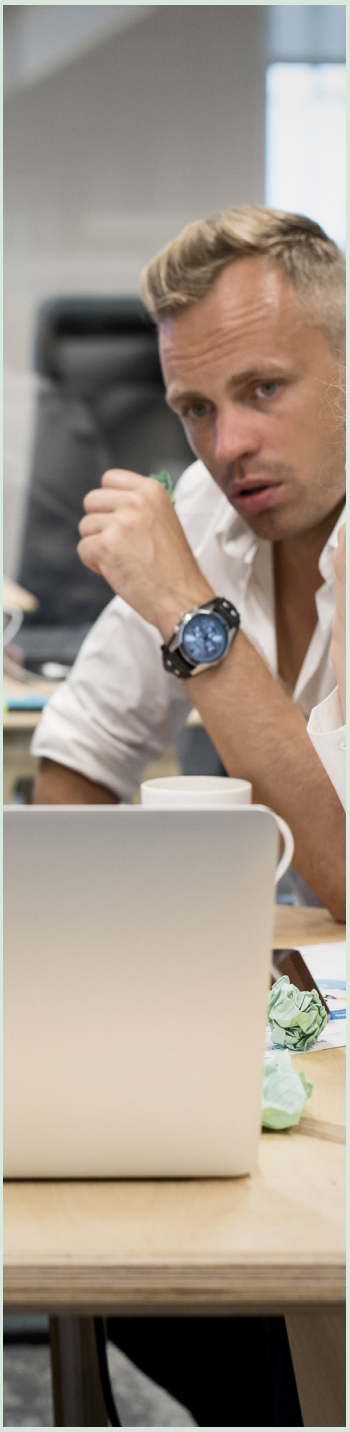
WHITE PAPER



STRATEGIES FOR DATA ARCHIVAL ON THE ORACLE DATABASE

Abstract

The financial services industry is witnessing a spurt in innovations with the advent of modern technologies, devices, and myriads of payment methods. The industry today is dealing with huge customer volumes and as a direct result, unprecedented amounts of customer data.

Inactive data or data that does not have current usage must be archived to free up space and ensure security. Data archiving is important for companies that want to manage their data growth without compromising on data quality, security, or retrievability.

This white paper describes the considerations and best practices for implementing a data archival and purge mechanism for Oracle databases. However, most of the considerations and advantages are generic and can be applied to any other domain or database.

Infosys®
Navigate your next

Table of Contents

Introduction

Decades ago, as disk space became easily available and increasingly affordable, it was possible to store millions of gigabytes of data without a second thought. Since storage space was no longer perceived as a precious commodity, housekeeping practices such as deletion of unwanted data or any maintenance activity on inactive data became rare.

Today financial institutes are dealing with data in terms of terabytes and zettabytes. In addition to burgeoning customer data, existing inactive data is now posing challenges for day-to-day operations. Systems are getting slower and batch jobs are taking much longer to process. With users are noticing the sluggishness, they are reporting issues with increasing frequency. Bogged down by unused historical data, databases are running out of space and taking longer to run critical operations. To overcome these challenges, businesses are having to invest in faster servers and more storage, all of which adds up to exorbitant costs.

Efficient and efficacious data archiving is the solution to most of these problems. Data archiving is the process of decoupling old and unused data from current data and storing it in a secure manner.

Data Retention Policies

Prior to finalizing the strategy for data retention and archival, it is important for businesses to define their data retention policies. Depending on the nature of business and regulatory requirements, the policies may vary. For example, financial institutions have legal requirements to retain data securely for several years.

Key aspects of data retention policies

- Data retention policies are provided by clients in conjunction with their legal departments
- The policy must clearly call out the duration of data retention
- Development teams can decide the retention period for internally used tables, log files, or audits that are not directly referred in client operations
- An effective data retention policy meets legal, regulatory, and business requirements
- If there are technical constraints, the business and technology teams must collaborate and take the final call
- Data retention policies can be applied to both archiving and purging

Sample data retention policy

Figure 1 illustrates various aspects to be considered by a business while defining the data retention policy.

