Oracle to SQL Server Migration Handbook

(Engineering References)

March 29, 2017

This document contains our consolidated engineering references. It is a living document that is intended to aid our internal team and our customers to complete specific Oracle to SQL server migration assessment tasks.

As we must respond to technology changes and are constantly evaluating how to better help our customers, this document is not intended to be used an official technical guide, but rather as engineering references that are subject to change. Similarly, we cannot guarantee that any information in this document will be error-free or kept up to date.

This document contains information obtained from Internet public domains. We have provided references in the form of URLs to indicate where the original information was obtained.

Table of Contents

Introduction

About the Assessment

This document focuses on the tasks of migrating Oracle database to the latest SQL Server Database. If migration requires changes to the feature/functionality, then the possible impact of each change on the applications that use the database must be considered carefully. This document assumes the reader is familiar with the database being migrated and the applications that use the database, thus able to assess the impact on the applications before and after database migration.

In this document, different solutions or approaches, workaround solutions for migrating Oracle database to SQL Server are described, each using one or more alternatives. Some methods are quick and easy, while others requiring greater effort to prepare and experiment.

SQL Server, Visual Studio tools, and SQL Server Migration Assistant (SSMA) described in this document are subject to change, so make sure you have latest versions of the documentation and tools before you start.

About this Document

The purpose of this document is to provide detailed explanation of features/functionalities enabled in the Oracle database, recommended migration approach, and any required workaround. This document can help migration planners and designers to understand the features used in source databases, effort involved in the migration.

This document was prepared based on the information available at the time, for example, the capabilities of Oracle Database, SQL DB Server, and migration tools such as SSMA.

Terminology Regarding Database Features

Acronym	Details
Feature ID	Sequence ID used in the scanning tool
Feature Enablement	This SQL script to determine if a feature is enabled (utilized) in the database.
Effort Score	Effort Score in the scale of 1-10 (1 represent low effort and 10 high). The scale is used to understand the level of complexity for the feature.
Category	Classification of the features

Assessment Summary

Oracle DB to SQL DB migration assessment exercise is to provide support and knowledge to aid customers to complete the following tasks:

- Pre-migration assessment
- Migration effort estimation
- Migration monitoring

Scanning tool, dashboard and documentation are provided to get insights at following areas:

- Database Features Enablement
- Level of migration complexity
- Magnitude of migration

Approximately 110 features/functionalities have been studied in Oracle databases. SQL Scripts have been developed to analyze level of feature enablement in Oracle DB. The SQL scripts are classified into three level:

Level 1 Scripts: To determine if the feature/functionality is enabled (utilized) within the source database

Level 2 Scripts: To analyze and categorize the features to understand the complexity of the migration:

- Simple: Migration can be performed using SSMA
- Medium: Migration can be performed using SSMA with additional customized migration steps
- Complex: Migration must be addressed outside of SSMA

Level 3 Scripts: To deep dive into the features that require customization/workaround. This assessment results help customers to further to analyze the cost and benefits of migration.

The overall migration assessment is carried through the following steps

1. Microsoft prepares a selected list of Oracle database features that are to be evaluated. A full list of features is documented in this document.
2. Microsoft team prepares Oracle SQL scripts to assess each of the features. The scripts are provided to customers.
3. The customer Oracle DBA(s) runs the Oracle SQL scripts against the Oracle Databases to be migrated, and shares the results with Microsoft team.

4. Using the results produced by customer, Microsoft team runs another set of programs to analyze the results. The key functions of the programs are:
    - ✓ Collect information on which Oracle Database features are enabled
    - ✓ Produce total counts of enabled Oracle Database features, grouped by complexity
    - ✓ Compare the counts with full list of features to derive the % of enablement
    - ✓ Summarize the migration complexity for each Oracle Database
5. The results of the above scripts are analyzed using Microsoft Power BI Desktop Application.
6. Power BI Desktop Application presents the results in the form of interactive reports
7. The results are shared with customer.

Feature List

Feature ID	Feature	Category	Notes
1	Database Audit	Security	
2	Security	Security	
3	Automated Maintenance Tasks	Admin	
4	Database Size	Performance	
5	Used Size	Performance	
6	Database Mail	Admin	
7	Collation	SQL	
8	Authentication	Security	
9	Database Version	Admin	
10	Data Encryption	Security	
11	Global Temporary Tables	Admin	
12	Table Partitioning	Admin	
13	Database Replication	Admin	
14	Database Mirroring	Admin	
15	Stored Procedures	SQL	
16	Memory Usage	Performance	
17	CPU Resources Usage	Performance	
18	Shut down	Admin	
19	Data Concurrency and Consistency	Performance	
20	Views	SQL	
21	Indexing	SQL	
22	Queries	SQL	
23	Database Backup	Admin	
24	Database Restore	Admin	
25	Log Files	Admin	
26	Trace Files	Admin	
27	Regular Expressions	SQL	
28	Constraints	SQL	
29	DML	SQL	
30	Data Types	SQL	
31	DDL	SQL	
32	Joins and Operations	SQL	
33	NoSQL Features	General	
34	Set Schema	General	
35	Oracle licensing components	General	
36	Logins	Security	

