WHITE PAPER

ORACLE DATA MIGRATION

A COMPARATIVE STUDY

Nagesh Mittal, Ravi Shankar Anupindi, Kalidhasan Velumani

Abstract

Data Migration is an important activity in almost every organization – arising from the constant endeavor to improve data storage and retrieval processes. To ensure that data is migrated with minimal effort, it is beneficial to use proven solutions and methods. Oracle provides various features, utilities, and solutions which can cater to different needs in data migration. By comparing these options, users will be able to make a justified decision on the solution to be implemented for the data migration.

Infosys®
Navigate your next

Overview

Data migration is generally seen as an overhead, as part of the system setup, and hence the effort and approach used for data migration is expected to be simple, which involves lesser effort and proven solutions, providing source-to-destination data mapping and validation. So very often teams working on data migration would either end up using high-end ETL tools like Informatica, SAP BO, SAS, and Oracle Warehouse Builder (OWB) which can be an overkill for the project need, or they end up writing custom PL/SQL scripts to perform the migrations which can be time consuming as well as error prone, without exploring out-of-the-box features available with the database.

The intent of this article is to provide an overview on different data migration options which are provided by Oracle, along with a comparison chart on some key parameters. The comparison of different approaches will help in evaluating and deciding the most suitable data migration approach as per the project need. We are sure this article will be useful as a handbook guide for technocrats involved in data migration, but it is not a one-stop solution for all data migration needs. If your data migration demands advanced migration capabilities, which cannot be addressed by these options, then it is recommended to explore advanced ETL tools available in the market.

"MIGRATIONS REPRESENT 60% OF ALL LARGE ENTERPRISE IT PROJECTS, AND ONLY 60% ARE COMPLETED ON TIME."

IDC Report[1]

Data Migration Process

At a high level, the data migration process involves the following steps:

1) **Scope identification** – Identify the data objects and data, for each object, that need to be migrated.
2) **Data mapping** – Map the data from source to target objects. In case source and target have different data models, transformation and mapping would be essential for migration.
3) **Option selection** – Identify the migration option suitable, as per system needs, such as the time taken and target DB, as well as data needs like transformation and volume.
4) **Migration** – Perform data migration to the destination system using the selected solution.
5) **Validation** – Perform audits, validations, and acceptance tests to validate and certify data at destination.

Oracle's Out-of-the-Box Offerings

Oracle provides various out-of-the-box solutions and utilities which can be used for data migration. These utilities and tools will provide the **added advantage of easy ORACLE INTEGRATION, PERFORMANCE OPTIMIZATION, AND LEVERAGE OF ORACLE EXPERTISE** in data handling and transformation. Many of the utilities are available as part of Oracle DB installation and support migration for both Oracle and non-Oracle databases.

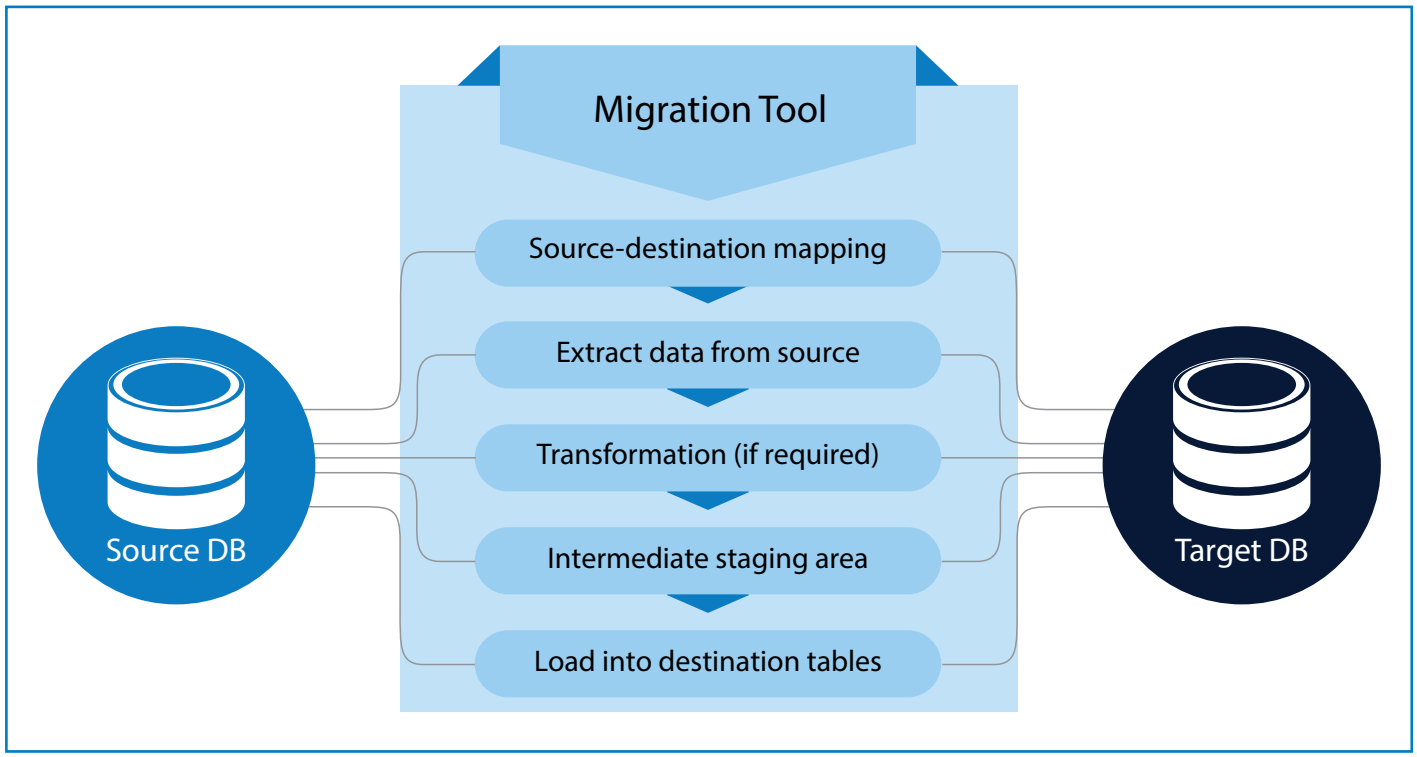


Fig 1. Data Migration Flow

As steps 2, 3, and 4 are crucial for migration and require proper evaluation, this article mainly focuses on them and covers various migration options provided by Oracle.

Data Migration Approach

Data migration is essentially the movement of data from one system to another. Oracle offers a rich set of tools and utilities for performing data movement across systems. Other than data migration, data replication and data backup/recovery solutions also fall under the umbrella of data migration solutions, since these also provide the capability of copying data from one system to another. ***In this document we have covered data migration tools as well as data replication and data backup/restore options available from Oracle.***

The table below represents the different options with support across different Oracle DB versions.

#	Migration Tools/Options	8i	9i	10g	11g	12c
Data Migration Tools						
1	Oracle Data Pump	✗	✗	✓	✓	✓
2	Transportable Tablespaces	✓	✓	✓	✓	✓
3	Copy Over DB Link	✓	✓	✓	✓	✓
4	SQL*Loader Utility	✓	✓	✓	✓	✓
5	SQL Developer	✗	✗	✓	✓	✓
6	SQL*Plus COPY Command	✓	✓	✓	✓	✓
7	Custom PL/SQL Procedures	✓	✓	✓	✓	✓
Data Replication Tools						
8	Oracle Golden Gate	✗	✗	✗	✓	✓
9	Oracle Streams	✗	✓	✓	✓	✓ *Not Recommended*
Data BackUp and Restore Tools						
10	Oracle Recovery Manager (RMAN)	✗	✓	✓	✓	✓
11	Oracle Active Data Guard	✗	✗	✗	✓	✓

Fig 2. Oracle Migration tools/options

The above mentioned tools are useful to perform data migration from Oracle to Oracle as well as to/from non-Oracle databases, details of which are explained further in this document.

With the advent of **Big Data,** there is an increasing demand for migration to NoSQL databases. There are different tools available in the market, which can be used to migrate from Oracle to NoSQL, some of which are discussed later in the document.

1. Oracle Data Pump

Oracle Data Pump utility enables very high-speed movement of data and metadata from one Oracle DB to another. It is available from Oracle Database 10g release 1 (10.1) and above. The below diagram shows the migration architecture using Oracle Data Pump.

