log

Optionally, an Oracle database instance can be configured to archive files of the online redo log once they fill. The online redo log files that are archived are uniquely identified and make up the archived redo log. By archiving completed online redo log files, older redo log information can be preserved for more extensive database recovery operations, while the pre-allocated online redo log files continue to be reused to store the most current database changes.

- Rollback Segments

  These are used for a number of functions in the operation of an Oracle database. In general, the rollback segments of a database store the old values of data changed by uncommitted transactions. Among other things, the information in a rollback segment is used during database recovery to ″undo″ any ″uncommitted″ changes applied from the redo log to the data files. Therefore, if database recovery is necessary, the data is in a consistent state after the rollback segments are used to remove all uncommitted data from the data files.

- Control Files

  The control files store the status information about the physical structure of the database. Certain status information in the control files is used to guide Oracle during instance or media recovery. Information such as the current online redo log file, or the names of the data files, is stored in the control files.

10.1.1 Backup Strategy

There are three basic backup methods available to you in Oracle. These are listed below with a description of each.

- Offline backups

  Offline backups are full backups of data files, online redolog files and control files taken while the database is shut down. A full backup should also include the parameter files associated with the database. These files should be all you need to restore the database to the current point in time. In Oracle, offline backups are performed at the operating-system level using commands such as `tar`, `cpio` or `dd`.

- Online backups

  Online backups can be performed while the database is in use. This allows system activity to continue uninterrupted while the backup takes place. You must be in ″archivelog″ mode to perform online backups effectively. Note that online backups are useless without the archived redo logs (offline redo logs).

- Export

  Backing up by exporting files using the Export/Import utilities to back up tables and whole databases. The Export utility is usually not as desirable as the backup methods discussed above because it is much slower. Also, to recover a database, the import must recreate every database object one by one.

Most existing database environments will already have a backup strategy in place. If this is the case, you should make sure that you posses a current backup of your entire database environment. If you do not already have a strategy, then the following will provide you with some ideas on how to take a full backup of

your Oracle system. Also note that this section is for backing up and restoring Oracle. It does not describe how to back up Oracle data so that it may be restored to DB2.

Database administrators typically perform regularly scheduled offline backups during convenient time periods, such as weekends. They may also perform more frequent online backups at shorter intervals.

The frequency at which you back up your database system will be determined by a number of factors. These include the size of the database and how critical the information is. Also the availability of performing an offline backup may impact your backup policy.

Without control files, you cannot recover your database from any backup. Control files should be backed up each time you perform an offline backup. You must also have a copy of the control file, with the same file structure as your database to recover from an online backup. For this reason, it is a good idea to back it up immediately following the online backup procedure.

10.1.2 Redo Log Archiving

The redo log files of an Oracle server records changes made to database blocks when transactions are committed. These files can then be used to recover a damaged database by rolling forward from the most recently archived version of the database.

When the database is running in archivelog mode, the Oracle server daemon (ARCH) archives the completed online redo log files. These archived files can then be used to restore the database in case of software or hardware failure. When you use noarchivelog mode, the completed files are simply over written in a round-robin fashion.

The decision whether to operate in archivelog mode or noarchivelog mode depends on a balance of the costs and benefits of each mode. Weigh the cost of the additional disk space you will require against the cost of data loss and recovery time to arrive at a decision.

The following parameters will be set in the init.ora file if log archiving is being used.

- log_archive_dest

  Indicates the destination for the archiving of the redo log file.

- log_archive_format

  Indicates the name of the file created.

- log_archive_start

  Indicates whether archiving is to be automatic or manual when the instance is started.

10.2 Oracle 7 Recovery

In every database system, the possibility of a system failure is always present. Should system failure occur, the database must be recovered quickly, and with as little detrimental impact on users as possible.

Recovering from any type of system failure requires:

1. Determining which data structures have been corrupted or lost
2. Restoring the corrupted data from an archive/backup
3. Restarting the database and possibly replay logs (rollforward)
4. Checking database integrity

The aim is to return to normal operations as quickly as possible and, at the same time, to insulate database users from any problems, such as the loss or duplication of work. The recovery process varies, depending on the type of failure and the database files affected by the failure.

Recovery from instance failure

To recover from an instance failure, you may simply need to shut down the instance, possibly using the ′IMMEDIATE′ or ′ABORT′ options for the shutdown process. Then restart the instance as normal. Automatic recovery procedures will perform the following steps:

- Roll forward all transactions in the redo log files
- Roll back uncommitted transactions
- Release all locks on Oracle resources

You should now be able to return to normal operations.

Recovery from media failure

Media failure happens when a non-recoverable error occurs during a read or write operation involving one or more database files. For example, a disk-head crash that destroys any files associated with a database constitutes a media failure.

There are two different methods for recovering from a media failure. The method to be used depends upon the archive method being used.

- Noarchivelog

  Recovery from a media failure is a simple restoration of the most current full backup. All work performed after the most recent full backup is lost.

- Archivelog

  Recovery from a media failure involves an actual recovery procedure to reconstruct the damaged database to a specified transaction consistent state that existed prior to the media failure. This will involve restoring the backup and the replay of logs. You should refer to your Oracle documentation for detailed information on this procedure.

10.3 DATABASE 2 for AIX Version 2 Backup

DATABASE 2 for AIX Version 2 provides several different mechanisms to back up the database or the individual tablespaces. You should be familiar with these before setting up your database environment because backup considerations may influence the way in which you spread data across the tablespaces.

Like all database systems, DB2 is susceptible to database corruption due to events such as user error or media failures. DB2 provides the capability to back up the databases at different levels. This chapter will discuss the different methods available.

10.3.1 Backup Methods

There are several methods available to back up existing DB2 databases. These include:

- Offline database-level backup
- Online database-level backup
- Offline tablespace-level backup
- Online tablespace-level backup

Unlike Oracle, DATABASE 2 for AIX Version 2 provides backup and restore capabilities at the database-manager level. The backup command is provided by the database manger, and the database manager will keep track of what backups have been performed. When performing a database backup using the database manager, there is no additional requirement to back up configuration files.

Using the backup command, you may back up the entire database or simply back up individual tablespaces. Also, a backup in DATABASE 2 for AIX Version 2 may be performed while the database is either online or offline.

10.3.1.1 Database-Level Backup

It is advisable to back up the database on a regular bases. This backup may be to disk, tape or to ADSTAR Distributed Storage Manager (ADSM).

To perform an online backup, you need to retain the database logs. This is similar to Oracle′s archivelog mode and is turned on by setting the ′LOGRETAIN′ parameter on for the database. The command to do this is:

```
UPDATE DATABASE CONFIGURATION FOR database-name USING LOGRETAIN ON
```

The database backup may be performed through the database director or via the command line. Examples of the command line version of the backup command may be seen in Figure 32 on page 134.

10.3.1.2 Tablespace-Level Backup

If you have an extremely large database, then you may find that backing up at the tablespace level is a more viable option. At this level, you are able to backup the individual tablespaces, rather than the entire database. As with the database backup, a tablespace backup may be performed either online or offline. To ensure that restored tablespaces are synchronized with the rest of the database, a tablespace being restored must be rolled forward to the end of the logs. For this reason, tablespace-level backup and restore can be performed if

roll-forward recovery is enabled. To enable Roll-forward recovery, you need to turn 'LOGRETAIN' on. Also, before log archiving may be enabled, you must take a full offline backup of the entire database. This is done to provide a starting point for the logs. This is discussed in more detail in 10.3.2, "DATABASE 2 for AIX Version 2 Recovery" on page 137.

If the database backup is written to disk, a single file will be generated containing the backup data. The file being generated will use the following naming convention.

`dbname.type.db2instance.node.yyyymmddhhmmss.seq`

where

dbname	Database Name or Alias
type	0 for a database level backup or 3 for a tablespace level backup
db2instance	Instance Name
node	Reserved
yyyymmddhhmmss	Timestamp
seq	Sequence number

An example of the files created can be seen in Figure 32. Also, information about backup and restore operations may be viewed through the database director or by using the command `list backup` or `list history`.

```
$ db2 BACKUP DATABASE A  TO /u/jcb/SVG
Backup successful. The timestamp for this backup image is : 19950509174543
$ ls /u/jcb/SVG
A.0.jcb.0.19950509174543.001

$ db2 BACKUP DATABASE A ONLINE TO /u/jcb/SVG
Backup successful. The timestamp for this backup image is : 19950509175629

$ ls /u/jcb/SVG
A.0.jcb.0.19950509174543.001
A.0.jcb.0.19950509175629.001

$ db2 BACKUP DATABASE A TABLESPACE users ONLINE TO /u/jcb/SVG
Backup successful. The timestamp for this backup image is : 19950509175931

$ ls /u/jcb/SVG
A.0.jcb.0.19950509174543.001
A.0.jcb.0.19950509175629.001
A.3.jcb.0.19950509175931.001
```

Figure 32. DB2 Backup Examples

10.3.1.3 Planning Your Backups

A proper backup of the database is critical. You need to be sure that you are going to be able to use the backup to recover correctly. You will need to make sure that you backup will allow you to restore the database to a consistent state. If you already have a backup strategy for you Oracle database then you should be able to map this to a backup strategy for DB2. However, due to some the differences in DB2 you may like to modify the strategy to make full use of the additional backup features found in DB2. Some of these area that will require further consideration include:

- Who will be able to perform backups?
- Will the backups be online or offline?
- Will I use logging and how will I manage archiving log files?