37	Triggers	SQL	
38	Data Dictionary	SQL	
39	Privileges	Security	
40	Access Control	Security	
41	Log Shipping	Admin	
42	Tables	SQL	
43	Data Navigator	Tools	
44	SQL Book Marks	Tools	
45	Charts	General	
46	Favorites	Tools	
47	Monitor	Tools	
48	Compare	General	
49	Cluster	SQL	
50	Column-level check constraint	SQL	
51	Packages	SQL	
52	Synonyms	SQL	
53	Sequences	SQL	
54	Snapshot	SQL	
55	Operators	SQL	
56	Built-in-Functions	SQL	
57	Locking Concepts and Data Concurrency Issues	SQL	
58	Change data capture	SQL	
59	Data Collector	Tools	
60	Database Logs	General	
61	File Groups	SQL	
62	Text Search	SQL	
63	Functions	SQL	
64	Linked Server instances	General	
65	Service Broker	Admin	
66	Processes and Threads	General	
67	High availability	General	
68	Scalability	Admin	
69	In-Memory optimization	General	
70	Connection multiplexing	Admin	
71	Connection Pooling	Admin	
72	Database Queuing	Admin	
73	Incremental backup and recovery	Admin	
74	Instead of Triggers	SQL	
75	Parallel load	General	
76	Sample scan	SQL	
77	Transparent Application Failover	SQL	
78	Fast-start fault recovery	General	

79	SQL Optimizer Plan Stability (Stored Outlines)	Admin	
80	Online index rebuilds	SQL	
81	Export transportable tablespaces	SQL	
82	Materialized Views	SQL	
83	Bitmap indexes	General	
84	Oracle Parallel Query (OPQ)	SQL	
85	Parallel DML	SQL	
86	Parallel index rebuilding	SQL	
87	Parallel index scans	SQL	
88	Parallel backup & recovery	Admin	
89	Oracle connection manager (CMAN)	General	
90	Oracle Streams	General	
91	Function-based in indexes	SQL	
92	Tablespace point in time recovery (TSPITR)	SQL	
93	Flashback Data Archive	Admin	
94	Automated SQL Tuning	SQL	
95	Oracle Licensing Components	Admin	
96	Parallel Shared pool	Admin	
97	Parallel Buffers		
98	Oracle DB Vault	Admin	
99	Advanced Queue	Admin	
100	Event Triggers	SQL	
101	Supplemental Logging	Admin	
102	Degree of Parallelism	Admin	
103	Oracle Resource Profiler	Admin	
104	Case Sensitive Password	Security	
105	RAC Cluster	Admin	
106	Block Change Tracking	Admin	
107	Streams, CDC, Capture	Admin	
108	Advanced rewrite	Admin	
109	Feature Usage Statistics	Admin	
110	Data Guard Replication	Admin	

Feature Details and Migration Approach

Database Logging

Feature ID	25
Feature	Database Logging
Description	Redo logs are transaction journals. Each transaction is recorded in the redo logs. Though redo generation is expensive operation, Oracle uses online redo logs as hot backups in case of instance crashes to ensure recoverability to a consistent state. The online redo log files contain the information necessary to replay a transaction, committed or not. Even uncommitted transactions can be written to the online redo log files. Before a commit is complete, the transaction information is written to the online redo log files. And changes to your rollback or undo segments are also written to the online redo log files. In that sense, they also contain the information to undo a transaction.
Category	HA/DR

To Find Feature Enablement	Oracle gave users the ability to limit redo generation on tables and indexes for better performance by setting them in NOLOGGING mode. Be careful never to use NOLOGGING option under Data guard setup. DB replication relies on redo logs.
	
	On the other hand, FORCE LOGGING can be used on tablespace or database level to force logging of changes to the redo. This may be required for sites that are mining log data, using Oracle Streams or using Data Guard (standby databases).
	
	`SELECT force_logging FROM v$database;`
	
	`SELECT tablespace_name, force_logging FROM dba_tablespaces;`
	
	`SELECT * FROM v$logfile;`
	
	To create a table in NOLOGGING mode:
	`CREATE TABLE t1 (c1 NUMBER) NOLOGGING;`
	
	To enable NOLOGGING for a table/database:
	`ALTER TABLE t1 NOLOGGING;`
	`ALTER DATABASE force logging;`