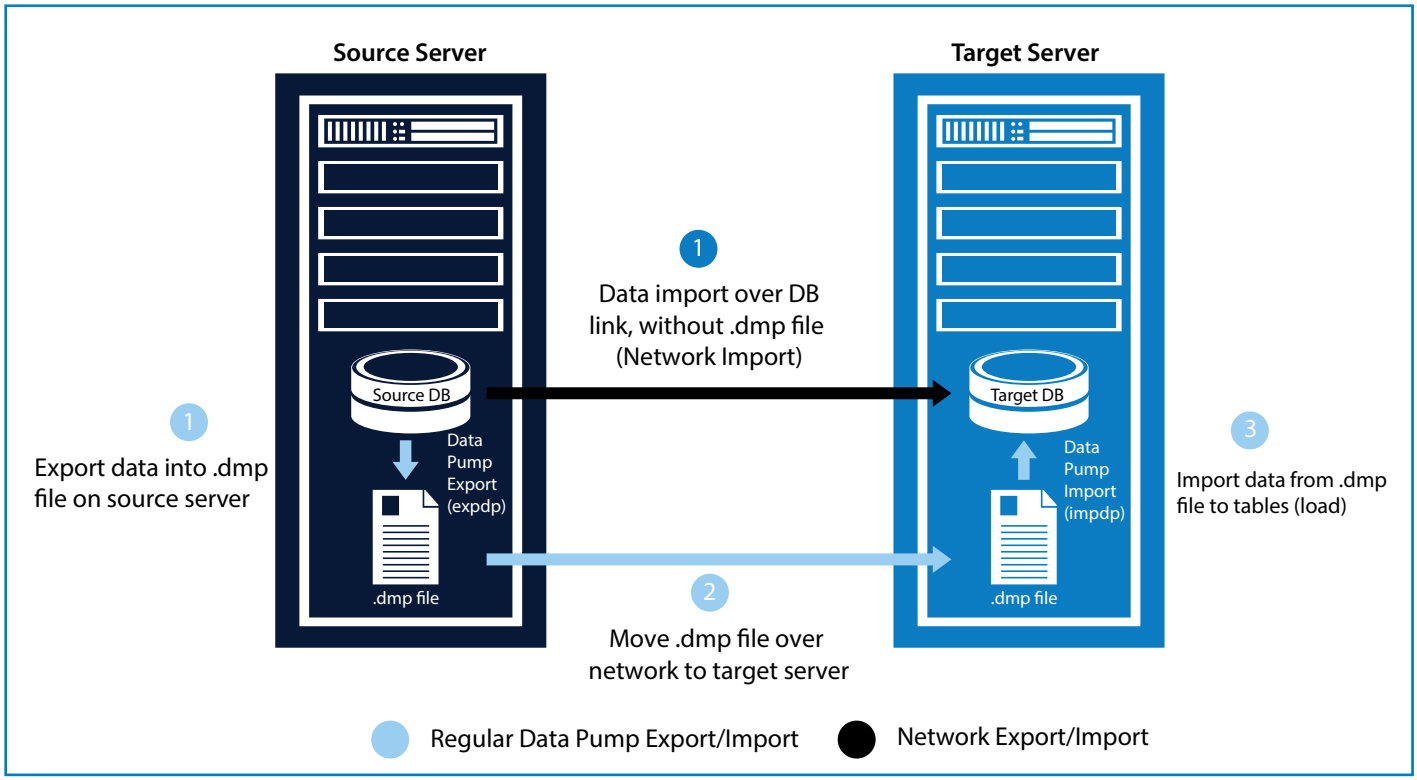


Fig 3. Oracle Data Pump Migration

Characteristics

- Enables very high-speed movement of bulk data and metadata
- Enhanced version of original/legacy import (imp) and export (exp) utilities
- Simplest approach to export objects from source to target
- Performed as a server side job and hence more efficient (unlike legacy exp/imp where dump file is created at the client location)
- Expdp and impdp are the command line utilities for Data Pump

Following are the new capabilities available with Oracle Data Pump:

#	Features	Description
1	**Tuning**	Automatically tunes the export and import processes. Tuning parameters like BUFFER, COMMIT, COMPRESS, DIRECT and RECORDLENGTH are not required
2	**Parallelism**	Being a server side job, parallel processing can be done to make data export and import more efficient
3	**REMAP**	Tables can be imported into a different tablespace from the source database

This approach is a good candidate for **performing one-time data migration** of complete schema, or selective tables (as needed), for new instance or test/build environment setup.

Supported Interfaces

This section throws light on different interfaces available for Oracle Data Pump operation.

1. Network Export/Import

- In this approach, client dependency on dump file creation and transfer are avoided by creating DB link directly from the target to the source.

```
expdp username/password@dbname tables=EMP network_link=<dblink name> directory=TEST_DIR dumpfile=EMP.dmp logfile=expEMP.log

impdp username/password@dbname tables=EMP network_link=<dblink name> directory=TEST_DIR logfile=impEMP.log
```

Note: No dump file required for network import

- To enable secured data transfer for sensitive information, the following encryption parameters are supported by Oracle Data Pump:
  - ENCRYPTION
  - ENCRYPTION_MODE
  - ENCRYPTION_PASSWORD
  - ENCRYPTION_ALGORITHM (AES128, AES192, AES256)

2. Interface with PL/SQL

Oracle provides *DBMS_DATAPUMP* built-in package to invoke Data Pump API(s). These API(s) should be invoked with the specific export mode based on the migration requirement.

#	Supported Modes	Description
1	**FULL**	Do the complete source migration
2	**SCHEMAS**	Migrate only the schemas
3	**TABLES**	Migrate only the selected tables
4	**TABLESPACES**	Migrate all tables contained in the tablespace
5	**TRANSPORT_TABLESPACES**	Migrate the metadata for tables in selected (transportable) tablespaces

Illustrative PL/SQL code invoking Data Pump API with export mode set to FULL

```
DECLARE
        dpump_handler  NUMBER;
BEGIN
        dpump_handler:= DBMS_DATAPUMP.OPEN (
                        operation => 'EXPORT',
                        job_mode => 'FULL',
                        job_name=> 'FULLEXPJOB',
                        version => 'COMPATIBLE');
```

```
                    DBMS_DATAPUMP.ADD_File (
                            handle    => dpump_handler,
                            filename => 'expdp_via_plsql.dmp',
                            directory=> 'DUMP_DIR',
                            filetype    =>1);
                    DBMS_DATAPUMP.ADD_File (
                            handle    => dpump_handler,
                            filename => 'expdp_via_plsql.log',
                            directory=> 'DUMP_DIR',
                            filetype    => 3);
DBMS_DATAPUMP.Start_Job (dpump_handler);
END;/
```

3. Oracle (external) tables

Oracle provides external table to support data dump utility. ORACLE_DATAPUMP access driver can be used to unload data to data pump export file(s) and subsequently reload it though the standard SQL select statement.

```
/* to unload data from table (emp) into an external file (emp.dmp) */
CREATE TABLE tbl_ext_data_pump
        ORGANIZATION EXTERNAL
(
                TYPE ORACLE_DATAPUMP
                DEFAULT DIRECTORY test_dir
                LOCATION('emp.dmp')
        )
        AS SELECT * FROM emp;
/* to see the data of the dump file through external table (tbl_ext_data_pump) */
SELECT * FROM tbl_ext_data_pump;
```

2. Transportable Tablespaces

Tablespace is the logical storage unit, where data is stored in one or multiple physical data files associated with the corresponding tablespace. User data is associated to different tablespaces from system data (SYS, SYSTEM). The following diagram illustrates steps involved in migration using transportable tablespaces:

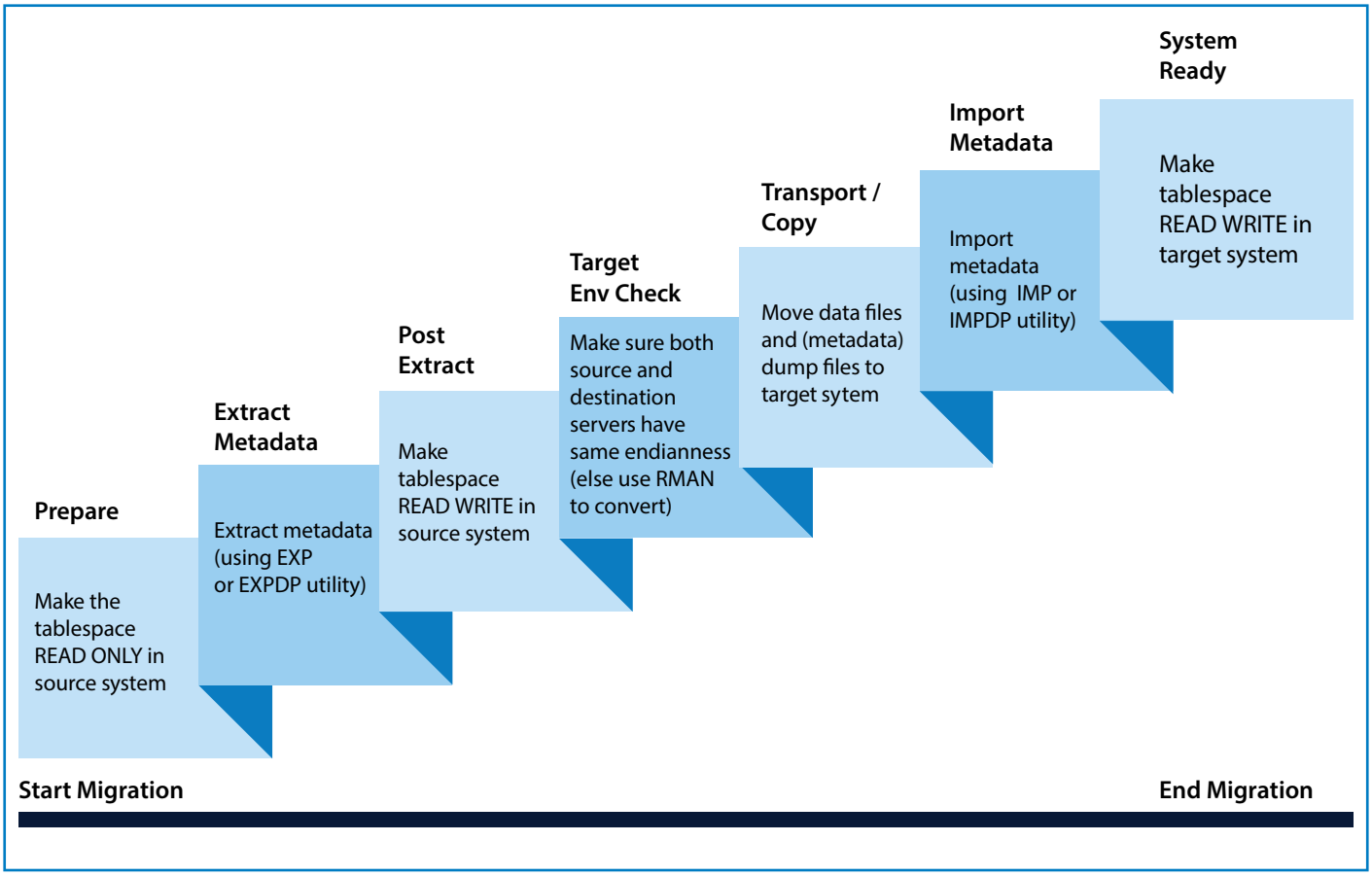


Fig 4. Transportable Tablespaces Migration

Transportable tablespaces provide an option to **migrate the complete user tablespace** across instances. This is **faster than the data pump** as only metadata is transferred to destination using data export/import, and the actual data files are just copied to destination database server.

Characteristics

- ✓ Option of copying the read-only tablespace to multiple databases
- ✓ Metadata migration can be invoked from **Data pump** using **Transportable Tablespace** mode
- ✓ Can be invoked from Backup with **RMAN and tablespace is not required to be put in read only mode**
- ✓ From 10g, it is possible to move the transportable tablespaces **across different platforms or operating systems,** provided both platforms have the same **endian format**. Supported platforms and endian format can be checked using V$TRANSPORTABLE_PLATFORM view
- ✓ Enables data archival by transporting the old tablespace to archival system
- ✓ Can be invoked through data pump utility also
- ✓ Not required to be of the same block size between source and target database (from Oracle 9i onwards)

For a tablespace to be transportable, it must be totally self-contained. This can be validated by using DBMS_TTS.TRANSPORT_SET_CHECK procedure as mentioned below.