As you are able to assign different levels of authority to users in DB2, you should decide who will be able to perform backups on the different databases. Users with SYSADM, SYSCTRL or SYSMAINT may perform a backup of any database. SYSADM and SYSCTRL users may then restore the backup to an existing database or to a new database, while SYSMAINT may only restore to an existing database. A user with DBADM authority on a database may export or load data for that database only.

Once you have decided who is going to be able to perform database backups, you then need to decide where. If you have been backing up your Oracle database, then you will probably use the same device to back up in DB2.

Deciding on which databases will have logging turned on is also something that will need consideration. As DB2 allows multiple databases, you may decide to split up your existing Oracle database into separate DB2 databases. If they can be logically split, then you may also be able to use circular logging for less critical information while using archive logging for the more crucial information.

Other factors will influence the above decisions, such as:

- Can I bring my database offline to perform a backup?
- Can I run a backup unsupervised?
- Will my backup fit onto a single tape, or will I need to change tape?

Once you have decided on the backup strategy to use, you will need to test it. In actually performing the backup, there are a few points that you will need to consider. These include:

- Unlike Oracle, when performing an offline backup, the database manager will need to be running. The offline backup process needs to obtain an exclusive lock on the database being backed up.
- You must not include temporary tablespaces in your backups. If you do include a temporary tablespace, the backup will fail.
- If a system crash occurs during a critical stage of backing up a database, you cannot successfully connect to the database until you reissue the backup command and successfully complete the backup. The database will be in a backup pending state until the backup is successfully performed.
- During the backup procedure, an internal buffer is filled with data to be backed up. When this buffer becomes full, the data is copied to the backup

medium. You have the ability to choose multiple buffers and I/O streams to improve the performance of the backup procedure.

- The recovery history file is updated automatically with summary information whenever you carry out a backup or restore.

Table 23 lists the different types of backups available and compares the steps required by Oracle and DB2 to perform them.

BACKUP	Oracle 7	DATABASE 2 for AIX Version 2
database offline	• SHUTDOWN database • cpio or tar (data files, control file, redo log files, init.ora, config.ora) and dd for the raw devices. • STARTUP database	• No users/applications connected to the database • BACKUP DATABASE dbname TO path/device
database online	• Database started and archive on • for each tablespace of the database, do ALTER TABLESPACE ts_name BEGIN BACKUP • cpio or tar (data files, control file, redo log files, init.ora, config.ora) and dd for the raw devices. • For each tablespace of the database, do ALTER TABLESPACE ts_name END BACKUP • Backup control file used: ALTER DATABASE BACKUP CONTROLFILE TO name_cf_svg and saved name_cf_svg	• Database started and ′LOGRETAIN′ on • BACKUP DATABASE dbname ONLINE TO path/device
tablespace offline	• SHUTDOWN database • cpio or tar (data files of the tablespace, control file, redo log files, init.ora, config.ora) AND dd for the raw devices. • STARTUP database	• Database started, archived ON, and must be accessed in exclusive mode • BACKUP DATABASE dbname TABLESPACE ts_name TO path/device
tablespace online	• Database started and archived ON • For the tablespace that you want save, do ALTER TABLESPACE ts_name BEGIN BACKUP • cpio or tar (data files) AND dd for the raw devices. • ALTER TABLESPACE ts_name END BACKUP • Backup control file used: ALTER DATABASE BACKUP CONTROLFILE TO name_cf_svg and saved name_cf_svg	• Database started and archived ON • BACKUP DATABASE dbname TABLESPACE ts_name ONLINE TO path/device

Table 23. Backup in DB2 and Oracle

It is possible to back up the database without using the DB2 backup utilities. As with Oracle, you may bring the database instance offline, and use the operating system to backup the files/containers. However, using this method does not record any history information within the database and may make it more difficult to recover log files and perform a roll forward recovery.

10.3.2 DATABASE 2 for AIX Version 2 Recovery

There are three ways to recover a damaged database. The method you use will depend upon the type of backups that you have made and the severity of the damage. The first of these three methods is Crash Recovery. This is the method used if transactions or the database is abnormally terminated due to power loss or perhaps a network failure. If the actual data integrity is lost due a more severe failure, such as a corrupted disk, then the second method of restore recovery might be used. This is restoring the database or tablespace from your backup media. Finally the third method of recovery is Roll-forward recovery. This may be used after a restore recovery and involves replaying the log files to bring the database back to a consistent state just before the failure. A Roll-forward recovery may also be used if after a crash recovery, one or more tablespaces are left in a Roll-forward pending state. These scenarios are covered in more detail in the following sections.

10.3.2.1 Crash Recovery

Should your system fail while transactions are in progress, Crash Recovery will try to return your system to a consistent state. Crash Recovery may be started by using the `restart database` command or by enabling automatic restart. Automatic restart is enabled through the 'AUTORESTART' parameter. During Crash Recovery, any transaction that has not committed will be rolled back. Once these transactions have been rolled back then your database should be in a consistent state and ready for users to continue operations.

Figure 33 is an example of the timeline for a Crash Recovery scenario. The three transactions are incomplete when the database crash occurred. This means that all three transactions will be rolled back. All transactions that were not committed at the time of the failure will need to be redone.

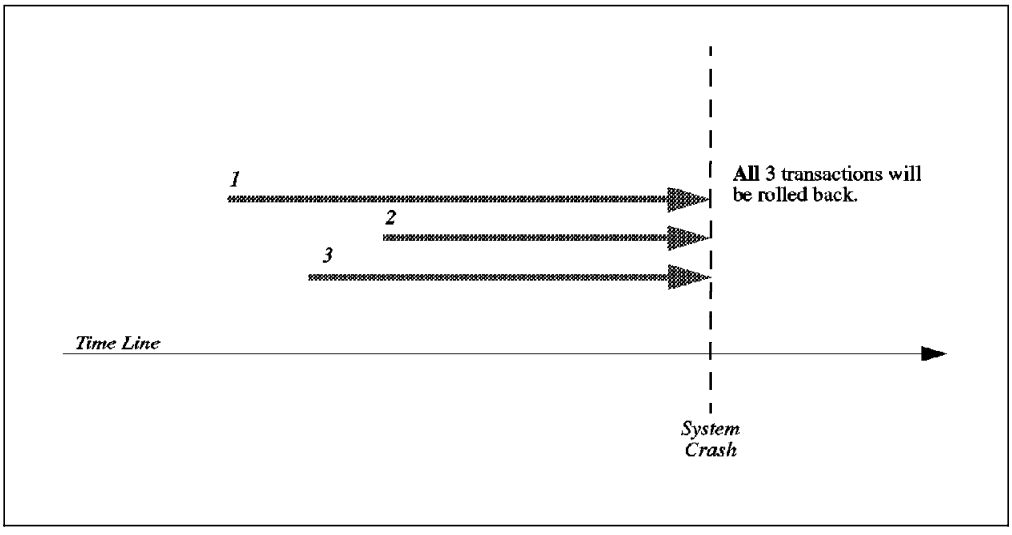


Figure 33. Crash Recovery Timeline

It is possible that one or more tablespaces many be left in a Roll-forward pending state after a crash recovery. Any tablespaces in this state will require further recovery before they may be accessed by users. This is covered in further detail in 10.3.2.3, "Roll-forward Recovery" on page 138. Other tablespaces that are not in this state may be accessed by users.

10.3.2.2 Restore Recovery

On occasion, crash recovery may not be able to restore your database to a consistent state. This may be due to problems such as disk corruption. In this situation, it would be necessary to recover your database from a backup. Figure 34 is an example of the timeline for a Restore Recovery.

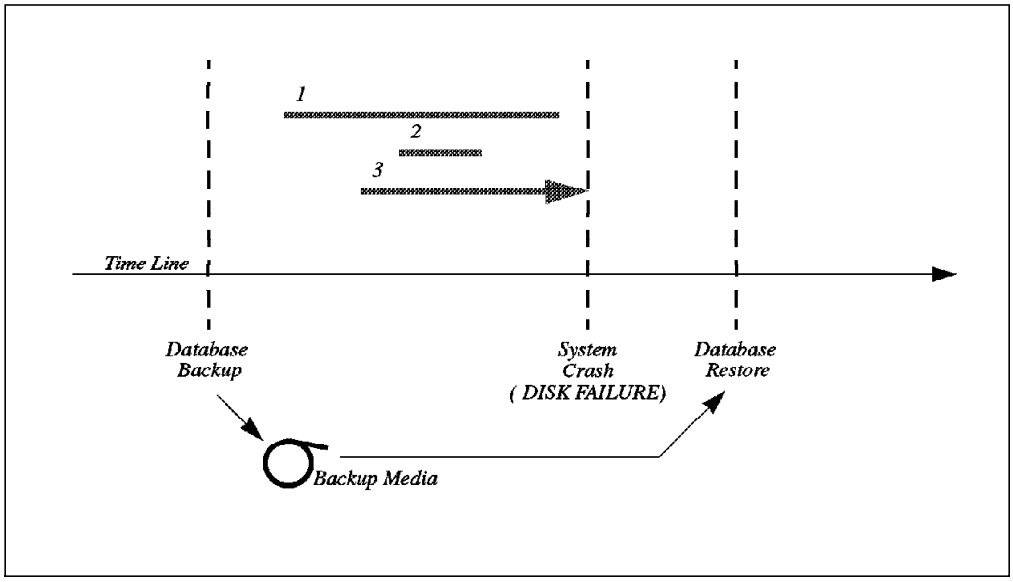


Figure 34. Restore Recovery Timeline

When performing a Restore Recovery, the transactions completed after the backup was taken will be lost.