Recommendation	**Feature Description:** SQL Server requires a transaction log in order to function. That said there are two modes of operation for the transaction log: Simple

	and Full. In Full mode, the transaction log keeps growing until you back up the database. In Simple mode: space in the transaction log is 'recycled' every Checkpoint. SQL Server ensures data durability and recovery capabilities using Write-Ahead Logging, hardening a log record before a transaction begins. SQL Server can write log records describing a DB modification before it writes the actual change to the data or object. If SQL Server can't write log records, it won't commit. For this reason, its recommended leaving log auto-growth enabled. Log file: `C:\Program Files\Microsoft SQL Server\MSSWL\Data\MyDB.Idf` **Feature Comparison:** Like Oracle redo logging, SQL Server records database transactions in transaction logs. Each transaction record contains the undo and redo image of the transaction. Database logging in SQL Server is typically sent through a single log .ldf file. On the surface, this appears to be much different from oracle where logs are broken up into groups of logs called Redo Log Groups, but both architectures are very similar when look at the structure of the .LDF. Each physical .LDF file is a group of Virtual Log Files, (VLFs), that behave much like a Redo Log Group does in Oracle. VLFs can be viewed by running DBCC LOGINFO; Archiving is controlled via a periodic BACKUP LOG job in SQL Server. VLFs are compressed and set to .TRN files. After backup, VLF is cleared and can be reused. This differs from Oracle where they ARC internal process automatically moves full log files to an archive directory as they fill up, not on a reoccurring schedule. These files typically have a .ARC extension in Oracle are just copied/renamed right from the Redo Log Group.
Migration Approach	Migrating Transaction Logs In Oracle, information on transactions and the changes they make is recorded in REDO logs. The redo logs are common to the entire instance. In SQL Server, transactional changes are logged in the transaction log for the database whose objects are involved in the transaction. A database is created with a single default transaction log. The default transaction log has to be sized or new ones added based on the update activity against the database. To add a transaction log to a database using T-SQL, use the following syntax: `ALTER DATABASE database`

	`{ ADD LOG FILE < filespec > [ ,...n ]`
	`where <filespec> ::=`
	`( NAME = logical_file_name`
	`[ , FILENAME = 'os_file_name' ]`
	`[ , SIZE = size ]`
	`[ , MAXSIZE = { max_size | UNLIMITED } ]`
	`[ , FILEGROWTH = growth_increment ] )`
	
	Database Logging is enabled by default in SQL Server.
	
	Logging is based on three recovery models: simple, full, and bulk-logged. The recovery model for new databases is taken from the Model database. After the creation of the new database, you can change the recovery model using SSMS or following T-SQL:
	To set the Recovery Model:
	`USE master ;`
	`ALTER DATABASE model SET RECOVERY FULL ;`

References	http://searchoracle.techtarget.com/answer/What-information-do-redo-log-files-contain http://www.databases-la.com/?q=node/33 http://www.dba-oracle.com/concepts/archivelog_archived_redo_logs.htm http://users.wfu.edu/rollins/oracle/archive.html https://msdn.microsoft.com/en-us/library/ms190925.aspx http://www.sqlshack.com/beginners-guide-sql-server-transaction-logs/

Database Backup

<table>
<tr><td>Feature ID</td><td>23</td></tr>
<tr><td>Feature</td><td>Database Backup</td></tr>
<tr><td>Description</td><td>The following methods are valid for backing-up an Oracle database:
<table>
<tr><td>Export/Import</td><td>Exports are "logical" database backups in that they extract logical definitions and data from the database to a file. Using exports for your backups captures a snapshot in time of your database. use the CONSISTENT=Y to ensure the export dump in this snapshot is consistent across the board. using export for your backups does limit your recovery options. You cannot perform point-in-time recovery. You cannot roll forward any transactions performed after the export dump was created.</td></tr>
<tr><td>Cold or Off-line Backups</td><td>A cold (or off-line) backup is a backup performed while the database is off-line and unavailable to its users. shut the database down and backup up ALL data, log, and control files.</td></tr>
<tr><td>Hot or On-line Backups</td><td>A hot (or on-line) backup is a backup performed while the database is open and available for use (read and write activity). one can only do on-line backups when the database is ARCHIVELOG mode. Each tablespace that needs to be backed-up must be switched into backup mode before copying the files out to secondary storage (tapes). When in backup mode, Oracle will write complete changed blocks to the redo log files. Normally only deltas are logged to the redo logs. Also backup the control files and archived redo log files. The backup this way is an inconsistent backup because redo is required during recovery to bring the database to a consistent state.</td></tr>
<tr><td>RMAN Backups</td><td>while the database is off-line or on-line, use the "rman" oracle provided utility to back up the database. RMAN has many other features that the traditional hot and cold backup scripts cannot perform. Those features include, but are not limited to:

– Ability to perform incremental backups.
– Ability to recover one block of a datafile.
– Ability to perform the backup and restore with parallelization.
– Ability to automatically delete archived redo logs after they are backed up.</td></tr>
</table></td></tr>
</table>

	– Ability to automatically backup the control file and the SPFILE. – Ability to restart a failed backup without having to start from the beginning. – Ability to verify the integrity of the backup, and to test the restore process without having to actually perform the restore.