Retention policy: OLRP120

Policy title: Bank cards/credit cards - credit/loan accounts

Retention requirement: Active + 7 Years

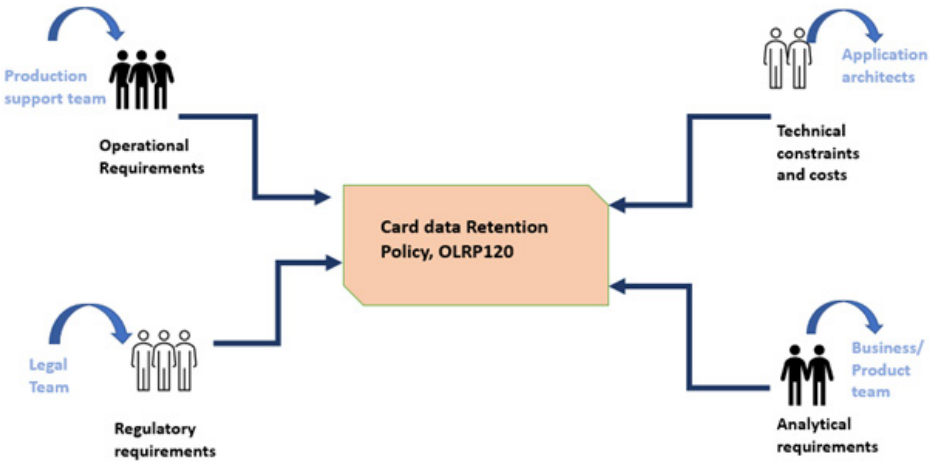


Figure 1: Key aspects of data retention policy

Common Challenges in Data Storage and Retrieval

A well-implemented data archival mechanism can address many common data storage and retrieval problems faced by financial systems.

Problem	Solution
Financial systems are an excessively regulated industry. However, many businesses accidentally delete or purge data that they should archive as per regulatory requirements.	Implementing an archival solution is a great first step. Data archival and retrieval solutions automatically help prepare responses to any compliance audit or information request for business communication data.
Content explosion causes servers to slow down or go offline resulting in productivity problems.	Data archiving is the best strategy to deal with such a situation. Archival improves system performance and reduces downtime by deleting data from the live system after archival.
Huge amounts of data can keep team members busy searching for or recovering historic data as per business needs. This takes away time from other productive business-critical activities.	A good data archiving solution empowers users to access the archive and quickly retrieve necessary information instead of spending countless man hours recovering data.
Accidental data leakage can occur if data is not archived securely.	A streamlined archival process ensures that data does not leak under any circumstances.
If data is not archived securely, there could be security breaches leading to unauthorized data access.	By securely archiving data, businesses can track information and increase protection to prevent unauthorized access.

Advantages of Early Implementation of a Data Archival System

It is best to plan data archival as part of the initial implementation design. At this stage, there is visibility into table categorization and the impact of data archival on each table is well understood. If data retention policies are not defined at this stage, we can assume (after consulting with product/ business team) about 7-8 years of data retention depending on the business domain and implement data archival. Benefits of implementing data archival in the initial stages of the project cycle include:

- Lower testing efforts compared to later in the cycle
- Prevention of issues such as sluggish systems and running out of space
- If archival model is implemented early in the project, we can avoid one-time data deletion which is expensive and could have business impact

Considerations for Implementing Data Archival

Two major considerations for data archival are:

1. Categorization of the tables
2. Deciding the archival or purge frequency

Table categorization

Database tables are at the core of data archival. Tables can be classified into three types:

Base or metadata tables Config tables which store master information or data about base tables i.e., Metadata. These tables are not considered in archival or purge operations as these are master tables. For example, country master, currency master, configuration master.

e.g., Country and Currency are two base tables, but together currency of a particular country is Metadata.