10.3.2.3 Roll-forward Recovery

For most database environments, this will be the most common type of recovery required, if the Crash Recovery is unable to correct any inconsistencies after a failure. There are two types of Roll-forward recovery available. The first is a database-level recovery and the second a tablespace-level recovery. The timeline shown in Figure 35 would be similar for either type of recovery.

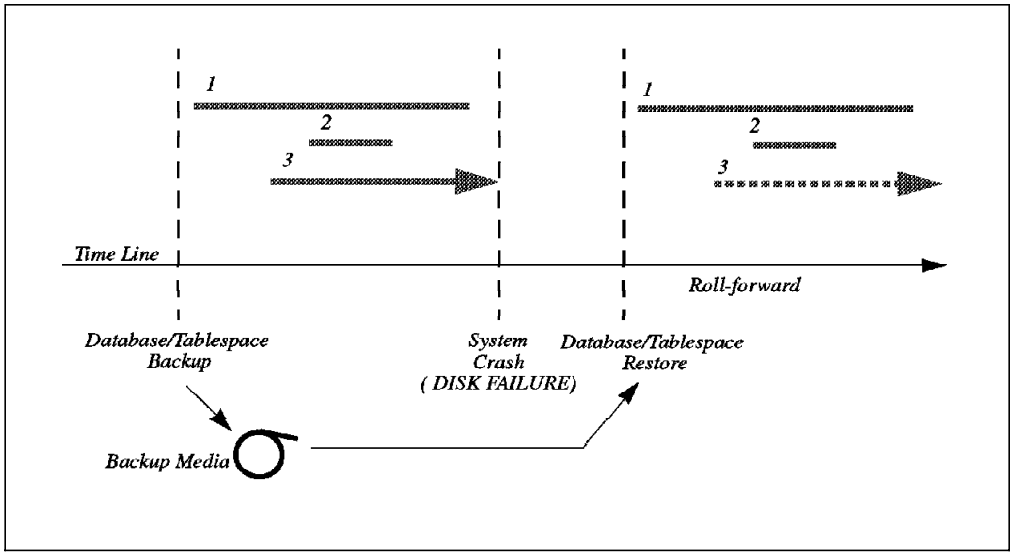


Figure 35. Roll-Forward Recovery Timeline

After restoring the backup, you may need to replay the logs. By performing a Roll-forward on the logs, you will be replaying any transactions that were performed after the database- or tablespace-level backup.

Looking at the timeline in Figure 35 on page 138, we can see that transactions 1 and 2 were completed (committed) before the crash occurred. However, transaction 3 was still in progress. During the Roll-forward process, all transactions are re-played. At the end of the logs any transaction that has not been committed will be rolled back. In the above example, transaction 3 will be rolled back.

Roll-forward Recovery must be enabled before tablespace-level backups and restores will be allowed. To enable Roll-forward Recovery, you must enable 'LOGRETAIN'. This is done through the database directory or via the `update database configuration` command. You must also have a full database backup before 'LOGRETAIN' will be enabled. The is to provide a starting point for log recovery.

10.3.3 DB2 Load Utility

When migrating data from the Oracle environment to DB2, the load utility will be used to load the data. The load utility is intended for the initial load or append of large amounts of data into database tables.

The load utility will handle delimited ASCII files (DEL), non-delimited or aligned column ASCII files (ASC) or the PC Version of the Integrated Exchange Format (IXF). It will handle all the DB2 data types, including large objects (LOBs) and user defined types (UDTs).

The load utility is much faster than performing numerous SQL inserts, as is done by the `import` command. This is because load will write formatted pages into the database rather than performing multiple inserts into the database. There are four basic phases involved in the load process. These are:

1. Load

   The data is loaded into the tables page by page.

2. Build

   Index are created/modified on the table.

3. Delete

   Rows that caused a unique key violation are removed from the table.

4. Setting state

   Finally, the removal of the check pending state from the tables.

For further information on the load utility, you should refer to *DATABASE 2 Administration Guide Version 2*.

10.4 Logging

The previous sections have discussed logging as a part of roll-forward recovery. To perform a Roll-forward Recovery, you would use archival logging which is done by setting 'LOGRETAIN' on. If 'LOGRETAIN' is not turned on, the default type of logging is used. This is called circular logging.

10.4.1 Circular Logging

As mentioned above, circular logging is when 'LOGRETAIN' is not enabled. This also means that Roll-forward Recovery is not possible.

Circular logging is the default type of logging for DATABASE 2 for AIX Version 2. This involves the allocation of a defined number of logfiles with a defined size. The default number of logfiles is three primary logs, and two secondary logs The default size of these logfiles is 1000 * 4 KB pages (4 MB). The primary logfiles will be preallocated when the database is created. The secondary logfiles will be dynamically created if they are required.

These log files are used in a circular fashion. This means that logging information will be lost when the logfile is reused by the database manager. Circular logging is useful for the first recovery method (Crash Recovery).

If you require that the logs be saved so that you may perform Roll-forward Recovery, you need to use archival logging.

10.4.2 Archival Logging

If either 'LOGRETAIN' or 'USEREXIT' is enabled, the log files will be retained and become online archive log file for use in Roll-forward Recovery. This is called log retention or archive logging. Unlike circular logging, the logfiles are not normally reused.

The numbering scheme for the archival logging is Sxxxxxxx.LOG, where xxxxxxx ranges from 0000000 to 9999999. The log files are stored in the directory specified in the database configuration and may be defined when the database is created or modified at a later date.

In Oracle 7, when you use the archivelog mode, the log files are always used in circular mode, but when a log files becomes inactive, it is stored in the log_archive_dest subdirectory. DB2 does not work in this fashion. As mentioned above, the logfiles are not reused when archival logging is enabled. It is up to the user to back up these log files. To assist in this, there is a user exit capability that may be invoked whenever a logfile is closed.

10.4.3 User Exits

User exits allow you to specify your own program/application to be used for the archiving and retrieval of log files when using a user exit Roll-forward Recovery is allowed, and 'LOGRETAIN' is not required. This gives you more flexibility in managing the logging environment.

As with turning 'LOGRETAIN' on, you must take a full database-level backup immediately before or after enabling the user exit capability.

Different combinations of using 'LOGRETAIN' and 'USEREXIT' are possible. Table 24 on page 141 summarizes the possible combinations and includes a short description of the effect.

When the user exit program is invoked, the database manager passes control to the executable file, db2uexit. Samples of the user exit programs are available in the sample directory that comes with DATABASE 2 for AIX Version 2. These include:

- db2uexit.cadsm

This program uses the ADSTAR Distributed Storage Manager utility to archive and retrieve database log files.

- db2uexit.ctape

  This program archives and retrieves the database log files using tape media.

- db2uexit.cdisk

  This program archives and retrieves the database log files using disk media.

These are only sample programs and may require modification to suit your environment. Detailed information on user exits can be found in the *DATABASE 2 Administration Guide Version 2*

LOGRETAIN	USEREXIT	Description
NO	NO	Circular logging and Roll-forward is disabled. Logs are stored in the SQLOGDIR.
YES	NO	Archival logging with Roll-forward enabled. Logs are stored in the SQLOGDIR subdirectory.
NO	YES	Archival logging with Roll-forward enabled. Logs are used by the user exit program and are erased from the SQLOGDIR subdirectory.
YES	YES	Archival logging with Roll-Forward enabled. Logs are used by the user exit program and are kept in the SQLOGDIR subdirectory.

Table 24. 'LOGRETAIN' and 'USEREXIT' Combinations

Chapter 11. Performing a Conversion

This chapter discusses the actual process of performing a conversion on a sample Oracle environment. The chapter may be used as a guideline to performing your own conversion process, but should be used after you are familiar with the concepts covered throughout this book.

Each Oracle environment being converted will be different in some way, and it would be impossible to try and cover all facets of the conversion process in this chapter.

11.1 Overview

The conversion process from Oracle 7 to DATABASE 2 for AIX Version 2 may be separated into two different types of operations:

1. Operations which can be automated. The automated conversion of the following objects is possible.
   - DDL
   - Data
   - Synonym
   - Users
   - Indexes
   - Grant
2. Operations that will have to be carried out manually. The following objects are different to the extent that converting from Oracle to DB2 is a completely manual process.
   - Views
   - Triggers
   - Procedures
   - Applications

Where possible, we have included the SQL statements or sample scripts that will aid you in the conversion process. You may need to modify these scripts to better suit you own environments, but they provide a good base on which to start.

11.2 Extracting Oracle Tablespaces, Users and Roles

The first phase of conversion is to find out what we are dealing with by obtaining object/structure information from Oracle. This information includes tablespaces, users, roles, indexes, tables, and so on.

The following Oracle database example, shown in Figure 36 on page 144, is used for this sample conversion.