TS_LIST is the list of tablespace(s) to be checked, and INCL_CONSTRAINTS specifies whether referential integrity checks should be included or not.

```
EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK (
                                        TS_LIST => 'TEST_DATA',
                                        INCL_CONSTRAINTS => TRUE);
```

After executing this API, any violations can be checked using the view TRANSPORT_SET_VIOLATIONS. If there are no violations for transportable tablespace check, this view will be empty.

Constraints / Limitations

- Objects and their dependencies (materialized views or partitions) should be completely contained within the same tablespace
- Target database should not have same tablespace name that is already existing
- Tablespace needs to be placed in read-only mode until migration is complete (unlike RMAN option)
- SYSTEM tablespace and objects owned by SYS user (e.g., Java Classes, views, synonyms, privileges, sequences) cannot be transported
- Materialized views and FBI (Function-based indexes) cannot be transported

If any of your migration needs falls within the above limitations, we recommend performing a proof of concept to confirm the same

Transport Tablespace using RMAN

In the standard transportable tablespace method, first put the tablespaces in READ-ONLY mode and then export the Schema Metadata (using exp/expdb utility) and the corresponding data files. So, no transaction is allowed on the system at that time as tablespace is in READ-ONLY mode.

To overcome this limitation RMAN procedure can be used which creates a temporary clone database, thereby eliminating the need to put the tablespace in READ-ONLY mode.

Exporting through RMAN

```
RMAN>  TRANSPORT TABLESPACE DATA
       TABLESPACE DESTINATION '<<location name>>'
       AUXILIARY DESTINATION '<<location name>>'
       DATAPUMP DIRECTORY TTS_METADATA_DIR;
```

Importing through RMAN

```
SQL> alter database datafile '<<data file name>>' offline;
RMAN> switch datafile '<<data file name>>' to copy;
RMAN> recover datafile '<<data file name>>';
SQL> alter database datafile '<<data file name>>' online;
```

3. Copy over DB link

A DB (database) link is a pointer that defines a one-way communication channel from an Oracle database to another Oracle database.

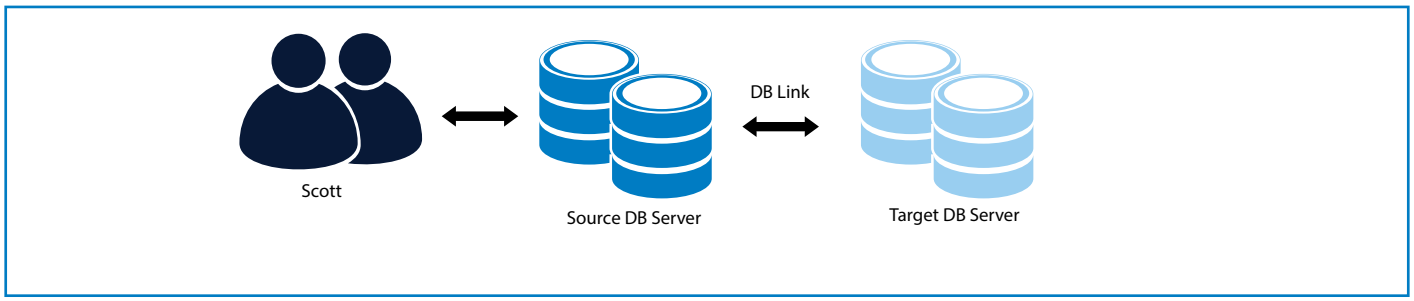


Fig 5. Data Migration Using DB Link

Database link can be created using CREATE DATABASE LINK command, and specifying the remote database along with username and password to connect.

```
/* create database link to connect to "targetdb" through user "scott" */

CREATE DATABASE LINK targetdblink
                    CONNECT TO scott IDENTIFIED BY <pwd>
                    USING 'targetdb';
```

Characteristics

✓ Very simple, direct, and most widely used approach for copying data across DBs

✓ Data can be migrated into target tables by copying the data from source over DB link using one of the following SQL options:

- CREATE TABLE AS SELECT **(CTAS)** statement

```
/* create a new table in the target database by selecting over DB Link from source */
CREATE TABLE t_local AS
SELECT col 1, col 2, col 3, … col n FROM t_source@targetdblink
WHERE type = 'ABC';
```

- INSERT INTO TABLE SELECT **(IITS)** statement

```
/* insert data from source table into an existing table in the target database */
INSERT INTO t_local (col 1, col 2, col 3, col n)
SELECT col 1, col 2, col 3, col n FROM t_source@targetdblink
WHERE type = 'ABC';
```

- Materialized Views **(MView)**

  Materialized Views are snapshots of the data created using queries on one or more remote tables through DB link. Based on the select query, data from remote tables is accessed over DB link and copied to local database. Refresh mode and interval can be specified in MView definition.

```
/* create a materialized view using remote tables over db link */
CREATE MATERIALIZED VIEW local_mview
REFRESH FAST
START WITH SYSDATE
NEXT SYSDATE+1
AS SELECT col 1, col 2, col 3, col n FROM t_source@targetdblink
WHERE type = 'ABC';
```

These commands provide the **flexibility to perform selective data copy,** by specifying the filter through WHERE clause in the SQL statements.

Constraint

- Both the source and target database servers should be available in the same network (both the systems should be able to ping each other).

Considerations

A few considerations while evaluating this approach for data movement:

- **Can lead to network overhead** when migrating huge amounts of data
- **Partitions cannot be selected** for copy over DB link. Instead WHERE clause should be specified to do partition pruning and copy data at partition level.
- Data structure of source and target databases can be different as this approach uses only Insert/Update/Select statements that provides **more flexibility** during migration
- **Security** can be a cause of concern as data is transferred over the wire
- Remote objects cannot be analyzed and also granted privileges via DB link

This approach can be used for both one-time as well as incremental data migration

4. SQL*Loader Utility

SQL*Loader is the primary method to **load data from external flat files** (ASCII/TEXT/CSV) **into tables** of an Oracle Database.

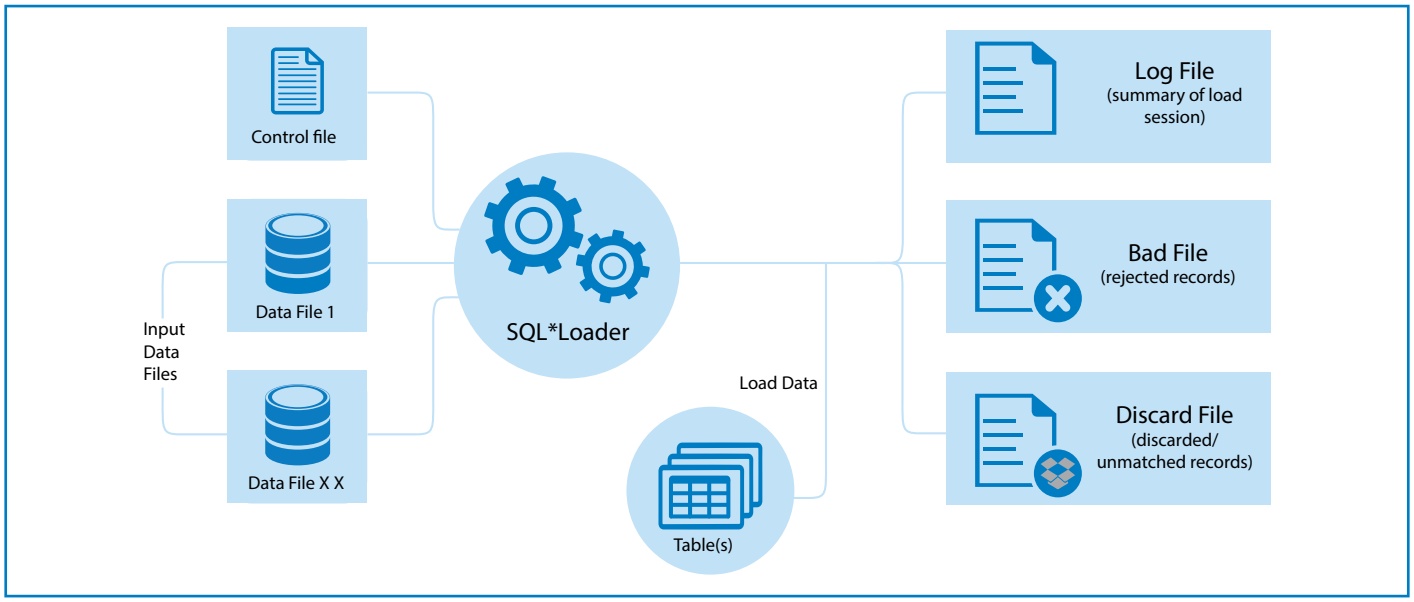


Fig 6. Data Migration Using SQL*Loader Utility

SQL*Loader works based on a control file, which specifies the location of data, how data should be parsed and interpreted, and target table where data should be loaded. It also takes inputs from one or more datafiles, which contain data to be loaded into the database.

Output from SQL*Loader for a load session is a table where data is loaded, a log file, and a bad file (with details of bad formatted records) or discard file (with details of rejected records).