Unused tables are created for temporary usage such as for data correction, data verification, and temporary data back-up. Such tables are not needed in the long run and must be dropped periodically by developers as part of good coding practices.

Transactional tables are the main tables considered for data archival. They store transaction related data and tend to grow significantly over time. Examples include tables that store account, customer, and transaction related data.

e.g., Account, Customer, transactions related data

Transactional tables can further be classified into two categories.

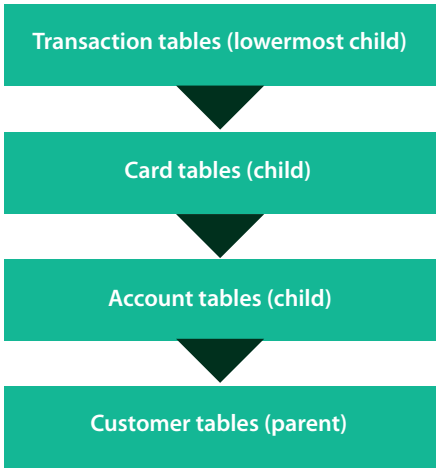
Tables with referential integrity

Deletion of data in tables with referential integrity must follow a bottom-up approach. The lowermost child must be deleted first. Then its immediate parent is deleted. Then the next parent and so on. Deleting data in a different order will break this referential integrity and cause problems with the data, and it will give foreign key errors and won't allow data deletion

Consider this example. A customer has an account, the account has cards, and the cards have transactions against them. The sequence of deletion of tables is shown below.

e.g., Customer has Account, Account has cards, and Cards have transactions.

Sequence of deletion will be as below for this example.



Tables not having referential integrity

Audit and batch tables may follow in this category.

Audit and batch tables must be purged on a regular basis as data older than 6 months in these tables is typically not required. Such tables do not have any referential integrity and can be deleted based on dates. Examples include batch execution data and log tables. This is the set of non-referential integrity.

e.g., some batch execution data or log tables. These can be deleted flat on basis of dates.

Archival or purge frequency

Based on analysis of data such as how fast a table is populated, we need to decide whether a table must be purged or archived on a daily/weekly/monthly or yearly basis. Regular archival or purge processes can be dynamic so that new tables can be added easily using a configurable set-up. Table 1 illustrates a configuration table.

GROUP_NAME=>GROUP_ SEQUENCE=>TABLE_NAME=>TABLE_ SEQUENCE=>TABLE_WHERE_ CLAUSE=>DATA_PURGE_INTERVAL

Column name	Description
GroupName	Group of tables to which the table to be deleted belongs. For tables with referential integrity, it will be the parent table. For tables with no referential integrity, it can be any table
TableName	The name of the table that needs to be purged
GroupSequenceNo	Sequence number in which the group will be purged (as per the referential integrity sequence)
TableSequenceNo	Table Sequence No inside group, on which table will be purged. In Oracle tables have primary foreign key relationships. To delete data from parent tables, data from al child tables need to be deleted first. Tables need to be deleted in a particular sequence hence sequence of table is needed.
PurgeFrequency	The frequency at which the table is purged - daily/weekly/ monthly/yearly.

Table 1: Master configuration table

A single procedure will then purge or archive data using a dynamic delete clause based on the entries in the configuration tables.

The same approach can be used for one-time data deletion. Additionally, we need to provide a start date, end date, number of accounts, and the conditions to limit the rows for deletion.

Best Practices for Data Archival or Purging

Before arriving at the strategy for data archival, here are a few questions we need to answer. The answers will help formulate the best approach.

1. Which version of Oracle database are you running?
2. Are you running the standard edition or enterprise edition?
3. Should the system be running while data is being deleted or is downtime ok?
4. What is the duration available to execute the purge?
5. What is the percentage of data to be deleted?
6. How will the replication (Golden gate) across data centers be handled?