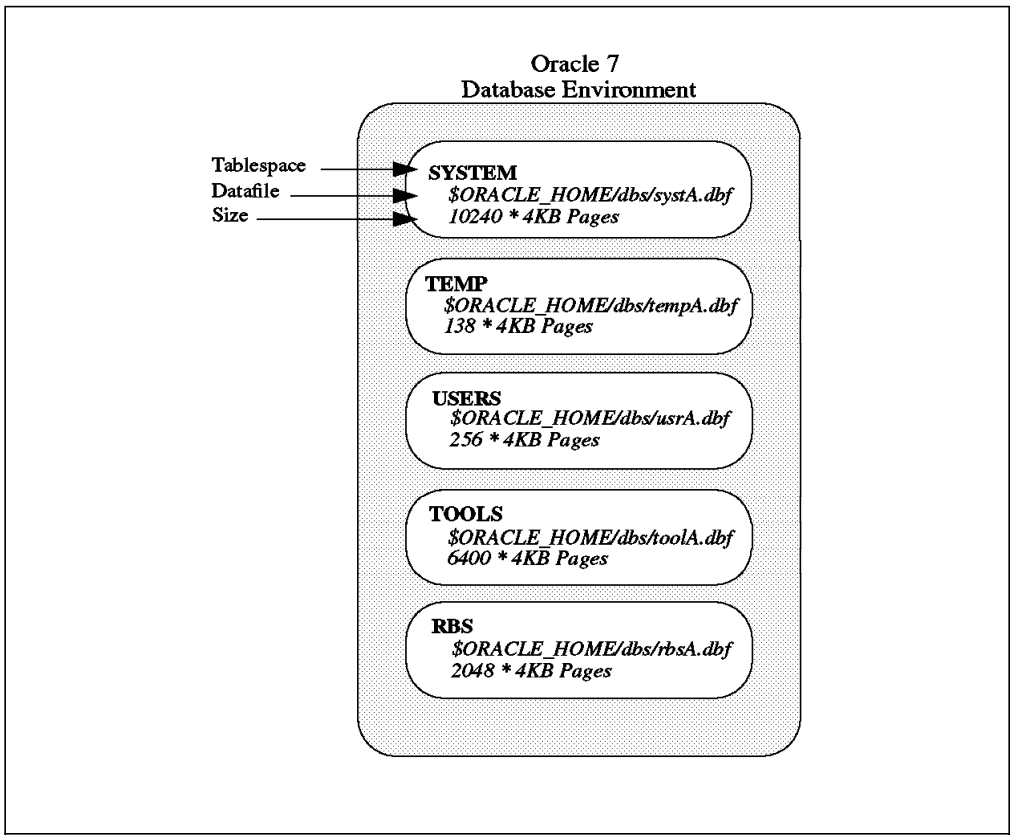


Figure 36. Tablespace Organization in Oracle

To obtain a text description of the database and the objects contained within the database, the Oracle exp utility is used. The chapter will refer to an export file. This file is generated by using the following command;

```
exp sys/password full=y rows=n file=oracle.export
```

The Oracle user 'sys' is used to obtain the information and the appropriate password should be given. This file should be generated and stored as it will be used throughout this chapter.

11.2.1 Extracting Tablespace Information

To build equivalent tablespace structures in DB2, you will need to obtain the tablespace information from Oracle. You are able to obtain this information from the export file by using the command:

```
$ grep " CREATE TABLESPACE " export.file
```

or by using the SQL query:

```
SQL> set head off
SQL> select TABLESPACE_NAME ||' ' || FILE_NAME ||' '|| BYTES/4096
  2  from dba_data_files
  3  order by TABLESPACE_NAME;

RBS /home/oracle/dbs/rbsA.dbf 2048
SYSTEM /home/oracle/dbs/systA.dbf 10240
TEMP /home/oracle/dbs/tempA.dbf 138
TOOLS /home/oracle/dbs/toolA.dbf 6400
USERS /home/oracle/dbs/usrA.dbf 256
```

We can base the size and structure of the new DB2 database on this tablespace information.

11.2.2 Extracting Roles and Users

We now need to find out what roles are defined within the Oracle environment. The export file contains the Oracle `create role` commands for all the roles within Oracle. Using the Korn shell script shown in Figure 37, we can extract these lines and convert them into operating-system commands that will create groups for each role.

```
#!/usr/bin/ksh
cat $1 | sed -e '/ENDSYS/,$d' | grep "CREATE ROLE " | \
         sed -e 's/CREATE ROLE/mkgroup/g'| \
         awk -F"'" '{ print $1,tolower($2) }'
```

Figure 37. Korn Shell Script for Converting Roles

The command to use this script would be:

```
roles.ksh oracle.export > db2_groups.ksh
```

In this example, the Korn shell script is called `roles.ksh`. The Oracle export file is `oracle.export` and the output is redirected to the file `db2_groups.ksh`. The file `db2_groups.ksh` will now contain the necessary commands for creating groups that match the Oracle roles.

A similar command and script may be used for creating the users. Figure 38 is an example of the Korn shell script for extracting the users from the export file and creating an operating-system command to create each of the users.

```
#!/usr/bin/ksh
cat $1 | sed -e '/ENDSYS/,$d' | grep "CREATE USER " | \
         sed -e 's/CREATE USER/mkuser/g'|\
         awk -F"'" '{ print $1,tolower($2) }'
```

Figure 38. Korn Shell Script for Converting Users

The commands generated for creating users and group use all the default system options. You may need to modify these so that the users are added to the correct default group. Adding the users to the correct groups (roles) is covered in 11.2.3, “Translating Granted Roles to Group Membership” on page 146.

As the use of the users 'sys' and 'system' is different in DB2, more consideration needs to be given to these users. It is possible to create these users under the operating system, and add them to the instances primary group. This will give the users system administration authority for the instance. However, it is recommended to use the DB2 groups `SYSADM`, `SYSMAINT` and `SYSCTRL` for granting users the different levels of administrative authority.

11.2.3 Translating Granted Roles to Group Membership

The granting of roles to a user now equates to adding the users to the operating-system group that is equivalent to the role. The export file contains a list of all the grant commands. The commands dealing with granting roles to users needs to be extracted and changed into chgroup commands. The script shown in Figure 39 is a sample script that will do the extraction and execution of the operating-system mkgroup command.

```
#!/usr/bin/ksh
# Extract the Roles and convert into Operating system mkgroup commands
#

# Function to extract created roles from export file
#
function get_roles
{
cat $1 | sed -e '/ENDSYS/,$d' | grep "CREATE ROLE " | \
    awk -F"' '{  print tolower($2) }'
}

# Given role extract grant commands for users
#
function get_user
{
cat $1 | sed -e '/ENDSYS/,$d' | grep -i "^GRANT \"$2\"" | \
    awk -F"' '{print tolower($4)}'
}

# Main Loop
#
for role in get_roles $1
do
   for user in get_user $1 $role
   do
      # Extract current users for the group/role
      usr_list=lsgroup -a users $role | cut -d'=' -f2

      # Check if the list is empty
      if [ "$usr_list" = " " ]
      then
         # Add the user to the group
         chgroup users=$user $role
      else
         # Check if user already in group
         echo $usr_list | grep $user > /dev/null 2>&1
         if [ $? -ne 0 ]
         then
            new_list=echo $usr_list$user | sed -e 's/ /,/g'
            chgroup users=$new_list $role
         fi
      fi
   done
done
```

Figure 39. Korn Shell Script for Adding Users to Groups/Roles (grant.ksh)

Note that example script for adding users to groups performs no error checking. Also, unlike the first two examples, it will execute the chgroup command. The script needs to be executed as root: otherwise you will get 'permission denied' errors on the chgroup command.

11.3 Creating the DB2 Database and Environment

Before we start creating tablespaces and database objects that match the Oracle environment, we need to create a database within the DB2 instance. An operating-system user and group should be selected to become the instance owner and instance group. These may be existing users or a new user and group. Once they have been decided upon, you will need to run the `db2icrt` command as the root user. This will set up the instance environment.

When the DATABASE 2 for AIX Version 2 database is created, there are three default tablespaces created. These are:

- SYSCATSPACE, for the system catalogs
- TEMPSPACE1, for the temporary tables
- USERSPACE1, for user tables

If these default names will fit into your new environment, then you only need to select the size and location of the tablespaces. If you need to change the names of the temporary and user tablespaces, you need to create further tablespaces.

For our example, we are going to create new tablespaces for the temporary tablespace and the user tablespace. So, we only need to be concerned with the size and location of the system tablespace at this stage.

The name of the system tablespace cannot be changed, but you may select its size and location. We will use a System Managed tableSpace (SMS) for the system catalogs. For an SMS tablespace, we only need to select the location, or we may let it default to the instance owner's home directory. We have chosen to take the default; so the create database command would look like the following:

```
$ db2 CREATE DATABASE dbname
```

If you chose to place the user and tablespace containers into the directory `/tmp/db` while you set up the correct tablespaces and containers, you would use the commands:

```
$ mkdir -p /tmp/db/usr /tmp/db/tmp
$ db2 "CREATE DATABASE dbname
>       USER TABLESPACE MANAGED BY SYSTEM ('/tmp/db/usr')
>       TEMPORARY TABLESPACE MANAGED BY SYSTEM ('/tmp/db/tmp')"
```

11.3.1 Creating Tablespaces

Once you have a database created, then you can start building the tablespaces that corresponds to the tablespaces from Oracle. In the previous steps, we extracted the tablespace information from Oracle and found that the following tablespaces existed.