Characteristics

- ✓ Can load data from multiple datafiles and into multiple tables during same load session
- ✓ Can perform remote data loading
- ✓ Selective data loading (load records based on record values)
- ✓ Supports complex object relational data
- ✓ Supports both conventional and direct path loading
- ✓ Supports loading from non-Oracle database

Loading data from Non-Oracle database

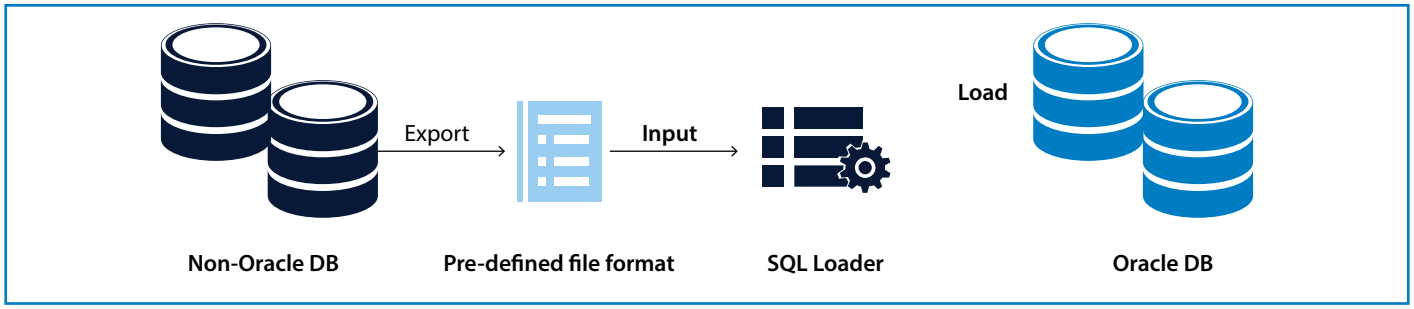


Fig 7. Loading data from Non-Oracle database

- Non-Oracle database will export the data into a pre-defined file format
- SQL loader will read this file and load the data into Oracle database tables

*Illustrative Code Sample to load data using SQL*Loader*

```
/*create data file (test_data.data) and all the column data should be delimited by comma */
10001,"Scott Tiger", 1000, 40
10002,"Frank Naude", 500, 20

/*create control file (test_loader.ctl) that has the mapping of input data file and the target table in database */
load data infile 'test_data.dat' into table emp_data
        fields terminated by "," optionally enclosed by '"'
        (Empno, empname, sal, deptno)

/* use sqlldr utility to load the data from the file, test_data.dat to the database table emp, using control file test_loader.ctl*/
sqlldr username@server/password control=test_loader.ctl
```

5. SQL Developer

SQL Developer is the GUI-based free tool offered by Oracle as Database IDE. Oracle SQL Developer allows database users and administrators to do their database tasks though UI without writing any code and is supported from Oracle 10g onwards and runs on any OS that supports Java.

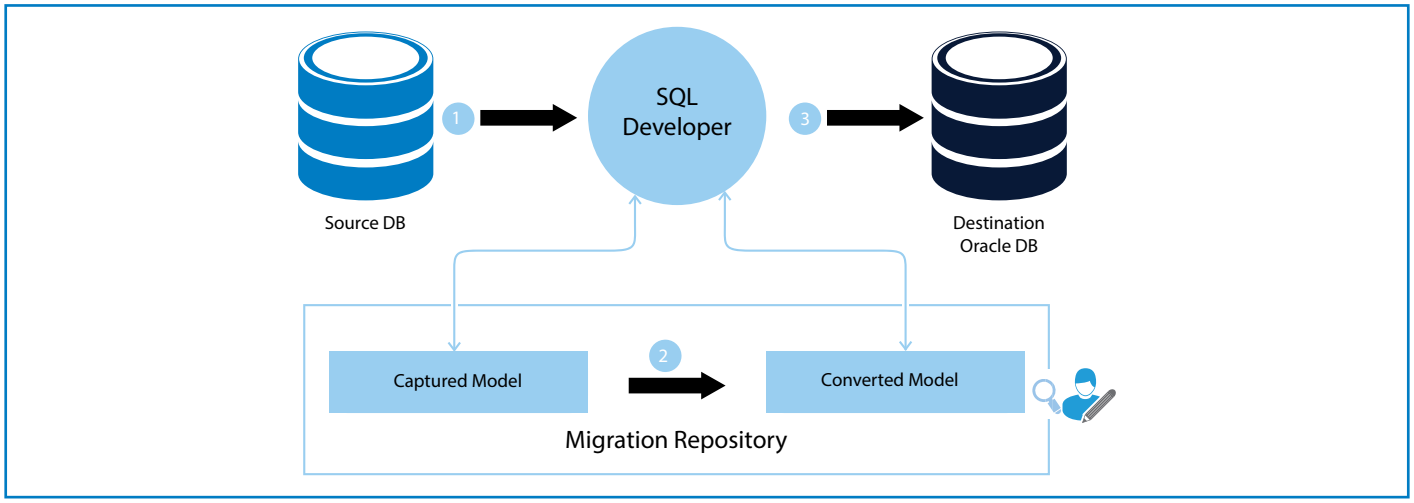


Fig 8. SQL Developer Migration Architecture

Migration Steps

1. Source DB structure to be migrated is displayed in the **captured model & stored in migration repository**
2. Destination DB structure is generated as **converted model** with the information stored in migration repository
3. Manual inspection and changes can be done on these models by the user.
4. Once done, destination DB schema objects are created through SQL developer and then data is migrated

SQL Developer is the **primary tool provided by Oracle for migrating non-Oracle databases to Oracle.**

SQL developer supports the below-mentioned non-Oracle databases and versions for migration to Oracle:

Supported Non-Oracle Database	Supported Versions
Microsoft SQL Server	7.0, 2000, 2005, 2008 R1, and 2008 R2
Microsoft Access	97, 2000, 2002, 2003, 2007
MySQL	3.x, 4.x, 5.x
Sybase Adaptive Server	12 and 15
IBM DB2	LUW 8.x, 9.x
Teradata	12, 13

Characteristics

- It can also migrate DB objects (data model/structure) and application code apart from data:
  - DB Objects (like tables, synonyms, views)
  - Data (like table data, materialized view data)
  - DB application code (like function, package, procedure, view)
- 2 modes of migration – online and offline:
  - Online migration

    Connect directly to the source database and select the objects for migration. Suitable for less amount of data.
  - Offline migration

    Files prepared from non-Oracle database are used for migration, and this does not require connecting to the database. This approach is more suitable for migrating large amounts of data. Offline data move scripts can be generated using the migration wizard.
- Provides Migration Wizard which guides users through different steps of data migration, by capturing the minimal details and performing all migration tasks in the background.

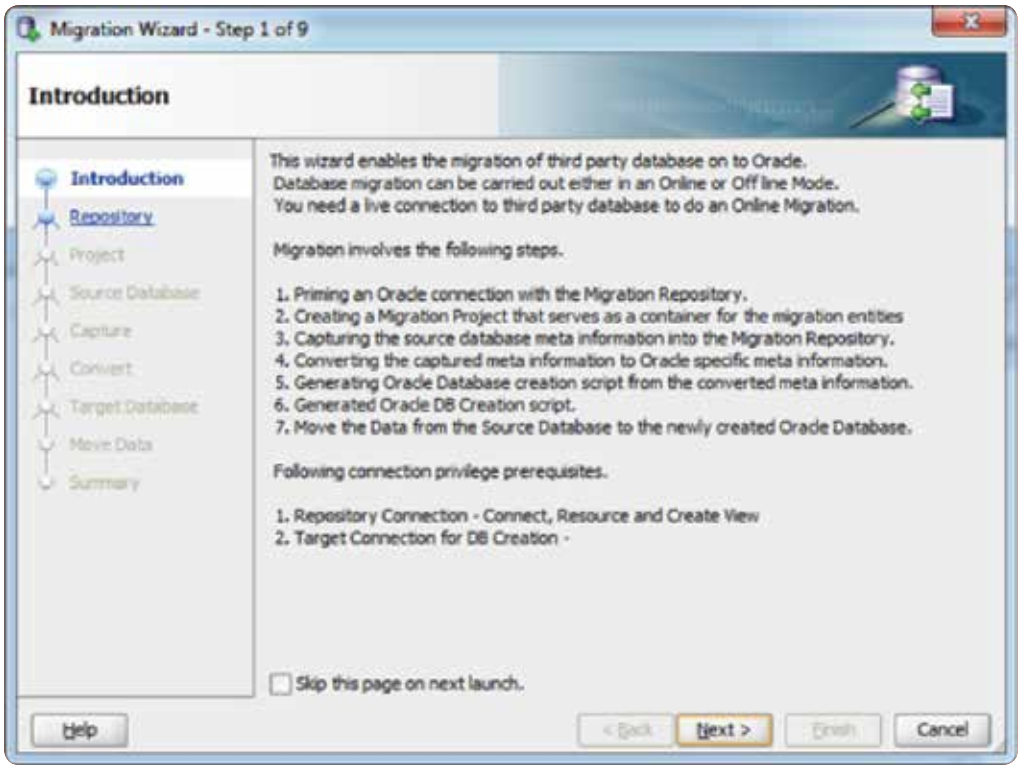


Fig 9. SQL Developer Migration Wizard

- Provides translation scratch editor tool to translate 3rd party SQL statements to PL/SQL

Consideration

This tool can be considered for the following migration requirements

#	Source DB	Target DB
1	Oracle	Oracle
2	Non-Oracle	Oracle

Migrating with SQL Developer is the easiest and the most convenient option for database migration to Oracle.

It reduces time, risks, and cost for migrations and takes advantage of proven best practices to mitigate migration issues.

6. SQL*PLUS COPY command

SQL*PLUS COPY command can copy data between two Oracle instances as long as both are in the same network. It **simply sends data over SQL*Net** and copies data from the source table to the destination table provided in the command.

COPY command **provides various options** for target like **CREATE** a new table, **REPLACE** existing table, **INSERT** into existing table or **APPEND** rows to existing table.