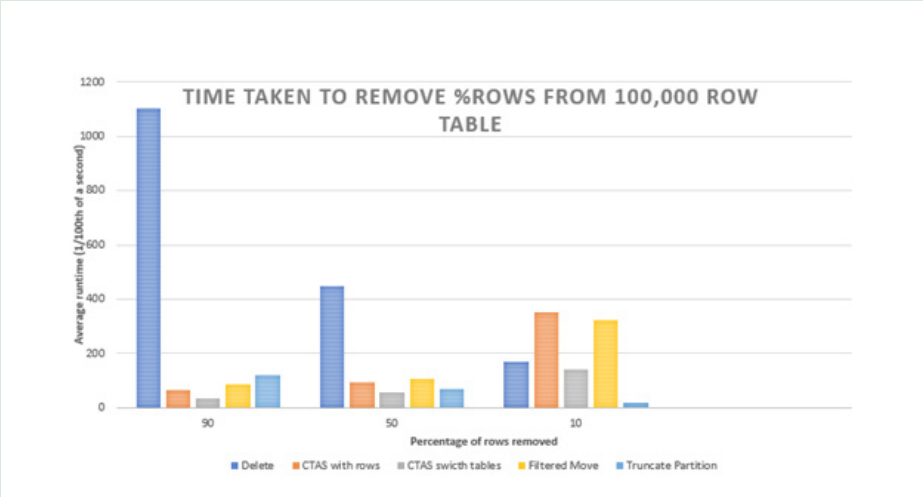


Figure 2 – Time taken to remove % rows from a 100,000-row table

Let us now look at different techniques to purge or archive data based on specific business requirements. Figure 2 illustrates the various methods to remove data based on the scenario.

Bulk deletion

Some businesses, such as banking systems, cannot afford any downtime in their systems. In such cases, bulk deletion is the best option. Bulk delete is useful when you cannot delete in a single SQL statement due to huge volume of data deletion that will generate several undo/redo logs.

Use parallel hint to make deletion faster. The degree of parallelism can be 8/16/32 based on database configurations and is best suggested by your DBA.

Remove rows with CTAS

Remarks: CTAS is not recommended for critical applications as this requires downtime.

Use the create table as select (CTAS) approach if the data to be purged or archived is much greater than the data to be retained. For example, you want to archive closed accounts and in the entire system there are 80% closed accounts and only 20% active accounts. Instead of deleting 80% data using bulk delete, it is easier to insert 20% data into new tables. In such situations, bulk delete is not an option because of the risk of system downtime.

1. Create a new table saving the rows you want to retain
2. Truncate the original table
3. Load the saved rows back into the original table with insert as select

To load the required rows into the main table, use "INSERT INTO TABLE SELECT" or "BULK INSERT" or IMP/IMPDP. Import works faster than bulk inserts. However, for import activities root access/DBA interventions are required.

CTAS with rename table

Remarks: CTAS is not recommended for critical applications as this requires downtime.

This process is like removing tables with CTAS and can be used when the data to be archived is greater than data to be retained.

1. Create a new table saving the rows to be retained
2. Drop or rename the original table
3. Rename the new table to the original
4. Create metadata on table (indexes/ constraints/triggers)

CTAS does not copy the metadata. Care must be taken to check that metadata is intact after the activity.

Renaming a table could cause dependent objects to become invalid and may need to be recompiled.

Table partition

When you partition a table, you logically split it into many sub-tables. You can then run operations which only affect rows in a single partition. This provides a faster way to remove all the rows in a particular partition. Drop or truncate the partition!

For example, if you partition the table on insert date, you can quickly wipe out data in the oldest partition.

Delete data with a filtered table move

Oracle DB version required for this option is 12c Release 2 (12.2).

The advantage of this option is that it works while the application is running. However, it can fail with an ORA-02266 if there are enabled foreign keys pointing to the table. Careful planning is needed for use on parent tables and determining the sequence of operation to prevent failure.