Recovery Manager (RMAN) is an Oracle Database client that's recommended way to perform backup and recovery tasks on your databases and automates administration of your backup strategies. RMAN ships with the database server and doesn't require a separate installation. The RMAN executable is located in your ORACLE_HOME/bin directory. It greatly simplifies backing up, restoring, and recovering database files.

Using RMAN, you can take a hot backup for your database, which will take a consistent backup even when your DB is up and running.

RMAN can be manual or automated by scripting with crontab scheduler or configured via Enterprise Manager Database Control Tool. RMAN optimizes performance by compression.

The RMAN BACKUP command supports backing up the following types of files:

- Datafiles and control files
- Server parameter file
- Archived redo logs
- RMAN backups
- The current server parameter file

Other files as network configuration files, password files, and the contents of the Oracle home, cannot be backed up with RMAN. Likewise, some features of Oracle, such as external tables, may depend upon files other than the datafiles, control files, and redo log. RMAN cannot back up these files. Use some non-RMAN backup solution for any files not in the preceding list.

Incremental backups can only be created with RMAN.

RMAN supports backup encryption for backup sets. You can use wallet-based transparent encryption, password-based encryption, or both.

Category	HA/DR
To Find Feature Enablement	Issuing below script will: • report on all RMAN backups like full, incremental & archivelog backups. • And will give you RMAN backup status along with start and stop timing.

```
select SESSION_KEY, INPUT_TYPE, STATUS,
 to_char(START_TIME,'mm/dd/yy hh24:mi') start_time,
 to_char(END_TIME,'mm/dd/yy hh24:mi') end_time,
```

```
 elapsed_seconds/3600 hrs
 from V$RMAN_BACKUP_JOB_DETAILS
 order by session_key;
```

Following script will give you SID, Total Work, Sofar & % of completion:

```
SELECT SID, SERIAL#, CONTEXT, SOFAR, TOTALWORK,
ROUND (SOFAR/TOTALWORK*100, 2) "% COMPLETE"
FROM V$SESSION_LONGOPS
WHERE OPNAME LIKE 'RMAN%' AND OPNAME NOT LIKE '%aggregate%'
AND TOTALWORK! = 0 AND SOFAR <> TOTALWORK;
```

```
SELECT start_time, end_time, input_type, input_type, status
FROM v$rman_backup_job_details
ORDER BY 1;
SELECT vbd.file#, vrbjd.start_time, vrbjd.end_time,
vbd.incremental_level, vrbjd.input_type, vrbjd.status
FROM v$rman_backup_job_details vrbjd, v$backup_datafile vbd
WHERE vbd.completion_time BETWEEN vrbjd.start_time AND
vrbjd.end_time
AND vrbjd.input_type <> 'ARCHIVELOG'
ORDER BY 2,1;
```

While executing backup, RMAN will generate backup logs, you can verify its backup logs to verify status of RMAN backups.

Additionally, You can query to V$RMAN_STATUS dictionary view for completed job information:

```
select OUTPUT from V$RMAN_OUTPUT;
```

To determine if RMAN is running a full backup or incremental backup, use INPUT_TYPE column from dictionary view `V$RMAN_BACKUP_JOB_DETAILS`