Execution Options

1. From Source Database (COPY TO command should be used)

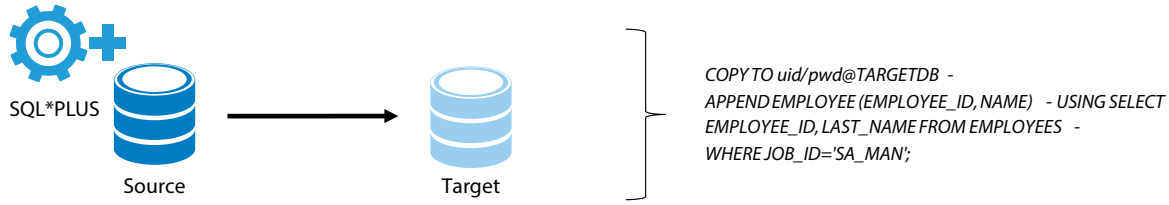


Fig 10. COPY command from Source DB

2. From Destination Database (COPY FROM command should be used)

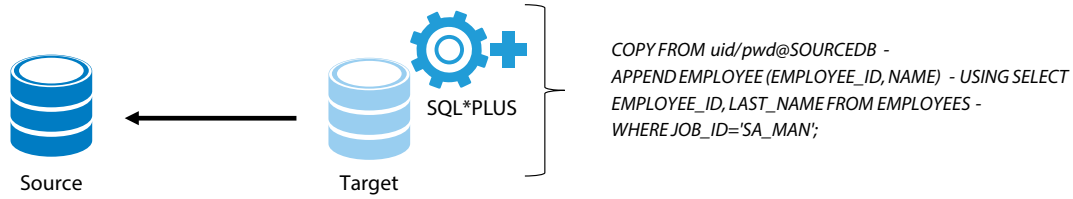


Fig 11. COPY command from Target DB

3. From Common/Neutral Database (COPY FROM/TO command should be used)

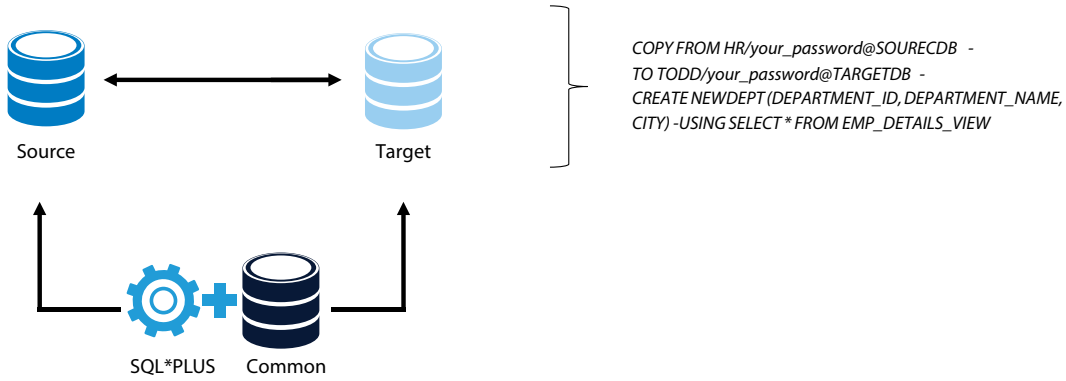


Fig 12. COPY command from Neutral DB

Characteristics

- ✓ **Alternative to DB Links,** and hence useful when DB links are not allowed
- ✓ Specific Data selection using SQL query
- ✓ COPY can be done between Oracle and non-Oracle databases
- ✓ **CLOB/BLOB** types not supported by COPY command
- ✓ Set the value of the variables, **ARRAYSIZE** (number of rows to fetch together at one time) & **COPYCOMMIT** (number of batches after which COPY commits changes to the database) high to improve the performance of this strategy
- ✓ Since it is SQL*plus command, it can't be invoked from SQL or PL/SQL

This approach is useful for both one-time migration as well as incremental data migration or synchronization

It is **deprecated from Oracle 10g** and instead of COPY, data pump is suggested.

7. Custom PL/SQL Procedures

Very often project teams tend to write custom PL/SQL scripts to perform data extract, transformation, and load into target Oracle database.

Since developers are working on the actual data and system(s), using this approach provides advantages like **close coupling** of migration scripts with source and destination data models, **easy mapping** of transformation requirements, and **customizing** the scripts as per ease of use.

Although there are certain advantages due to the developers being close to the actual data, this approach has some limitations as well, such as data transportation method being **sub-optimal** because of manual approach, need of separate **validation and checks,** additional **effort on performance tuning** for transformation and data extract/load procedures, and most importantly **dependency on knowledge** and level of details done by developer.

Irrespective of whatever the custom solution that is going to be considered, the following 5 phases of migration activity have to be followed:

1. Preparation
2. Execution
3. Validation
4. Testing
5. Tuning

Hence this approach is **not always an optimal solution, although it is very closely bound** to the actual system and data.

8. Oracle Golden Gate

Oracle Golden Gate is a **solution for real-time data integration and replication** between heterogeneous systems. It is a software package which enables **high availability, real-time integration, transactions capture, and data replication** between operational and analytical enterprise systems.

It is more suitable for **data replication** requirements rather than data migration. It supports bi-directional replication ensuring systems are operational 24/7, and distribution of data across enterprise to optimize the decision-making process. It also **supports replication from non-Oracle databases** (Oracle MySQL, MS SQL Server, Sybase, and IBM DB2).

Golden Gate Architecture

Golden Gate captures the transactions and DML changes written to redo and/or archive logs in Oracle database. From redo log files, trail files are generated which are transferred to the target and then used to read and load the data into target tables/objects.

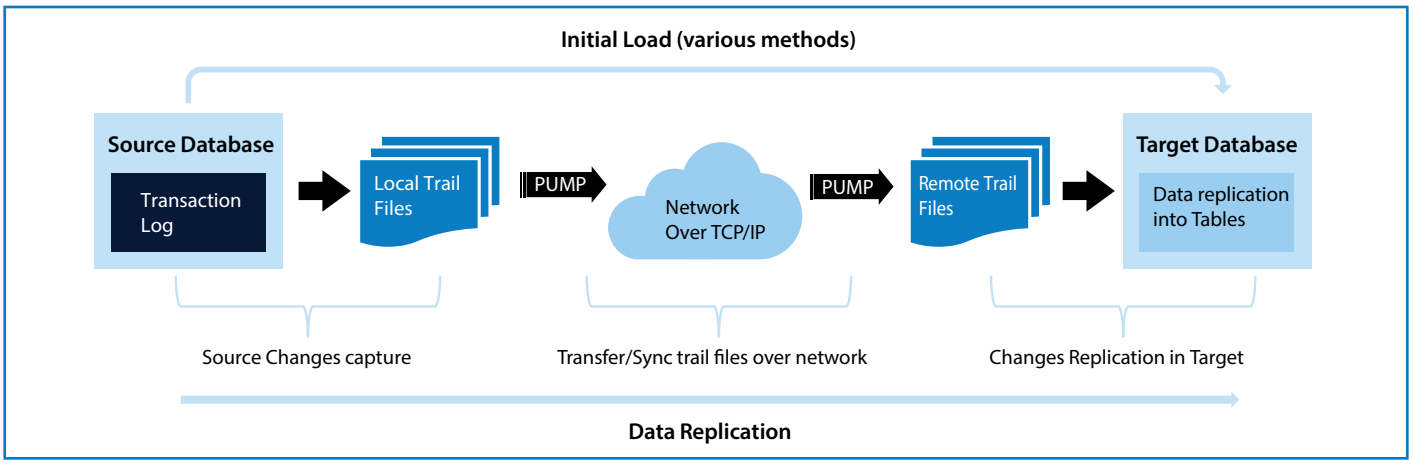


Fig 13. Golden Gate Architecture

Oracle Golden Gate architecture comprises of three primary components:

1) **Capture**
   This component captures the transactional changes and DDL changes from source system, using transaction and redo log(s).

2) **Trail Files**
   Contain details of the operation for changes on data in transportable and platform independent format.

3) **Delivery**
   Gets the data from trail files on remote/target database, and applies the data changes into the target tables.

Initial load of the data can be done separately to perform initial sync-up of data between source and destination. This can be done using Data Pump or any other option

Key Characteristics

- ✓ Trail files contain data changes in transportable and platform independent format and provide continuous data capture from source, even if target system is unavailable. They can be used for recovering and synchronizing the data after system is online.
- ✓ Zero downtime operational capability
- ✓ Supports transaction integrity to avoid data consistency issues
- ✓ Real-time data capture and replication
- ✓ Event-based interaction and capture from source system
- ✓ Conflict detection and resolution for data replication

Golden Gate's **replication process** keeps the **source** and **target** in **sync** providing real-time integration between the systems

Golden Gate Replication Strategies

Oracle Golden Gate provides various replication strategies to ensure a comprehensive real-time information environment.

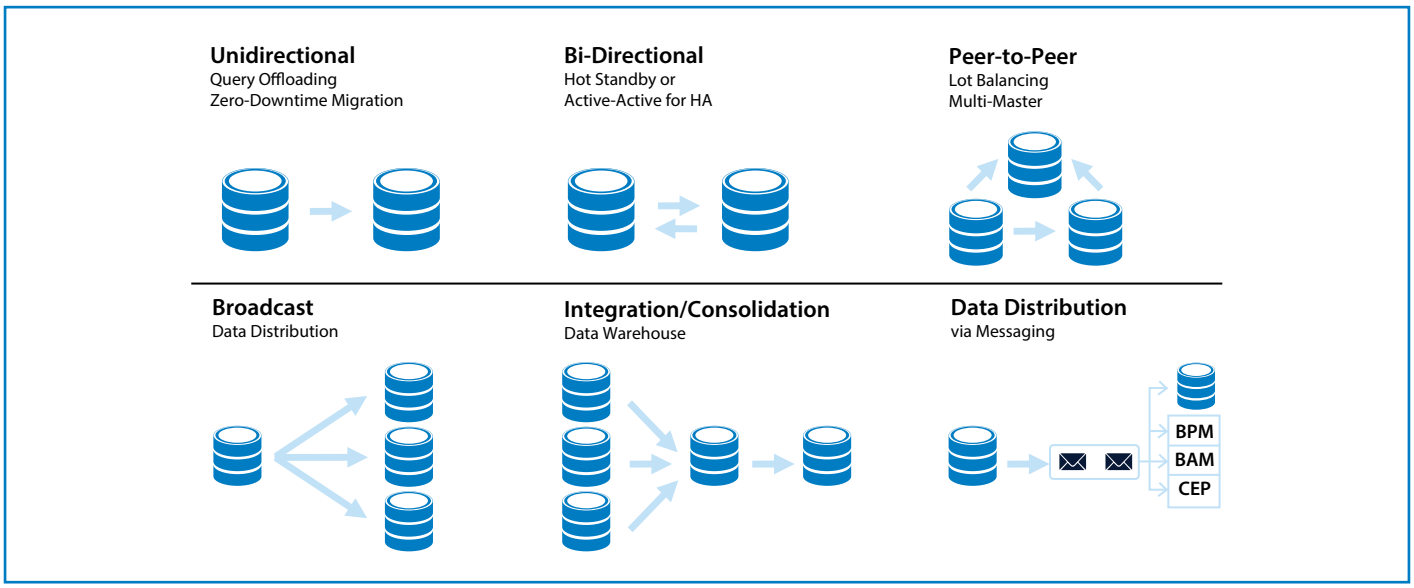


Fig 14. Golden Gate Replication Strategies

Replication Strategy	Characteristics
One-to-One	• Real-time feeding of reporting DB, enabling Live Reporting • Supports DDL replication
One-to-Many	• Dedicated site for backup data • Dedicated site for Live Reporting, separate from backup • Minimizes corruption of data, as backup is separate from reporting DB
Many-to-One	• Centralized data center consolidating information from remote sites • Useful for retail industry, central order processing • Useful for multiple bank branches serving same customer account • Data feeds to operation data store/data warehouse, supporting Operational Business Intelligence
Cascading	• Data distribution from master to multiple systems to carry out a transaction
Bi-Directional (Active-Active)	• Live standby and high availability • Load distribution and performance scalability
Bi-Directional (Active-Passive)	• Fastest possible recovery and switchover • Reverse direction data replication ready

Following is the list of supported source and target databases for Golden Gate replication providing a heterogeneous replication environment:

Supported Source Databases	Supported Target Databases
• c-tree • DB2 for Linux, UNIX, Windows • DB2 for z/OS • MySQL • Oracle • SQL/MX • SQL Server • Sybase • Teradata	• c-tree • DB2 for iSeries • DB2 for Linux, UNIX, Windows • DB2 for z/OS • Generic ODBC • MySQL • Oracle • SQL/MX • SQL Server • Sybase • TimesTen

Golden Gate is tightly integrated with Oracle database and uses lightweight streaming APIs built exclusively to provide better performance and scalability. It is a licensed product from Oracle and is now the tool of choice for data synchronization and replication between two systems.