Tablespace Name	Data file/container	Size in 4 KB Pages
RBS	/home/oracle/dbs/rbsA.dbf	2048
SYSTEM	/home/oracle/dbs/systA.dbf	10240
TEMP	/home/oracle/dbs/tempA.dbf	138
TOOLS	/home/oracle/dbs/toolA.dbf	6400
USERS	/home/oracle/dbs/usrA.dbf	256

We will not recreate the RBS or the TOOLS tablespaces as they are not required by DB2 This leaves us with the SYSTEM, TOOLS and USERS tablespaces.

The SYSTEM is going to remain as the default SMS tablespace created with the database was created. This tablespace resides under the instance owners home directory.

For the remaining two tablespaces, we are going to create then in a filesystem called /data. These will be Database Managed tableSpaces (DMS), and we are going to define the sized to match the sized defined by Oracle. The commands to create these tablespaces are as follows:

```
$ db2 " CREATE REGULAR TABLESPACE users
>        MANAGED BY DATABASE USING (FILE '/data/users.dat' 256)"

$ db2 " CREATE TEMPORARY TABLESPACE temp
>        MANAGED BY DATABASE USING (FILE '/data/temp.dat' 138)"
```

With DB2, it is possible to create a separate tablespace for any large objects that may be contained in files. Also, you may decide to create a tablespace for indexes. To create these tablespaces, you would use the following:

```
$ db2 " CREATE REGULAR TABLESPACE idxts
>        MANAGED BY DATABASE USING (FILE '/data/index.dat' 500)"

$ db2 " CREATE LONG TABLESPACE lobts
>        MANAGED BY DATABASE USING (FILE '/data/lobs.dat' 10000)"
```

When creating the tablespace, you must be connected to the database in which they are going to be assigned because this happens during the create phase. Once you have created the tablespace, you may drop the default tablespaces that are no longer required. The order in which you create and drop the tablespaces is not important, except for the temporary tablespace. At all times, at least one temporary tablespace must exist.

11.3.2 Granting Access to the Database

The granting of access to the database will require some consideration. The first step is to set up the user environment. This is simply done by including the file sqllib/db2profile into the user's logon profile. In our example, the instance owner is the user db2 and the instance home directory is /home/db2. For this example, each users profile would include the line:

```
. /home/db2/sqllib/db2profile
```

This sets up the environment for the user to access the instance. The next phase it to grant connect privilege to the database. The way in which you do this will depend upon the environment. Your choices are:

1. Grant access to everyone:

   ```
   db2 CONNECT TO databasename
   db2 GRANT CONNECT ON DATABASE TO PUBLIC
   ```

2. Create a group for the database, and grant access to members of the group:

   ```
   mkgroup new_group (This needs to be done as root)
   db2 CONNECT TO databasename
   db2 REVOKE CONNECT ON DATABASE FROM PUBLIC
   db2 GRANT CONNECT ON DATABASE TO PUBLIC
   ```

3. Grant access to the individual users:

   ```
   db2 CONNECT TO databasename
   db2 REVOKE CONNECT ON DATABASE FROM PUBLIC
   ```

```
db2 GRANT CONNECT ON DATABASE TO user1,user2,group1
```

You may extract the grant commands from the Oracle export file and use these commands to grant connect privilege to either users or group/roles. The script shown in Figure 40 will build the DB2 commands.

```
#!/usr/bin/ksh

#
# Find all grant connect users
#
function connects
{
cat $1 | sed -e '/ENDSYS/,$d' | grep '^GRANT "CONNECT"' | \
         awk '{ print tolower($4) }' | tr -d '"'
}

#
# Create db2 connect command for each user, group/role
#
for principal in connects $1
do
   # Check if principal is a user
   lsuser $principal >/dev/null 2>&1
   if [ $? -eq 0 ]
   then
      echo "db2 grant connect to user $principal"
   fi

   # Check if principal is a group
   lsgroup $principal >/dev/null 2>&1
   if [ $? -eq 0 ]
   then
      echo "db2 grant connect to group $principal"
   fi
done
```

Figure 40. Grant Connect Permission to DB2

Note that the conn.ksh script above will grant connect permission to both a user and a group if they have the same name.

11.4 Table, Views, Data, Constraint, and Index Conversion

The conversion process for the objects contained in this section will need to be repeated for each user or schema in the Oracle environment. For our examples, we will use the default user scott.

Again, where possible, scripts have been included to help you in the conversion process.

11.4.1 Table Conversion

This phase will extract the create table statements from Oracle export files and convert them into DB2 statements.

The first step is to use the exp utility to extract the definitions for the objects owned by that user. An example of this command would be:

```
$ exp scott/tiger rows=n file=scott.export
```

Once we have this file, we may start modifying the command to the DB2 environment. The sample script shown in Figure 41 on page 151 is an example that will help to modify the `create table` statements.

```
#!/usr/bin/ksh

# Modify PCTFREE and TABLESPACE parameters
#
function pct_tab
{
    sed -e 's/\(^.*\)PCTFREE\(.*\)TABLESPACE\(.*\)$/\1IN\3/g'
}

# Convert Number Datatypes
#
function dt_num
{
    sed -e 's/ REAL/ FLOAT/g' \
        -e 's/ NUMBER(\*/ NUMERIC(31/g' \
        -e 's/ NUMBER/ NUMERIC/g' \
        -e 's/ FLOAT([0-9]*)/ float/g' \
        -e 's/ FLOAT(\*)/ float/g'
}

# Convert Binary Datatypes
#
function dt_bin
{
    sed -e 's/ LONG RAW/ BLOB(2G)/g' \
        -e 's/ RAW(255)/ VARCHAR(255) FOR BIT DATA/g' \
        -e 's/ RAW(\([0-9]*\))/ CHAR(\1) FOR BIT DATA/g'
}

# Convert Character Datatypes
#
function dt_char
{
    sed -e 's/ VARCHAR2/ VARCHAR/g' \
        -e 's/ LONG/ CLOB(2G)/g' \
        -e 's/ CHAR(255)/ VARCHAR(255)/g'
}

# Convert Date Datatypes
#
function dt_date
{
    sed -e 's/ DATE/ TIMESTAMP/g'
}

# Modify the default clause
#
function dt_def
{
    sed -e 's/ DEFAULT .*,/ default/g'
}

# Add ';' to end of each create table
#
function stmt_end
{
    sed -e 's/$/;/g' | tr -d '"'
}

grep -w "CREATE TABLE " $1 | \
    pct_tab | dt_num | dt_bin | dt_char | dt_date | dt_def | stmt_end
```

Figure 41. Converting DDL for Tables

You should redirect the output of this script to a file and perform further modifications on that file, if required. Information that may need to be added to the generated file includes index and LOB tablespace information or 'NOT NULL' parameters to the primary key columns. If no modifications are to be made, the following steps should be all that is required:

```
$ exp scott/tiger rows=n file=scott.export
$ db2 connect to database_name
$ table_ddl.ksh scott.export > scott.sql
$ db2 -t -f scott.sql
```

Note the above user will need to have access to both the Oracle and the DB2 environments. The tables created will belong to the user executing the `scott.sql` file, and they will need to set up the privileges accordingly.

11.4.2 Moving Data from Oracle to DB2

In migrating the data from Oracle to DB2, we will need to export the data from Oracle into flat files. By doing this, we get the data into a format that DATABASE 2 for AIX Version 2 can recognize. These files may be delimited ASCII or non-delimited ASCII. The tools that can be used to load the information are `import` or `load`.

To extract the information from Oracle you will need to write either SQL statements or an small application that will select the rows and export this information to a file. If this is the case, there are a couple of points that need consideration, including the date and binary data types.

An example of the SQL statements to create an export file are shown in Figure 42.

```
SET NEWPAGE 0
SET TERMOUT OFF
SET SPACE 0
SET LINESIZE 78
SET PAGESIZE 0
SET ECHO OFF
SET FEEDBACK OFF
SET HEADING OFF
SET SQLPROMPT ""
SPOOL emp.dat
SELECT empno, ename, job,mgr,
       TO_CHAR(hire_date,'YYYY-MM-DD-HH24.MI.SS'),
       sal,comm,deptno
FROM emp
/
exit;
```

Figure 42. SQLPLUS Command to Export an Oracle Table (oraexp.sql)

In the previous examples, the date data type in Oracle has been replaced by the timestamp data type in DB2. However, you may only require the day, month and year to be stored in the new data type. If this is the case, then you should use the date data type in DB2. This conversion is covered in more detail in Chapter 5, "Data Types" on page 53. You may also use the `COL` command to format the file.

Binary objects may be loaded from the original data files that were used to create the Oracle object if they still exist. If these files are no longer available, a program needs to be written to extract the information from Oracle. Once this has been done, the data may be loaded into DB2 using the load command with the 'LOBSINFILE' option.

You should familiarize yourself with both the load and import utilities in DB2. before undertaking the process of exporting data from Oracle.

To load the file into DB2 using the load utility, the following command could be used.

```
db2 "import from 'emp.dat' of ASC modified by T reclen=99 \
    method L (1 10, 11 20, 21 29, ..... ) \
    insert into oracle.emp"
```

In the above example, you would need to set the reclen to the length of a table row (line in file), and the remaining columns will nee to be added into the parameter list for method L. The load command could also be used instead of the import command.