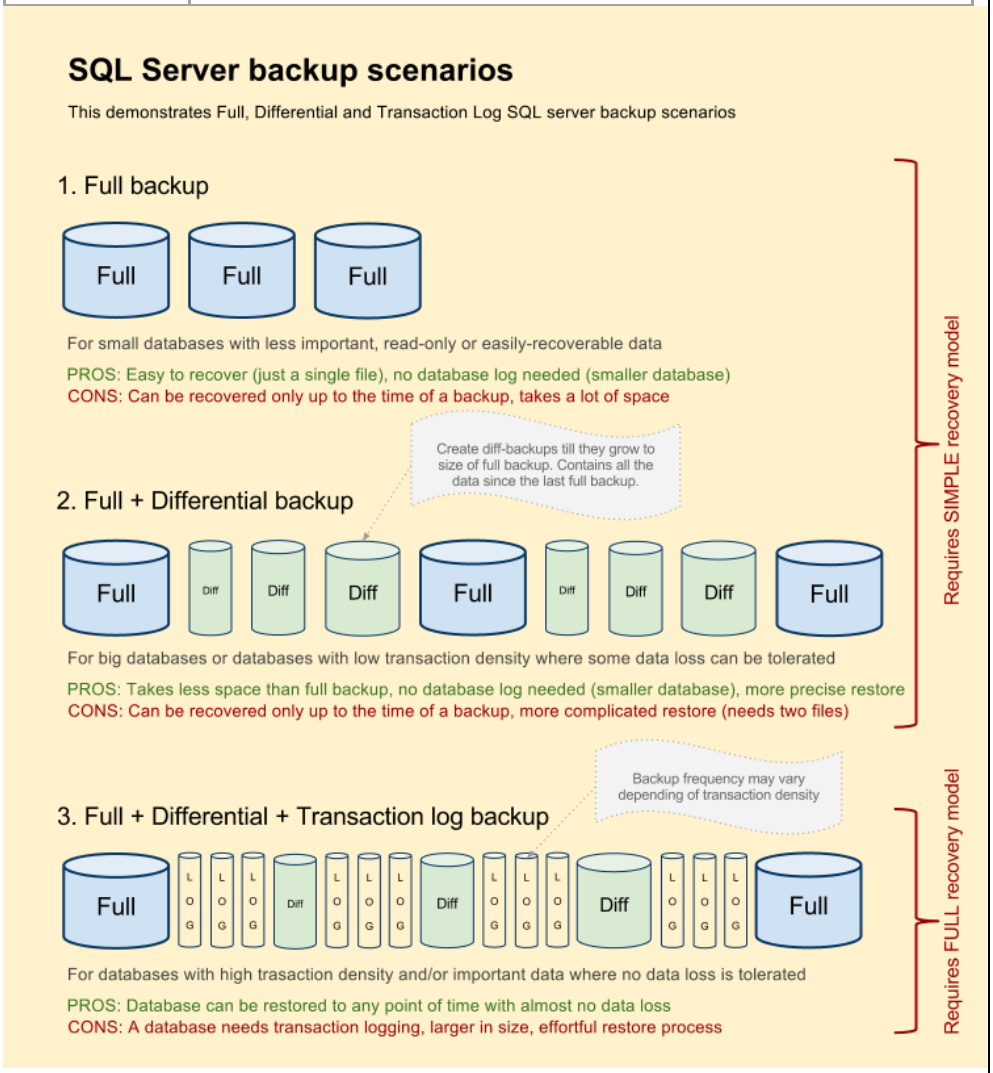
Recommendation

Feature Description:

In SQL Server, different types of backups can be create based on recovery model:

Full	a complete database backup which truncates the transaction log of inactive records
Differential	a backup of all of the changed data pages since the last full backup. Usually smaller than a full backup, assuming that not all pages have changed
Log	transaction log backup containing all transactions since the last transaction or full backup. Also truncates the log of all inactive log records
File	a way to backup individual database files
Filegroup	a way to backup a group of files contained inside of a filegroup

Copy-Only	a backup which can be taken without disrupting the log chain. Great for taking a copy of a production database for development purposes
Mirror	allows you to backup to more than once device simultaneously
Partial	similar to filegroup but will back up the primary, all read/write filegroups and optionally, read only filegroups



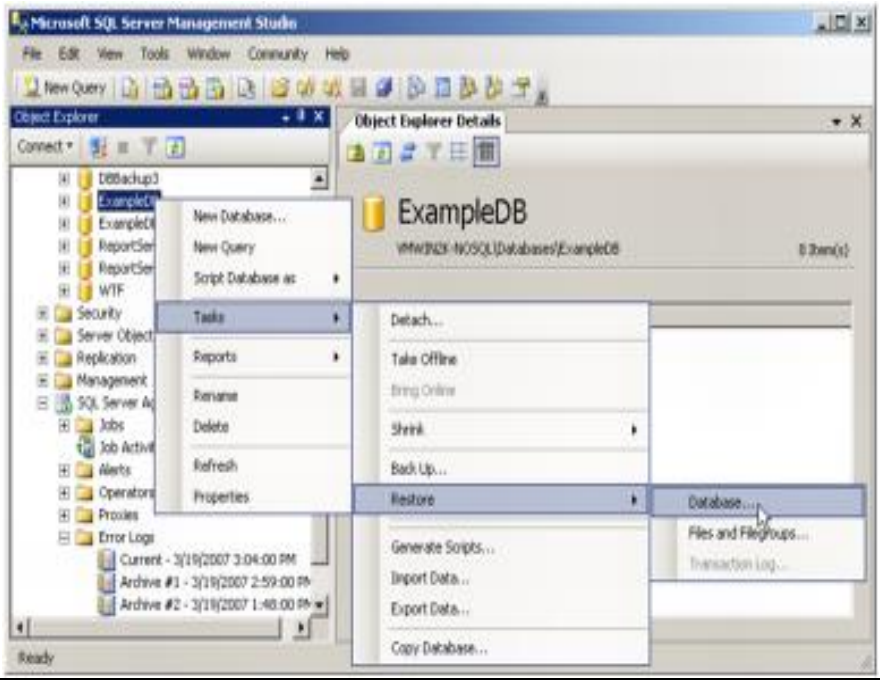
In SQL Server, use Maintenance Plans for scheduling backups. Use the Back Up Database Task in SQL Server Management Studio (SSMS) to add a backup task to the maintenance plan.
There are fine grained options to create backups for all system databases (master, msdb, model), all user databases, specific databases, portion of database- Files & Filegroups; backup type, set backup extension type, verify backup integrity and whether Back up the database to a file or to tape.