9. Oracle Streams

Oracle Streams is a unified solution for sharing the data and events, within or across databases, in a stream. Oracle streams allow data propagation between Oracle databases (homogeneous) as well as non-Oracle and Oracle databases (heterogeneous). Oracle Streams was introduced with Oracle 9i Release 2 (9.2) and is available only with Enterprise Edition.

Oracle Streams Architecture

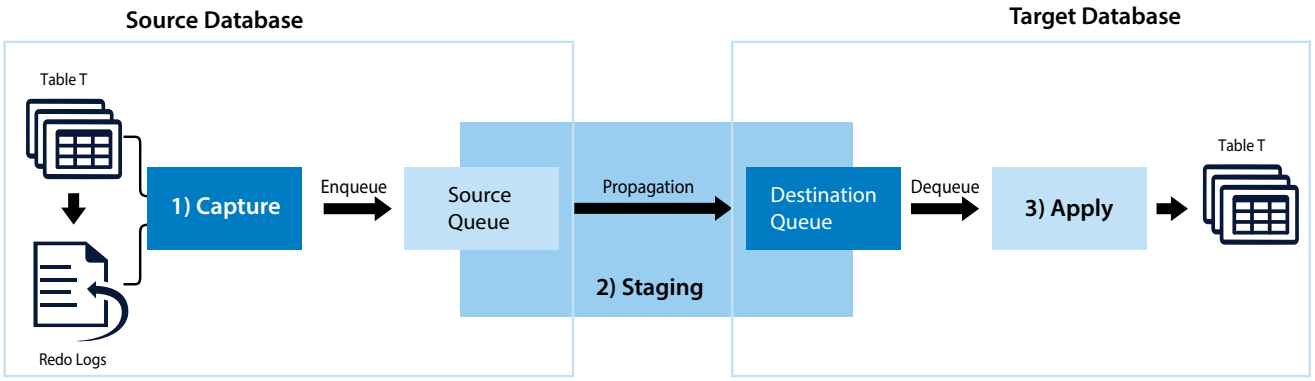


Fig 15. Oracle Streams Data Replication Architecture

Oracle streams perform the data replication using 3 main processes (Capture, Staging, and Apply):

1) **Capture:** Captures data updates, events, or messages from source database via redo logs, and formats as Logical Change Records (LCR) to be put into the queue as messages.

2) **Staging:** This process stores messages into a queue. Messages can be propagated to another queue, within the same or a different database. Propagation is scheduled using job queues.

3) **Apply:** Consume LCRs from queue and apply into target database either directly or by invoking user-defined package for further processing. This provides more flexibility for processing the messages.

Characteristics

- ✓ Messages can be queued using redo and archive logs (implicit capture) or by users and applications manually (explicit capture)
- ✓ Integrated feature of Oracle Database, no separate installation required
- ✓ Can capture DMLs as well as DDLs
- ✓ Rule-based filtering providing more control on data capture
- ✓ Flexibility of data transformation during apply stage through custom functions or procedures
- ✓ 1-1, 1-N, N-1, hub-and-spoke configurations by subscription to staging area
- ✓ Data sharing between Oracle and Non-Oracle databases (using database gateway)
- ✓ With Oracle 11g, extended datatype supported – XMLType, VARRAY, Custom Objects & tables

Oracle provides 2 inbuilt packages to set up and interact with the Oracle streams - DBMS_STREAMS_ADM & DBMS_PROPAGATION_ADM. Some important APIs are listed below:

API	Description
DBMS_STREAMS_ADM.ADD_TABLE_RULES	To specify the rules for capture and apply processes
DBMS_STREAMS_ADM.START_CAPTURE	Start capture process on source
DBMS_STREAMS_ADM.START_APPLY	Start Apply process on target
DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES	Define propagation from source to target queue

Usages

Oracle Streams is a useful tool for:

- Data replication
- Real-time ETL for data warehouse needs
- System availability during database migration or application upgrade
- Message queuing
- Event management and notification
- Integration with non-Oracle databases
- Provides infrastructure for various other features of Oracle

Oracle Streams in a Heterogeneous Environment

Oracle to non-Oracle

- Captures changes in source Oracle database
- Apply process runs in source Oracle DB & applies changes to data in non-Oracle DB via DB specific gateway

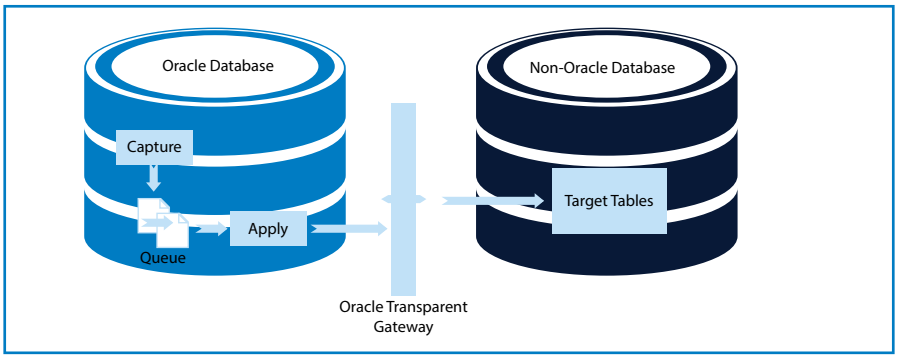


Fig 16. Oracle to non-Oracle migration using Oracle Streams

Non-Oracle to Oracle

- User application capture changes in source non-Oracle database
- Convert the captured changes to Logical Change Record (LCR) format for queuing
- Push the LCR events to Oracle DB for apply process

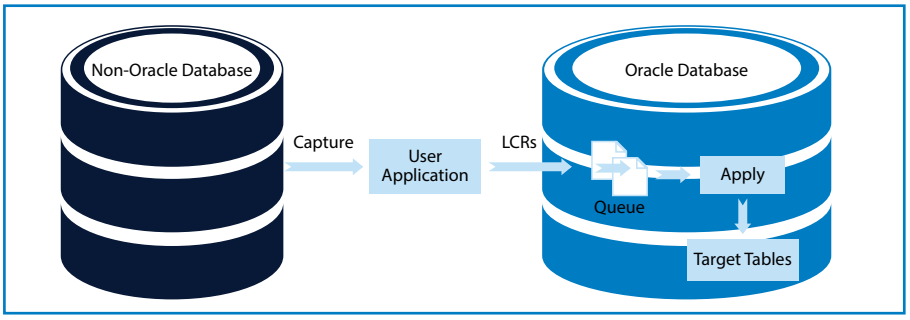


Fig 17. Non-Oracle to Oracle migration using Oracle Streams

Oracle Streams is useful for setting up data replications between live/online databases, and to keep source and destination systems in sync. It should be considered as a solution for data replication and synchronization rather than data migration.

For one time data migration, other options of transportable tablespaces, data pump, or SQL*Loader are preferred.

From 12c, Oracle Streams is deprecated and Golden Gate is the solution for data replication.

10. Oracle Recovery Manager (RMAN)

Oracle Recovery Manager (RMAN), introduced in Oracle 8i, is a tool for backup and restoration of Oracle database. RMAN is a part of standard database installation and can be launched from command line or through Oracle Enterprise Manager.

This approach should be considered as a **data backup and recovery utility** rather than a data migration utility.

RMAN performs incremental backup of database and using restore or recover command either complete or point-in-time recovery can be done. For performing the migration to a new database, control files and backup files can be transported to a new host and then using RMAN database can be recovered.

Considerations

✓ Database must be in ARCHIVELOG mode for performing online backup. Otherwise DB needs to be shut down for performing backup.

✓ Automated management of the backup files

✓ Easy to create a duplicate or standby database

✓ Can exclude any tablespace during copy, except SYSTEM or tablespaces containing rollback or undo segment.

✓ Database duplication over network using Active Duplication (from Oracle 11g)

✓ Can be a fully automated process

Limitations

The below-mentioned files will not be backed up by RMAN:

✓ Network configuration related files

✓ ORACLE HOME related files

✓ Password files

Important RMAN Commands

```
/* to do backup of current control file */
RMAN> backup current controlfile;
/* to back up the control file as part of a tablespace backup operation */
RMAN> backup tablespace users include current controlfile;
/* to backup server parameter file */
RMAN> backup spfile;
/* to force RMAN to back up a file regardless of whether it's identical to previously backed up file using force operation */
RMAN> backup database force;
/* to backup complete database */
RMAN> backup database;
/* to backup database with archivelogs */
RMAN> backup database plus archivelogs;
/* to backup all archive logs */
RMAN> backup archivelog all;
/* to backup specific data file */
RMAN> backup datafile 5 tag dbfile_5_bkp;
```

Usages

- Create a duplicate database using point-in-time recovery (PITR), from backup upto/until one week / one month ago or specified time/date
- Create duplicate database on same host or remote host
- Restore the database when one or all datafiles are lost, using specific datafile name
- Recover dropped tablespace by either complete database recovery, or cloning the database from backup and recreate tablespace using export and import from clone

11. Oracle Active Data Guard (ADG)

Oracle Data Guard is the solution for creating and maintaining standby databases, which are useful for data protection, disaster recovery, and high availability. In addition to the Oracle Data Guard which is inbuilt with Oracle EE, Active Data Guard is an option available from 11g onwards with advanced capabilities.