11.4.3 Referential Constraints

This step extracts the alter table statements from the Oracle export file and builds the referential constraint commands for DB2. To generate the export file, you need to use the same command as mentioned in the previous steps. If the file still exists, use it.

```
exp scott/tiger rows=n file=scott.export
```

The alter table commands that create unique indexes on the table are ignored during this phase. Indexes are created during the next step.

The sample script shown in Figure 43 will extract the appropriate alter table commands and convert them into DB2 commands.

```
#!/usr/bin/ksh

#
# Extract the users and convert into Operating system mkuser commands
#
cat $1 | grep -w "ALTER TABLE " | grep -v " ADD[ ]*UNIQUE " | \
         sed -e 's/\(^*\)USING INDEX .*/\1/g' | \
         sed -e 's/$/;/g' | tr -d ""'
```

Figure 43. Korn Shell Script to Convert Referential Constraints

Again, the output from this command should be redirected to a file so that it may be executed by DB2.

11.4.4 Index Conversion

This step extracts the Oracle commands that will create indexes on the tables. It will convert them into DB2 create index statements.

Once again, we will use the export file generated with the command:

```
exp SCOTT/TIGER rows=n file=SCOTT.dmp
```

The sample script show in Figure 44 on page 154 will extract all the Oracle create index statements as well at the alter table ... add unique statements and convert all these into DB2 create index statements.

```
#!/usr/bin/ksh

#
# Convert 'alter table' and 'create index' statements
#
cat $1 | grep -w "CREATE [A-Z ]*INDEX " | \
         sed -e 's/\(^.*\)PCTFREE.*/\1;/g' | tr -d '"'

cat $1 | grep -w " ADD[ ]*UNIQUE " | \
         sed -e 's/ALTER TABLE \(.*\) ADD [ ]*UNIQUE \(.*\) \
USING.*/CREATE UNIQUE INDEX XXXXXX ON \1 \2;/g' | tr -d '"'
```

Figure 44. Index Conversion

Once this script has executed, you can edit the output file to replace XXXXXXX in CREATE UNIQUE INDEX by an index name.

11.4.5 Synonym Conversion

As we did with the indexes, we can extract the synonym definitions from the export file for each user and convert them into DB2 statements.

The sample script shown in Figure 45 can be used to help extract and convert the statements. The output of this script should be redirected to another file so that any modifications can be made before you execute them.

```
#!/bin/ksh
FILE=$1
grep -w "CREATE [A-Z ]*SYNONYM " $FILE | \
sed -e 's/PUBLIC //g' -e 's/$/;/g' | tr -d '"'
```

Figure 45. Synonym Conversion

11.4.6 View Conversion

Unfortunately, views need to be converted manually. The SQL statements, select VIEW_NAME,':',TEXT from user_views, in Oracle will return information on what views are defined in Oracle. Also, the export file will contain the create view statements.

The script shown in Figure 46 on page 155 will extract all the create view statements from the Oracle export file and format them so that the DB2 command, db2 -t -f filename, will execute them. To do this, you need to redirect the output of the script to a file.

Attention

Be careful, The body of view can contain Oracle owner function. For more information about these functions, see Chapter 7, “SQL Language Elements” on page 71.

```
#!/usr/bin/ksh

#
# Extract the CREATE View statements from export file
function getviews
{
    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 | tr -d '"' | sed -e 's/^.select/select/g'
                read line
            done
            echo ";"
        fi
        read line
    done
}

cat $1 | getviews
```

Figure 46. View Conversion

The create view statements may need to be modified because some Oracle options are not available in DB2.

11.4.7 Grant Permission on User Objects

Many of the grant statements will work in DB2 without changing the format. However, there are some differences in granting some of the privileges. The script shown in Figure 47 will extract the statement from the export file, and you will be able to redirect the output to a file. Some modifications may need to be made to grant statements that deal with administrative permission, rather than object permission.

```
#!/bin/ksh
FILE=$1
grep "^GRANT " $FILE | tr -d '"'
```

Figure 47. User and Role Conversion

11.4.8 Procedure, Function and Trigger Conversion

Unfortunately, procedures, functions and triggers are not simple to move across to DATABASE 2 for AIX Version 2. This will be the most time-consuming part of the migration as most of the existing procedures, functions and triggers will have to be rewritten. You may find that it is possible to use constraints in DB2, rather than some trigger/procedure combinations.

Since these objects will vary differently between environments, because each is unique, the best we can offer here are pointers to where the information is stored.

You can find information about user procedure and function in the system table, user_source. This information may be extracted using the sqlplus command.

```
sqlplus SCOTT/TIGER <<END
   set head off;
   set line 256;
   select name,type,text
   from user_source;
END
```

You can find information about user triggers in the system table, user_triggers, and it may be extracted using a similar command.

Several of the chapters in this book discuss triggers, procedures and functions in the DB2 environment. Also, the topics are covered in the DATABASE 2 for AIX Version 2 literature.

Appendix A. Oracle 7 and DATABASE 2 for AIX Version 2 Limits

Table 25 describes certain maximums inherent in DATABASE 2 for AIX Version 2. Adhering to the most restrictive case can help programmers design application programs that are easily portable. For more information, see Appendix A - DATABASE 2 for AIX Version 2 SQL reference.

Table 25. DATABASE 2 for AIX Version 2 Limits

DATABASE 2 for AIX Version 2 maximums	**Limit**
Longest authorization name	8
Longest constraint name	18
Longest cursor name	18
Longest external program name	8
Longest host identifier	30
Longest schema name	8
Longest server (database alias) name	8
Longest command name	18
Longest unqualified column name	18
Longest unqualified package name	8
Longest unqualified table, view, alias, or index name	18
Most columns in a table	255
Most columns in a view	2000
Maximum length of a row including all overhead	4005 bytes
Maximum size of a table	64 GB
Maximum size of an index	64 GB
Most rows in a table	4 GB
Longest index key including all overhead	255 bytes
Most columns in an index key	16
Most indexes on a table	32 KB
Most tables referenced in a SQL statement or a view	Storage
Most host variable declarations in a precompiled program	Storage
Most host variable references in a SQL statement	1489
Longest host variable value used for insert or update	1 999 999 999
Longest SQL statement	32 KB
Most element in a select list	255
Most predicates in a WHERE or HAVING clause	storage
Maximum number of columns in a GROUP BY clause	255
Maximum total length of columns in a GROUP BY clause	4005 bytes
Maximum number of columns in a ORDER BY clause	255
Maximum total length of columns in a ORDER BY clause	4005 bytes
Maximum size of an SQLDA	64 KB
Most declare cursors in a program	storage
Maximum number of cursors opened at one time	storage
Most tables in a relational database	65535
Maximum number of prepared commands	storage
Maximum number of packages	Storage

For information about database limits in Oracle 7, see Appendix D of *The Oracle 7 Server Administration Guide*.

Appendix B. IBM SQL Reserved Words

This appendix describes the restrictions of certain names used by the database manager. IBM SQL and ISO/ANSI SQL92 include the reserved words listed below. These reserved words are not enforced by DATABASE 2 for AIX Version 2, but we suggest that you do not use them as ordinary identifiers in names that will have a continuing use.

ABSOLUTE	DAY	IDENTIFIED	OBID	SQL
ACQUIRE	DBA	IDENTITY	OCTET LENGTH	SQLCODE
ALLOCATE	DBSPACE	IGNORE	OFF	SQLERROR
ARE	DEALLOCATE	IMMEDIATE	OPEN	SQLSTATE
ASSERTION	DECLARE	INDICATOR	OPTIMIZE	STATISTICS
AT	DEFAULT	INITIALLY	OUTPUT	STOGROUP
AUDIT	DEFERRABLE	INNER	OUTER	STORPOOL
AUTHORIZATION	DEFERRED	INPUT	OVERLAPS	SUBSTRING
	DESCRIBE	INSENSITIVE		SYSTEM
BEGIN	DESCRIPTOR	INTERVAL	PAGE	SYSTEM USER
BIT LENGTH	DIAGNOSTICS	ISOLATION	PAGES	
BOTH	DISCONNECT		PART	TABLESPACE
BUFFERPOOL	DISPLACEMENT	JOIN	PARTIAL	TEMPORARY
	DOMAIN		PCTFREE	THEN
CASCADED		LABEL	PCTINDEX	TIMEZONE HOUR
CASE	EDITPROC	LANGUAGE	PLAN	TIMEZONE MINUTE
CAST	ELSE	LAST	POSITION	TRANSACTION
CATALOG	END	LEADING	PREPARE	TRANSLATION
CCSID	ERASE	LEFT	PRESERVE	TRIM
CHAR LENGTH	END-EXEC	LEVEL	PRIOR	TRUE
CHARACTER	ESCAPE	LOCAL LENGTH	PRIQTY	
	EXCEPTION	LOCKSIZE	PRIVATE	UNKNOWN
CHECK	EXEC	LOWER	PROCEDURE	UPPER
CLOSE	EXPLAIN			USAGE
CLUSTER	EXTERNAL	MATCH	READ	USING
COALESCE	EXTRACT	MINUTE	RELATIVE	
COLLATE		MODULE	RESET	VALIDPROC
COLLATION	FALSE	MONTH	RESOURCE	VALUE
COLLECTION	FIELDPROC		RIGHT	VARIABLE
CONCAT	FIRST	NAMED	ROW	VARYING
CONNECTION	FOUND	NAMES	ROWS	VCAT
CONSTRAINT	FULL	NATIONAL	RUN	VOLUMES
CONTINUE		NATURAL		
CONVERT	GET	NCHAR	SCHEDULE	WHEN
CORRESPONDING	GLOBAL	NEXT	SCHEMA	WHENEVER
CROSS	GO	NHEADER	SCROLL	WRITE
CURRENT DATE	GOTO	NULLIF	SECOND	
CURRENT TIME		NUMPARTS	SECQTY	YEAR
CURRENT TIMESTAMP	HOUR		SECTION	
CURRENT USER			SESSION_USER	ZONE
			SIZE	

Appendix C. Functions

This is a list of the Oracle functions in alphabetical order with the corresponding DATABASE 2 for AIX Version 2 function list beside it. Any listing labeled 'UDF Required' means that there is no equivalent DB2 function, and a User-Defined Function needs to be created.