Feature Comparison:

	There are variety of hot & cold backups available in both Oracle and SQL Server to suit any business environment. Starting with SQL Server 2014, SQL Server supports backup encryption. Oracle Standard Edition, on the other hand, does not have backup encryption.
Migration Approach	Backup mechanism cannot to migrated through SSMA tool. In SQL Server, use Maintenance Plans for scheduling backups. Use the Back Up Database Task in SQL Server Management Studio (SSMS) to add a backup task to the maintenance plan. There are fine grained options to create backups for all system databases (master, msdb, model), all user databases, specific databases, portion of database- Files & Filegroups; backup type, set backup extension type, verify backup integrity and whether Back up the database to a file or to tape. SQL Server's built-in backup options support disk, tape and the cloud as backup devices. SQL Server Managed Backup to Azure allows an automatic database backup to Azure, based on changes done in the database. This feature schedules, performs and maintains the backups–all a DBA needs to do is specify a retention period. SQL Server Backup to URL allows you to easily backup directly to Microsoft Azure Blob Storage, removing the need to manage hardware for backups To create a maintenance plan using the Maintenance Plan Wizard in SSMS • In Object Explorer, click the plus sign to expand the server where you want to create a maintenance plan. • Click the plus sign to expand the Management folder. • Right-click the Maintenance Plans folder and select Maintenance Plan Wizard. • Follow the steps of the wizard to create a maintenance plan. • Use the Back Up Database Task dialog to add a backup task to the maintenance plan. Backing up the database is important in case of system or hardware failure (or user errors) that cause the database to be damaged in some way, thus requiring a backed-up copy to be restored. This task allows you to perform full, differential, files and filegroups, and transaction log backups.
References	http://www.orafaq.com/wiki/Oracle_database_Backup_and_Recovery_FAQ http://searchdatabackup.techtarget.com/feature/Choosing-the-best-Oracle-backup-strategy-for-your-environment https://www.mssqltips.com/sqlservertutorial/6/types-of-sql-server-backups/ https://msdn.microsoft.com/en-us/library/ms189647.aspx (back up maintenance task) https://msdn.microsoft.com/en-us/library/ms187510.aspx

Database Restore

Feature ID	24
Feature	Database Restore
Description	It is easier to restore from off-line backups as no recovery (from archived logs) would be required to make the database consistent. Nevertheless, on-line backups are less disruptive and don't require database downtime. Point-in-time recovery (regardless if you do on-line or off-line backups) is only available when the database is in ARCHIVELOG mode. To restore a physical backup of a datafile or control file is to reconstruct it and make it available to the Oracle database server. To recover a restored datafile is to update it by applying archived redo logs and online redo logs, that is, records of changes made to the database after the backup was taken. If you use RMAN, then you can also recover datafiles with incremental backups, which are backups of a datafile that contain only blocks that changed after a previous incremental backup. You have a choice between two basic methods for recovering physical files. You can: – Use the RMAN utility to restore and recover the database – Restore backups by means of operating system utilities, and then recover by running the SQL*Plus RECOVER command
Category	HA/DR
To Find Feature Enablement	
Recommendation	**Feature Description:** You can restore database by using SQL Server Management Studio(SSMS) or Transact-SQL. SQL Server Management Studio makes the restore process simple. Select the restore point you want to use, since a SQL backup file can hold multiple backups you may see more than one restore point listed, and also you can overwrite the existing database or rename a database. You could also restore backups created on another SQL Server using the SQL Management Studio tool. To restore an encrypted database, you need access to the certificate or asymmetric key used to encrypt that database. Without the certificate or asymmetric key, you cannot restore that database. You must retain the certificate used to encrypt the database encryption key for as long as you need to save the backup. You could even restore an older version database to SQL Server 2016, that database will automatically upgrade to SQL Server 2016. Typically, the database becomes available immediately. **Feature Comparison:** Similar to Oracle, SQL Server provides utilities as well as SQL commands to restore backed up database.
Migration Approach	Restore mechanism cannot be migrated through SSMA tool.

	SQL Server restoration can be configured manually via utilities or SQL commands. Choosing appropriate Backup and Restore Strategy is governed by your application's DR SLA requirements- typically measured by Recovery Time Objective (RTO) and Recovery Point Objective (RPO).
	