Active Data Guard Architecture

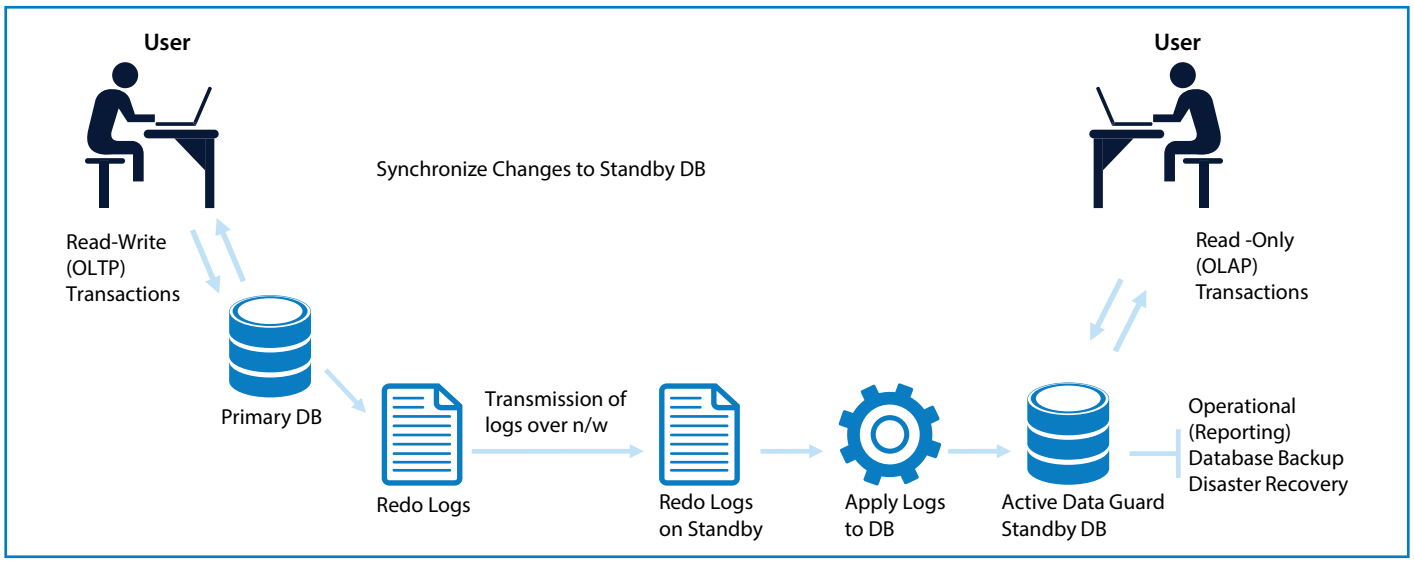


Fig 18. Oracle Active Data Guard Architecture

As shown in the above diagram, Active Data Guard configures standby database, which can be used as a backup and DR instance, as well as for performing operational (read-only) transactions.

Active Data Guard makes it possible to synchronize the OLTP triggered changes from Primary DB to the Standby DB, keeping the Standby DB online, and hence enabling the live reporting.

Data Guard Services

Data Guard architecture comprises of following key services for various capabilities:

- **Redo Log Transport** – Controls the automated transfer of archive redo logs from primary to standby database(s).
- **Log Apply** – To apply the redo log files on standby database and load the tables for data synchronization
- **Role Transition/Switchover** – Transition the database role from primary to standby and vice versa. This is useful during database failover operations.

Characteristics

- ✓ Standby database can be opened for read-only access without stopping data synchronization
- ✓ Live data sync-up enables offloading reporting and operational (read-only) transactions to standby database
- ✓ Efficient use of system resources by load distribution between primary and standby databases
- ✓ Transactions commit only after all redo data has been written on primary and at least one of the synchronized standby databases, providing maximum availability and data protection.
- ✓ Optimizes performance by allowing asynchronous update of standby redo logs
- ✓ Balancing data availability against performance requirements
- ✓ Supports automatic gap detection and resolution
- ✓ Integration with Oracle Database

Usages

- ✓ Setup a snapshot standby database to use for pre-production testing
- ✓ Using standby database as source for incremental backups, offloading from primary database
- ✓ Performing system upgrades/maintenance without downtime on primary database, using role switchover service.

Like RMAN, it should be considered as solution for **database backup** (with some additional features) rather than data migration.

Migration Approaches Comparison Chart

All the above-mentioned options are evaluated against the industry benchmarks and some of the commonly evaluated features like complexity, skill level, TTM, etc. The following table gives a holistic comparison of various options, based on Oracle 11g (SQL*PLUS Copy based on Oracle 10g).

#	Option(s)	Solution Complexity	Granularity	Time to Market	Scalability	Data Transformation	Support for Non-Oracle DB	Migration Scope
1	**Oracle Data Pump**	Low	Medium	High	Low	Vanilla Migration	No	One Time / Continuous
2	**Transportable Tablespaces**	Medium	Low	Medium	Medium	Vanilla Migration	No	One Time / Continuous
3	**Copy Over DB Link**	Low	High	Medium	Low	Moderate Transformations	No	One Time / Continuous
4	**SQL*Loader**	Medium	Medium	Medium	Medium	High Customization	Yes	One Time / Continuous
5	**SQL Developer**	Medium	High	Medium	Medium	Moderate Transformations	Yes	One Time / Continuous
6	**SQL*PLUS COPY**	Medium	High	Medium	Low	Moderate Transformations	Yes	One Time / Continuous
7	**Custom PL/SQL**	Medium	High	High	Low	High Customization	No	One Time / Continuous
8	**Golden Gate**	High	Medium	Low	High	Moderate Transformations	Yes	Continuous
9	**Oracle Streams**	High	High	Low	High	Moderate Transformations	Yes	Continuous
10	**RMAN**	High	Low	Medium	Medium	Vanilla Migration	No	Continuous
11	**Oracle Active Data Guard**	Medium	Medium	Low	High	Vanilla Migration	No	Continuous

Solution Complexity – This parameter specifies the knowledge level required on Oracle and the utility features for implementing a data migration solution

Granularity – The granular level of data selection that can be done, i.e., table level, record level, specific columns, etc.

Time to Market – Indicator of time required for data migration, and time required for new systems to be available online with data. Source and target might be out of sync, if any transactions are done on source database during this time.

Scalability – How scalable the migration option is, when data volume is increased in terms of number of records as well as number of entities/tables to be migrated.

Data Transformation – Indicator of how much data transformation and customization is supported by the specific approach. There are 3 levels of transformation:

1) **Vanilla Migration** – Source tables are copied as-is to the target without any change to structure or data
2) **Moderate Transformations** – Some structure changes (column add/remove, etc.) can be applied during migration
3) **High Customization** – Source and target data structures can be significantly different, as required for BI systems. Also data can be transformed based on custom processing rules

Support for non-Oracle DB – Indicates if migration can be done to/from Non-Oracle Databases.

Migration Scope – Indicates if the approach can be used for one-time migration only or whether it can be used for incremental/continuous data migration to keep source and target in sync.

Following table depicts some common use cases and data migration approaches which are suitable candidates for the use case requirements.

#	Use Case	Best Fit Migration Options
1	**Migrate the data to target system as-is, without any transformations**	• Oracle Data Pump • Transportable Tablespaces • RMAN • Oracle Data Guard
2	**Migrate data to target system with transformations**	• Copy Over DB Link • SQL*PLUS COPY • Custom PL/SQL • Golden Gate
3	**Migrate data from non-Oracle to Oracle database**	• SQL Developer • SQL*Loader • Golden Gate
4	**Setup DR or backup database**	• RMAN • Oracle Data Guard • Golden Gate • Oracle Streams
5	**Setup of real-time reporting environment, separate from transaction (OLTP) system**	• Golden Gate • Oracle Streams • Oracle Data Guard
6	**Set up multiple clones of source system, as Dev/Test/SIT environments**	• Transportable Tablespaces • Oracle Data Pump • Oracle Data Guard • RMAN
7	**Migration to new system in minimum possible time**	• Golden Gate • Oracle Data Guard • Oracle Streams
8	**Perform incremental backup and data sync-up, i.e., continuous data migration needs in live system**	• Golden Gate • Oracle Streams • Oracle Data Guard

As we can see from the above matrix, different data migration options are suited for different scenarios and user needs. Taking into account different parameters and impact to the migration, a suitable option should be selected and implemented.

This comparison is based on some key parameters as mentioned above to evaluate different data migration approaches.

Depending on the project scope and requirements, actual results may vary and hence a proof of concept is recommended to finalize the data migration approach that is to be used.

Oracle to NoSQL (Big Data) Migration

With Big Data taking the center stage in every system, it is natural for developers and designers as well to think about the question – How do I migrate my Oracle database to Big Data or NoSQL? The answer to this question is based on a key consideration: How Oracle and NoSQL models are mapped, and what data needs to be migrated from RDBMS to NoSQL.

Here we are not discussing what type of data and what specific data needs to be migrated, as that is a separate topic in itself. We are sharing some of the tools available in the market, which can be handy for migrating from Oracle to NoSQL database(s).

NoSQL databases are based on different data models, instead of a single model (relational for RDBMS)

#	NoSQL Models	Implementation
1	Document model	MongoDB, CouchDB
2	Key-value pair	Oracle NoSQL DB, DynamoDB
3	Column based	Cassandra, HBase
4	Graph database	Neo4J

So a single migration utility cannot cater to migration needs of different NoSQL databases. Due to this need, there are various tools and utilities which support one or other NoSQL DB.

We are enlisting some of the tools available below, and teams can choose any of these or some different tool(s) for migration based on their requirements.