Table 26 (Page 1 of 2). Oracle 7 and DATABASE 2 for AIX Version 2 Functions

Oracle	DATABASE 2 for AIX Version 2
ABS	ABS or ABSVAL
ADD_MONTHS	UDF Required
ASCII	ASCII
AVG	AVG
CEIL	CEIL or CEILING
CHARTOROWID	UDF Required
CHR	CHR
CONCAT	CONCAT
CONVERT	UDF Required
COS	COS
COSH	UDF Required
COUNT	COUNT
DUMP	UDF Required
EXP	EXP
FLOOR	FLOOR
GLB	UDF Required
GREATEST	UDF Required
GREATEST_LB	UDF Required
HEXTORAW	X
INITCAP	UDF Required
INSTR	POSSTR
INSTRB	POSSTR
LAST_DAY	UDF Required
LEAST	UDF Required
LEAST_UB	UDF Required
LENGTH	LENGTH
LENGTHB	LENGTH
LN	LN or LOG
LOG	LOG10
LOWER	LCASE
LPAD	UDF Required
LTRIM	LTRIM
LUB	UDF Required
MAX	MAX
MIN	MIN
MOD	MOD
MONTHS_BETWEEN	UDF Required

Table 26 (Page 2 of 2). Oracle 7 and DATABASE 2 for AIX Version 2 Functions

Oracle	DATABASE 2 for AIX Version 2
NEW_TIME	UDF Required
NEXT_DAY	UDF Required
NVL	NULLIF
POWER	POWER
RAWTOHEX	HEX
REPLACE	REPLACE
ROUND	ROUND
ROWIDTOCHAR	UDF Required
RPAD	UDF Required
RTRIM	RTRIM
SIGN	SIGN
SIN	SIN
SINH	UDF Required
SOUNDEX	SOUNDEX
SQRT	SQRT
STDDEV	UDF Required
SUBSTR	SUBSTR
SUBSTRB	SUBSTR
SUM	SUM
SYSDATE	CURRENT DATE
TAN	TAN
TANH	UDF Required
TO_CHAR	DIGITS or UDF
TO_DATE	DATE or CAST
TO_LABEL	UDF Required
TO_NUMBER	DECIMAL
TRANSLATE	TRANSLATE
TRUNC	TRUNC or TRUNCATE
UID	UDF Required
UPPER	UCASE
USER	USER
USERENV	UDF Required
VARIANCE	UDF Required
VSIZE	LENGTH

Appendix D. Oracle and DB2 System Catalog

This appendix lists the system catalog/data dictionary tables for both Oracle 7 and DATABASE 2 for AIX Version 2. These may be of use when trying to extract information from the database manager when performing the data conversion.

D.1 Oracle 7 Data Dictionary Tables/Views

More information about the tables/views listed in Table 27 may be found in the *Oracle 7 Server Administrator's Guide*.

Table 27 (Page 1 of 4). Oracle 7 Data Dictionary Views

Views	Description
ALL_CATALOG	All tables, views, synonyms, sequences accessible to the user
ALL_COL_COMMENTS	Comments on columns of accessible tables and views
ALL_COL_PRIVS	Grants on columns for which the user is the grantor, grantee, owner, or for which an enabled role or PUBLIC is the grantee
ALL_COL_PRIVS_MADE	Grants on columns for which the user is owner or grantor
ALL_COL_PRIVS_RECD	Grants on columns for which the user, PUBLIC or enabled role is the grantee
ALL_CONSTRAINTS	Constraint definitions on accessible tables
ALL_CONS_COLUMNS	Information about accessible columns in constraint definitions
ALL_DB_LINKS	database links accessible to the user
ALL_DEF_AUDIT_OPTS	Auditing options for newly created objects
ALL_DEPENDENCIES	Dependencies to and from objects accessible to the user
ALL_ERRORS	Current errors on stored objects that the user is allowed to create
ALL_INDEXES	Descriptions of indexes on tables accessible to the user
ALL_IND_COLUMNS	COLUMNs comprising INDEXes on accessible TABLES
ALL_OBJECTS	Objects accessible to the user
ALL_REFRESH	All the refresh groups that the user can touch
ALL_REFRESH_CHILDREN	All the objects in refresh groups, where the user can touch the group
ALL_SEQUENCES	Description of SEQUENCEs accessible to the user
ALL_SNAPSHOTS	Snapshots the user can look at
ALL_SOURCE	Current source on stored objects that user is allowed to create
ALL_SYNONYMS	All synonyms accessible to the user
ALL_TABLES	Description of tables accessible to the user
ALL_TAB_COLUMNS	Columns of all tables, views and clusters
ALL_TAB_COMMENTS	Comments on tables and views accessible to the user
ALL_TAB_PRIVS	Grants on objects for which the user is the grantor, grantee, owner, or for which an enabled role or PUBLIC is the grantee
ALL_TAB_PRIVS_MADE	User's grants and grants on user's objects
ALL_TAB_PRIVS_RECD	Grants on objects for which the user, PUBLIC or enabled role is the grantee
ALL_TRIGGERS	Triggers accessible to the current user
ALL_TRIGGER_COLS	Column usage in user's triggers or in triggers on user's tables
ALL_USERS	Information about all users of the database
ALL_VIEWS	Text of views accessible to the user

Table 27 (Page 2 of 4). Oracle 7 Data Dictionary Views	
Views	**Description**
DBA_2PC_NEIGHBORS	Information about incoming and outgoing connections for pending transactions
DBA_2PC_PENDING	Info about distributed transactions awaiting recovery
DBA_AUDIT_EXISTS	Lists audit trail entries produced by AUDIT NOT EXISTS and AUDIT EXISTS
DBA_AUDIT_OBJECT	Audit trail records for commands concerning objects, specifically: table, cluster, view, index, sequence, [public] database link, [public] synonym, procedure, trigger, rollback segment, tablespace, role, user
DBA_AUDIT_STATEMENT	Audit trail records concerning grant, revoke, audit, noaudit and alter system
DBA_AUDIT_TRAIL	All audit trail entries
DBA_CATALOG	All database tables, views, synonyms, sequences
DBA_CLUSTERS	Description of all clusters in the database
DBA_CLU_COLUMNS	Mapping of table columns to cluster columns
DBA_COL_COMMENTS	Comments on columns of all tables and views
DBA_COL_PRIVS	All grants on columns in the database
DBA_CONSTRAINTS	Constraint definitions on all tables
DBA_CONS_COLUMNS	Information about accessible columns in constraint definitions
DBA_DATA_FILES	Information about database files
DBA_DB_LINKS	All database links in the database
DBA_DEPENDENCIES	Dependencies to and from objects
DBA_ERRORS	Current errors on all stored objects in the database
DBA_EXP_FILES	Description of export files
DBA_EXP_OBJECTS	Objects that have been incrementally exported
DBA_EXP_VERSION	Version number of the last export session
DBA_EXTENTS	Extents comprising all segments in the database
DBA_FREE_SPACE	Free extents in all tablespaces
DBA_INDEXES	Description for all indexes in the database
DBA_IND_COLUMNS	COLUMNs comprising INDEXes on all TABLEs and CLUSTERs
DBA_JOBS	All jobs in the database
DBA_JOBS_RUNNING	All jobs in the database which are currently running, join v$lock and job$
DBA_OBJECTS	All objects in the database
DBA_OBJECT_SIZE	Sizes, in bytes, of various PL/SQL objects
DBA_OBJ_AUDIT_OPTS	Auditing options for all tables and views
DBA_PRIV_AUDIT_OPTS	Describes current system privileges being audited across the system and by user
DBA_PROFILES	Display all profiles and their limits
DBA_RCHILD	All the children in any refresh group. This view is not a join.
DBA_REFRESH	All the refresh groups
DBA_REFRESH_CHILDREN	All the objects in refresh groups
DBA_RGROUP	All refresh groups. This view is not a join.
DBA_ROLES	All Roles which exist in the database
DBA_ROLE_PRIVS	Roles granted to users and roles
DBA_ROLLBACK_SEGS	Description of rollback segments
DBA_SEGMENTS	Storage allocated for all database segments
DBA_SEQUENCES	Description of all SEQUENCEs in the database
DBA_SNAPSHOTS	All snapshots in the database
DBA_SNAPSHOT_LOGS	All snapshot logs in the database
DBA_SOURCE	Source of all stored objects in the database