With a filtered table move, you can migrate a subset of the data. This only moves the rows that match the WHERE clause. Provided you want to remove a large chunk of data, this approach is better than delete.

Exchange partition

On the Oracle database, EXCHANGE PARTITION can drastically improve data loading time while loading a relatively small amount of data into a target table that contains a much larger volume of historical data. The target can be any partitioned table – a dimension, a fact, or any other table.

Let us assume you have a source table with the new data that needs to be loaded into a partitioned table. Use the exchange partition mechanism to move data from the source table into the target partitioned table without actually moving any data. It is a data dictionary update. Exchange partition is an operation with no data movement and therefore the gain in performance is significant. In addition, the biggest advantage is that, unlike delete/insert processes, the table remains available for performing queries.

Benefits of Archiving Data

Beyond any doubt, implementing a robust data archival mechanism has several key benefits.

- Improved performance due to accelerated processing time, decreased backup and restore periods, reduced consumption of disk space, and faster lookups
- Efficient data management due to reduced data. Data duplication takes up an average of 25% of data allowance on software systems. An archiving solution allows you to identify this redundant data and remove it
- Reduced back-up window due to low volume of data to back-up. Databases with no archival mechanisms in place may take upwards of 18 hours for a single back-up
- Prevention of accidental deletion by defining policies to store deleted data for a specified period even after deletion. This helps recover the data if the deletion was not intentional

Conclusion

Businesses today need to be agile and keep up with customer expectations. To serve required content faster to customers, the underlying systems need to be fast and efficient. One of the common reasons for sluggishness of systems is lack of maintenance of underlying data. Poor housekeeping of data as well as lack of data purging and archival processes can impose costs in terms of poor customer experience and regulatory risks.

It is imperative for organizations to implement data archival policies early in their development cycle. Data archival strategies can be complex due to the interrelationships between cost and business impact of data deletion and regulatory and legal requirements of data retention.

Each business must collaborate with key stakeholders such as the legal department, top executives, and technology experts to arrive at a strategy that works best for their organization. Finally, the solution implemented must be flexible to allow for changes in the future.

Case Studies

Case study 1

Compliance and regulatory project requirement for a prominent US-based cards domain client

The client had stored data for over 15 years. The client wanted data of closed accounts older than 5 years to be purged.

The solution involved implementing a configurable data archival/purge model, that would be driven by a single master configuration table and a single procedure. A configurable master configuration table helps in easy maintenance including addition or removal of new tables in the module.

We adopted the bulk deletion technique as the business could not afford any downtime. Being in the cards domain, systems need to be available 24/7. The entire activity was based on accounts that had to be purged.

To improve the speed of bulk deletion, we tuned the underlying SELECT statement for optimum performance. For huge tables, we limited the data being processed based on dates and number of records.

Case study 2

Upgrading banking/telecom collection modules from version 4.7 to 4.10

A leading bank was in the process of tying up for business with some other banks. This meant that the customer volume was going to increase drastically. This was taken into consideration in the design for the archival module in the requirements analysis phase. A mix of methodologies was adopted based on different types of tables in the system.

Some of the processes adopted included:

- Partitioning of high-volume tables to make it easy to purge records
- Partitions were dropped where possible such as in audit tables
- Bulk deletion for certain situations

About the Authors

Priti Joglekar
Technology Lead, Banking and Financial Services, Infosys

Priti has about 11 years of IT experience. She is currently working with prominent clients in the banking sector. She has a wide range of experience in the telecom and financial domains.

Smita Pol
Technology Lead, Banking and Financial Services, Infosys

Smita has about 11 years of IT experience. She is currently working with top clients in the cards and payments domain. She is involved in technical architecture and development of software for large global clients in the telecom and financial domains.

References

Importance of Data Purging

How to Delete Millions of Rows Fast with SQL (oracle.com)

Regular Delete vs. Bulk Delete - Ask TOM (oracle.com)

For more information, contact askus@infosys.com

Infosys®
Navigate your next