#	Tools/Utilities	Description
1	**BigDataPumper**	Part of Big Data migration suite offered from spViewer Software (www.spviewer.com). It provides GUI-based software and supports migration from Oracle to these NoSQL databases – MongoDB, Couchbase, Cassandra, CouchDB, HBase
2	**Apache Sqoop**	Tool designed for bulk data transfers between RDBMS and Apache Hadoop, and can be used to populate data in HDFS or populated tables in Hive or Hbase. It is a top-level Apache project, and available under Apache License for customized implementation
3	**JSON-Based**	JSON provides a common interoperable platform and as such many NoSQL databases provide in-built utilities /tools to load the data from JSON file format. Oracle data can be converted into Target NoSQL JSON and can be migrated into the target database. Some JSON tools are as listed below:- • **JSON2SSTABLE** – for migration to Cassandra • **MongoImport** – for migration to Mongo DB

JSON using SQL

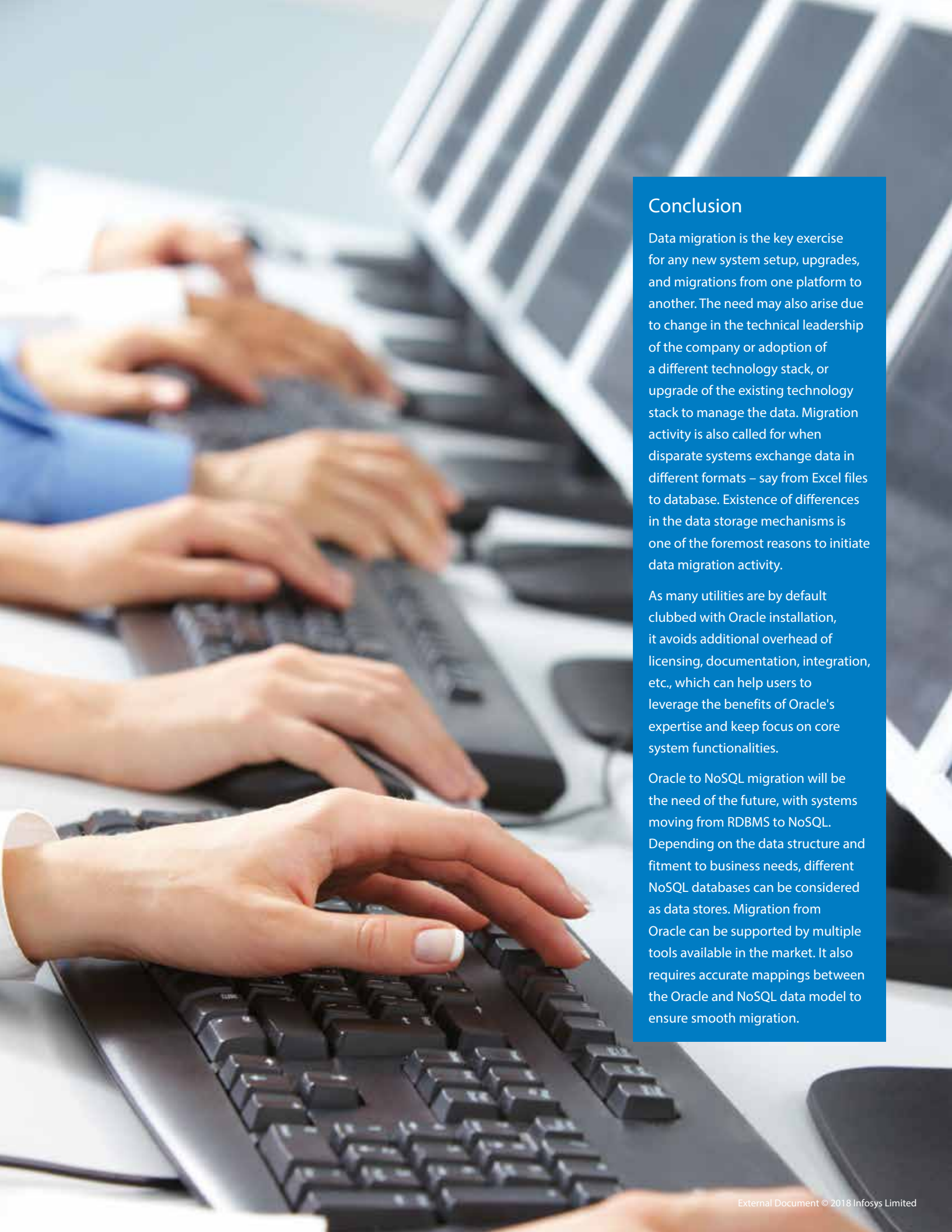
For migrating data from Oracle to any document model for NoSQL database, JSON formatted data can be generated and fed directly into the document model. JSON format can be created using simple SQL statements as described below:

```
SELECT
'{
        "FISCAL_ID":"'||fiscal_id||'",
        "FISCAL_PERIOD":"'||fiscal_period||'",
        "FISCAL_ST_DT":"'||TO_CHAR (fiscal_start_date,'DD-MON-YYYY') ||'",
        "FISCAL_END_DT":"'||TO_CHAR (fiscal_end_date,'DD-MON-YYYY') ||'",
"CRT_DT":"'||TO_CHAR (crt_date,'DD-MON-YYYY') || '",
"FISCAL_METRICES": {
                "Q1": {
                        "TOTAL_CASH_TRANSACTION":"'||q1_total_cash_transaction||'",
                        "TOTAL_MI_TRANSACTION":"'||q1_total_mi_transaction||'",
                        "TOTAL_WIRE_TRANSACTION":"'||q1_total_wire_transaction||'"
                },
                "Q2": {
                        "TOTAL_CASH_TRANSACTION":"'||q2_total_cash_transaction||'",
                        "TOTAL_MI_TRANSACTION":"'||q2_total_mi_transaction||'",
                        "TOTAL_WIRE_TRANSACTION":"'||q2_total_wire_transaction||'"
                },
                "Q3": {
                        "TOTAL_CASH_TRANSACTION":"'||q3_total_cash_transaction||'",
                        "TOTAL_MI_TRANSACTION":"'||q3_total_mi_transaction||'",
                        "TOTAL_WIRE_TRANSACTION":"'||q3_total_wire_transaction||'"
                },
                "Q4": {
                        "TOTAL_CASH_TRANSACTION":"'||q4_total_cash_transaction||'",
                        "TOTAL_MI_TRANSACTION":"'||q4_total_mi_transaction||'",
                        "TOTAL_WIRE_TRANSACTION":"'||q4_total_wire_transaction||'"
                }
        }
}' genval
FROM tg_fiscal_data;
```

This will generate JSON object as below, which can be fed into the NoSQL database.

```
{
        "FISCAL_ID":"0001265378",
        "FISCAL_PERIOD":"2010",
        "FISCAL_ST_DT":"01-JAN-2010",
        "FISCAL_END_DT":"31-DEC-2010",
        "CRT_DT":"05-JAN-2011",
        "FISCAL_METRICES": {
                "Q1": {
                        "TOTAL_CASH_TRANSACTION":"1000",
                        "TOTAL_MI_TRANSACTION":"100",
                        "TOTAL_WIRE_TRANSACTION":"10000"
                },
                "Q2": {
                        "TOTAL_CASH_TRANSACTION":"2000",
                        "TOTAL_MI_TRANSACTION":"200",
                        "TOTAL_WIRE_TRANSACTION":"20000"
                },
                "Q3": {
                        "TOTAL_CASH_TRANSACTION":"3000",
                        "TOTAL_MI_TRANSACTION":"300",
                        "TOTAL_WIRE_TRANSACTION":"30000"
                },
                "Q4": {
                        "TOTAL_CASH_TRANSACTION":"4000",
                        "TOTAL_MI_TRANSACTION":"400",
                        "TOTAL_WIRE_TRANSACTION":"40000"
                }
        }
}
```

In a nutshell, migration from Oracle to NoSQL database is specific to the target data store and data model, and different available utilities or tools can be leveraged to complete the migration. Since this is evolving, many new tools will be coming up to support different functionalities and ease the task of data migration between RDBMS and NoSQL.



Conclusion

Data migration is the key exercise for any new system setup, upgrades, and migrations from one platform to another. The need may also arise due to change in the technical leadership of the company or adoption of a different technology stack, or upgrade of the existing technology stack to manage the data. Migration activity is also called for when disparate systems exchange data in different formats – say from Excel files to database. Existence of differences in the data storage mechanisms is one of the foremost reasons to initiate data migration activity.

As many utilities are by default clubbed with Oracle installation, it avoids additional overhead of licensing, documentation, integration, etc., which can help users to leverage the benefits of Oracle's expertise and keep focus on core system functionalities.

Oracle to NoSQL migration will be the need of the future, with systems moving from RDBMS to NoSQL. Depending on the data structure and fitment to business needs, different NoSQL databases can be considered as data stores. Migration from Oracle can be supported by multiple tools available in the market. It also requires accurate mappings between the Oracle and NoSQL data model to ensure smooth migration.

References

1) IDC Storage and Data Service Overview, November 2013

2) Oracle Documentation
http://www.oracle.com/technetwork/documentation/index.html#database

3) Oracle Technology Network (OTN) – Database
http://www.oracle.com/technetwork/database/index.html

4) Database Migration Technologies
http://www.oracle.com/technetwork/database/migration/index.html

5) Oracle Golden Gate Overview
http://www.oracle.com/technetwork/middleware/goldengate/overview/index.html

6) Oracle Streams Features Overview
http://www.oracle.com/technetwork/database/streams-fov-11g-134280.pdf

7) Oracle Database High Availability Solutions
http://docs.oracle.com/cd/E11882_01/server.112/e17157/planned.htm

8) BigDataPumper
http://www.spviewer.com/bigdatapumper.html

9) MongoImport
http://docs.mongodb.org/manual/reference/program/mongoimport/

10) SSTABLE2JSON documentation
http://www.datastax.com/docs/1.0/references/sstable2json

11) SQL Server data migration approaches
http://www.infosys.com/microsoft/resource-center/documents/sql-server-data-migration-approaches.pdf

About the Authors

Nagesh Mittal

Technology Architect

Nagesh has primary expertise in Oracle database, advanced PL/SQL, performance monitoring, and tuning, and his areas of interest are data modeling, performance optimization, and Oracle technologies. He is a member of the Performance Engineering team under the Manufacturing Tech Group.

He can be reached at Nagesh_Mittal@infosys.com

Ravi Shankar Anupindi

Sr. Technology Architect

Ravi leads the performance Engineering CoE team under the manufacturing vertical within Infosys. His areas of interest include exploring latest technologies and leveraging them to derive business benefits out of them.

He can be reached at RaviShankar_Anupindi@infosys.com

Kalidhasan Velumani

Technology Architect

Kalidhasan is a core member of Performance Engineering CoE team under the manufacturing unit in Infosys, with expertise in Oracle database and technologies.

He can be reached at Kalidhasan_V@infosys.com

For more information, contact askus@infosys.com

Infosys®
Navigate your next