Table 27 (Page 3 of 4). Oracle 7 Data Dictionary Views	
Views	**Description**
DBA_STMT_AUDIT_OPTS	Describes current system auditing options across the system and by user
DBA_SYNONYMS	All synonyms in the database
DBA_SYS_PRIVS	System privileges granted to users and roles
DBA_TABLES	Description of all tables in the database
DBA_TABLESPACES	Description of all tablespaces
DBA_TAB_COLUMNS	Columns of all tables, views and clusters
DBA_TAB_COMMENTS	Comments on all tables and views in the database
DBA_TAB_PRIVS	All grants on objects in the database
DBA_TRIGGERS	All triggers in the database
DBA_TRIGGER_COLS	Column usage in all triggers
DBA_TS_QUOTAS	Tablespace quotas for all users
DBA_USERS	Information about all users of the database
DBA_VIEWS	Text of all views in the database
USER_AUDIT_OBJECT	Audit trail records for commands concerning objects, specifically: table, cluster, view, index, sequence, [public] database link, [public] synonym, procedure, trigger, rollback segment, tablespace, role, user
USER_AUDIT_STATEMENT	Audit trail records concerning grant, revoke, audit, noaudit and alter system
USER_AUDIT_TRAIL	Audit trail entries relevant to the user
USER_CATALOG	Tables, Views, Synonyms and Sequences owned by the user
USER_CLUSTERS	Descriptions of user's own clusters
USER_CLU_COLUMNS	Mapping of table columns to cluster columns
USER_COL_COMMENTS	Comments on columns of user's tables and views
USER_COL_PRIVS	Grants on columns for which the user is the owner, grantor or grantee
USER_COL_PRIVS_MADE	All grants on columns of objects owned by the user
USER_COL_PRIVS_RECD	Grants on columns for which the user is the grantee
USER_CONSTRAINTS	Constraint definitions on user's own tables
USER_CONS_COLUMNS	Information about accessible columns in constraint definitions
USER_DB_LINKS	Database links owned by the user
USER_DEPENDENCIES	Dependencies to and from a users objects
USER_ERRORS	Current errors on stored objects owned by the user
USER_EXTENTS	Extents comprising segments owned by the user
USER_FREE_SPACE	Free extents in tablespaces accessible to the user
USER_INDEXES	Description of the user's own indexes
USER_IND_COLUMNS	COLUMNs comprising user's INDEXes or on user's TABLES
USER_JOBS	All jobs owned by this user
USER_OBJECTS	Objects owned by the user
USER_OBJECT_SIZE	Sizes, in bytes, of various PL/SQL objects
USER_OBJ_AUDIT_OPTS	Auditing options for user's own tables and views
USER_REFRESH	All the refresh groups
USER_REFRESH_CHILDREN	All the objects in refresh groups, where the user owns the refresh group
USER_RESOURCE_LIMITS	Display resource limit of the user
USER_ROLE_PRIVS	Roles granted to current user
USER_SEGMENTS	Storage allocated for all database segments
USER_SEQUENCES	Description of the user's own SEQUENCEs
USER_SNAPSHOTS	Snapshots the user can look at

Table 27 (Page 4 of 4). Oracle 7 Data Dictionary Views	
Views	**Description**
USER_SNAPSHOT_LOGS	All snapshot logs owned by the user
USER_SOURCE	Source of stored objects accessible to the user
USER_SYNONYMS	The user's private synonyms
USER_SYS_PRIVS	System privileges granted to current user
USER_TABLES	Description of the user's own tables
USER_TABLESPACES	Description of accessible tablespaces
USER_TAB_COLUMNS	Columns of user's tables, views and clusters
USER_TAB_COMMENTS	Comments on the tables and views owned by the user
USER_TAB_PRIVS	Grants on objects for which the user is the owner, grantor or grantee
USER_TAB_PRIVS_MADE	All grants on objects owned by the user
USER_TAB_PRIVS_RECD	Grants on objects for which the user is the grantee
USER_TRIGGERS	Triggers owned by the user
USER_TRIGGER_COLS	Column usage in user's triggers
USER_TS_QUOTAS	Tablespace quotas for the user
USER_USERS	Information about the current user
USER_VIEWS	Text of views owned by the user
AUDIT_ACTIONS	Description table for audit trail action type codes. Maps action type numbers to action type names
COLUMN_PRIVILEGES	Grants on columns for which the user is the grantor, grantee, owner, or for which an enabled role or PUBLIC is the grantee
DICTIONARY	Description of data dictionary tables and views
DICT_COLUMNS	Description of columns in data dictionary tables and views
GLOBAL_NAME	global database name

D.2 DATABASE 2 for AIX Version 2 System Catalog Views

In Table 28, the system catalog views are listed. These views should be used whenever trying to extract information from the system catalogs. To list the actual tables in the system catalogs, you may use the `db2 list tables for system` command.

Table 28 (Page 1 of 2). Catalog Views in DATABASE 2 for AIX Version 2	
Views	**Description**
SYSCAT.CHECKS	Check constraints
SYSCAT.COLCHECKS	Columns referenced by check constraints
SYSCAT.COLDIST	Detailed column statistics
SYSCAT.COLUMNS	Columns
SYSCAT.CONSTDEP	Constraint dependencies
SYSCAT.DATA TYPES	Data types
SYSCAT.DBAUTH	Authorities on the database
SYSCAT.EVENTMONITORS	Event monitor definitions
SYSCAT.EVENTS	Events currently monitored
SYSCAT.FUNCPARMS	Function parameters
SYSCAT.FUNCTIONS	User-defined functions
SYSCAT.INDEXAUTH	Index privileges

Table 28 (Page 2 of 2). Catalog Views in DATABASE 2 for AIX Version 2	
Views	**Description**
SYSCAT.INDEXES	Indexes
SYSCAT.KEYCOLUSE	Columns used in keys
SYSCAT.PACKAGEAUTH	Package privileges
SYSCAT.PACKAGEDEP	Packages dependencies
SYSCAT.PACKAGES	Packages
SYSCAT.REFERENCES	Referential constraints
SYSCAT.STATEMENTS	Commands in packages
SYSCAT.TABAUTH	Table privilleges
SYSCAT.TABCONST	Table constraints
SYSCAT.TABLES	Table privileges
SYSCAT.TABLESPACES	Tablespaces
SYSCAT.TRIGDEP	Trigger dependencies
SYSCAT.TRIGGERS	Triggers
SYSCAT.VIEWDEP	View dependencies
SYSCAT.VIEWS	Views

Appendix E. User-Defined Functions

This appendix is an example of how to create a User-Defined Function. The example is for the COSH function which is implemented in Oracle, but not in DB2.

Figure 48 is an example of a User-Defined Function that accepts a single parameter of type double and returns the hyperbolic cosine of that value.

To be able to use this function, you will need to create this code, compile it and define the function to the DB2 database.

```
#include <math.h>
#include <sqludf.h>

void SQL_API_FN COSH(
        double  *inp,          /* input value               */
        double  *out,          /* output = cosh(inp)        */
        short   *inpnull,      /* input  null indicator     */
        short   *outnull,      /* output null indicator     */
        char    sqlstate[6],   /* sqlstate                  */
        char    fname[28],     /* fully qualifed func name  */
        char    finst[19],     /* func specific name        */
        char    msgtext[71]    /* msg text buffer           */
        )
{
        /* Calculate the cosh of the input */
        *out = cosh(*inp);

        /* exit */
        return;
}
```

Figure 48. Source Code for COSH UDF (cosh.c)

An example of a makefile that could be used to compile the User Defined Function is shown in Figure 49.

```
DB2INSTANCEPATH=/home/db2v2
CFLAGS=-I$(DB2INSTANCEPATH)/sqllib/include
LIBS=-lm

CSRCS=cosh.c
COBJS=cosh.o

udfs : $(COBJS)
        xlc -o udfs $(COBJS) $(LIBS) -H512 -T512 -bE:udfs.exp -e COSH
        cp udfs /home/db2v2/sqllib/function/udfs
        db2 -t -f udfs.sql
```

Figure 49. Makefile for COSH UDF (makefile)

The export file specified by the -b flag in the makefile is shown in Figure 50 on page 170.

```
#!
COSH
```

Figure 50. Export File for COSH UDF (udfs.exp)

Finally, to define the function within the database, you would need to use the create function statement. An example of this is shown in Figure 51. The execution of this statement is performed by the sample makefile given. The statement is stored in an ASCII file. In this example, the file is called 'udfs.sql'.

```
CREATE FUNCTION COSH(DOUBLE) returns DOUBLE
  external name 'udfs'
  language c
  parameter style db2sql
  not variant
  fenced
  not null call
  no sql
  no external action
  scratchpad
  no final call;
```

Figure 51. SQL File for COSH UDF (udfs.sql)

Index

E

F

G

H

I

J

K

L

M

N

P

R

S

T

U

V

W

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
DATABASE 2 for AIX Conversion Guide
Oracle 7.1 to DB2 Version 2
August 1995

Publication No. SG24-2567-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

Name	Address
Company or Organization	
Phone No.	

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department JN9, Building 821
Internal Zip 2834
11400 BURNET ROAD
AUSTIN TX
USA 78758-3493

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.