	Restore database using T-SQL
	`RESTORE` command restores backups taken using the `BACKUP` command:
	
	`--To Restore an Entire Database from a Full database backup (a Complete Restore):`
	`RESTORE DATABASE { database_name | @database_name_var }`
	`[ FROM <backup_device> [ ,...n ] ]`
	`[ WITH`
	`{`
	`[ RECOVERY | NORECOVERY | STANDBY =`
	`{standby_file_name | @standby_file_name_var }`
	`]`
	`| , <general_WITH_options> [ ,...n ]`
	`| , <replication_WITH_option>`
	`| , <change_data_capture_WITH_option>`
	`| , <FILESTREAM_WITH_option>`
	`| , <service_broker_WITH options>`
	`| , <point_in_time_WITH_options–RESTORE_DATABASE>`
	`} [ ,...n ]`
	`]`
	`[;]`
	
	Restore database using SQL Server Management Studio
	 

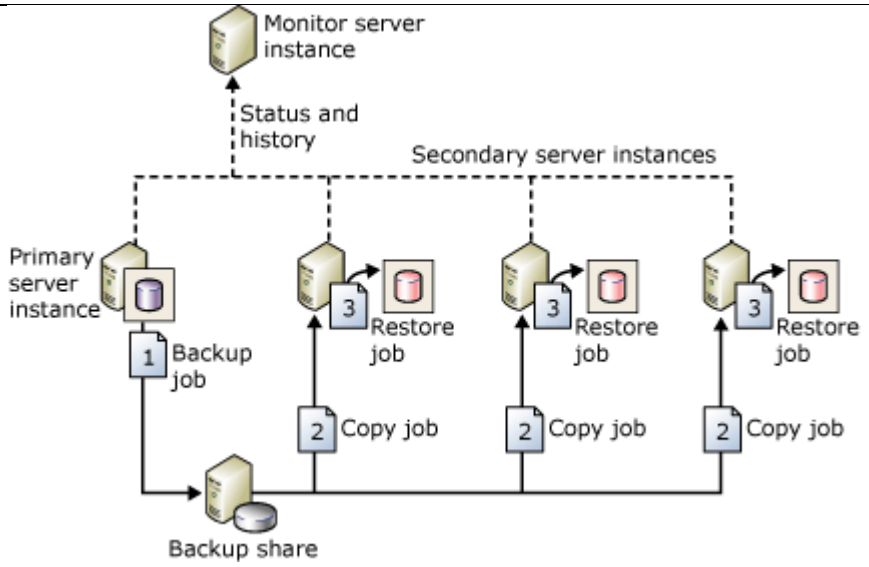
References	https://docs.oracle.com/cd/B19306_01/server.102/b14220/backrec.htm https://msdn.microsoft.com/en-us/library/ms189275.aspx https://msdn.microsoft.com/en-us/library/ms177429.aspx

	(Restore a Database Backup Using SSMS)

Log Shipping

Feature ID	41
Feature	Log Shipping
Description	• Oracle doesn't offer log shipping as a part of the core product, but it's possible to set up log shipping in Oracle. • Oracle's log shipping works by copying archived redo log files. There are no extra backup jobs to add. Instead, the log shipping jobs copy archived redo log files from the flash recovery area. Rotating out active redo log files will move them into the archive redo log file area. DBAs can take advantage of their existing backup strategy. It is still possible for an Oracle DBA to break log shipping by using NOARCHIVELOG mode or adding tablespaces or files without adding them on the secondary. Of course, a DBA can also use the FORCE LOGGING option to prevent users from switching to NOARCHIVELOG mode and breaking the log shipping. • When you're moving backups across the network, compression can help meet your recovery point objective. Oracle's compression is only found in either Oracle Enterprise Edition or customers using Oracle's backup to the cloud feature – Oracle Database Backup Service. However, it's trivial to leverage in-flight compression when moving files between Oracle instances. You can also use rsync to move files between primary and standby and enable the -z flag to ensure you get compression.
Category	HA/DR
To Find Feature Enablement	to check if the ARCHIVELOG mode is enabled: `SQL> archive log list;`
Recommendation	**Feature Description:** In SQL Server can do Log Shipping using SSMS or T-SQL scripts. Log shipping allows you to automatically send transaction log backups from a primary database on a primary server instance to one or more secondary databases on separate secondary server instances. The transaction log backups are applied to each of the secondary databases individually. An optional third server instance, known as the monitor server, records the history and status of backup and restore operations and, optionally, raises alerts if these operations fail to occur as scheduled. you can even make your log shipping secondary readable and use it for reporting using STANDBY mode. A log shipping configuration does not automatically fail over from the primary server to the secondary server. If the primary database becomes unavailable, any of the secondary databases can be brought online manually. Additionally, Log shipping can be used with following other features of SQL Server:

	• You can migrate from Log Shipping to Always On Availability Groups (SQL Server) • Database Mirroring and Log Shipping (SQL Server) • Log Shipping can be used in conjunction with Replication (SQL Server) SQL Server can compress backups in the Standard Edition of the product. This can either be enabled as a default SQL Server level setting or in the log shipping jobs. **Feature Comparison:** Like Oracle, SQL Server has support for log shipping options available, and can compress backups for better performance.
Migration Approach	SSMA doesn't support migrating Log Shipping. In SQL Server, you can set up Log Shipping manually using SSMS or T-SQL scripts. **A Typical Log Shipping Configuration** The following figure shows a log shipping configuration with the primary server instance, three secondary server instances, and a monitor server instance. The figure illustrates the steps performed by backup, copy, and restore jobs, as follows: 1. The primary server instance runs the backup job to back up the transaction log on the primary database. This server instance then places the log backup into a primary log-backup file, which it sends to the backup folder. In this figure, the backup folder is on a shared directory—the backup share. 2. Each of the three secondary server instances runs its own copy job to copy the primary log-backup file to its own local destination folder. 3. Each secondary server instance runs its own restore job to restore the log backup from the local destination folder onto the local secondary database. The primary and secondary server instances send their own history and status to the monitor server instance.



To configure log shipping using Transact-SQL

1. Initialize the secondary database by restoring a full backup of the primary database on the secondary server.
2. On the primary server, execute `sp_add_log_shipping_primary_database` to add a primary database. The stored procedure returns the backup job ID and primary ID.
3. On the primary server, execute `sp_add_jobschedule` to add a schedule for the backup job.
4. On the monitor server, execute `sp_add_log_shipping_alert_job` to add the alert job.
5. On the primary server, enable the backup job.
6. On the secondary server, execute `sp_add_log_shipping_secondary_primary` supplying the details of the primary server and database. This stored procedure returns the secondary ID and the copy and restore job IDs.
7. On the secondary server, `execute sp_add_jobschedule` to set the schedule for the copy and restore jobs.
8. On the secondary server, execute `sp_add_log_shipping_secondary_database` to add a secondary database.
9. On the primary server, execute `sp_add_log_shipping_primary_secondary` to add the required information about the new secondary database to the primary server.
10. On the secondary server, enable the copy and restore jobs.

References	http://docs.oracle.com/database/121/SBYDB/concepts.htm#SBYDB00010 https://www.brentozar.com/archive/2015/02/comparing-sql-server-oracle-log-shipping/ http://users.wfu.edu/rollins/oracle/archive.html

	https://msdn.microsoft.com/en-us/library/ms187103.aspx

Oracle Database Vault

Feature ID	98
Feature	Oracle Database Vault
Description	A standard problem with database security stems from the need for database administrators to have full access to the data they manage- a potential security hole. Oracle Database Vault with Oracle Database 12c provides greater access controls on data. It can be used to protect application data from the DBA and other privileged users as well as implementing robust controls on access to the database and application. The Database Vault Option allows you to restrict access granted with system-wide privileges, restrict administrative access to a defined realm of data, allowing for finer grained separation of administrative duties. A security administrator can set factors to define access to the database including Oracle commands available to the different classes of users and administrators and audit-specific dimensions of security. Realms can be defined for limiting access to specific database schemas and roles at a more granular level.
Category	Security
To Find Feature Enablement	`SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';`
Recommendation	In SQL Server, there is no direct equivalent feature to Database Vault. However, SQL Server does provide capability to restrict user data access from DBAs. Always Encrypted enables customers to confidently store sensitive data outside of their direct control. This allows organizations to encrypt data at rest and in use for storage in Azure, or to reduce security clearance requirements for their own DBA staff. With Always Encrypted, you can configure encryption for selected columns to protect sensitive data. These encrypted columns can then be managed by Access Control by keeping DBA restricted with privilege to decrypt or access sensitive data.
Migration Approach	SSMA can't migrate Database Vault features automatically. In SQL Server, configure Always On encryption. To access encrypted columns (even if not decrypting them) `VIEW ANY COLUMN` permissions need to be explicitly granted. **T-SQL example to enable encryption**

	The following Transact-SQL creates column master key metadata, column encryption key metadata, and a table with encrypted columns.
	
	`CREATE COLUMN MASTER KEY MyCMK`
	`WITH (`
	`KEY_STORE_PROVIDER_NAME = 'MSSQL_CERTIFICATE_STORE',`
	`KEY_PATH = 'Current User/Personal/f2260f28d909d21c642a3d8e0b45a830e79a1420'`
	`);`
	`----------`
	`CREATE COLUMN ENCRYPTION KEY MyCEK`
	`WITH VALUES`
	`(`
	`COLUMN_MASTER_KEY = MyCMK,`
	`ALGORITHM = 'RSA_OAEP',`
	`ENCRYPTED_VALUE = 0x04E234173C....154F86`
	`);`
	`----------`
	`CREATE TABLE [dbo].[Students] (`
	`[StudentID] INT IDENTITY (1, 1) NOT NULL,`
	`[SSN] CHAR (11) COLLATE Latin1_General_BIN2 ENCRYPTED WITH (COLUMN_ENCRYPTION_KEY = [ColumnEncryptionKey1], ENCRYPTION_TYPE = Deterministic, ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256') NOT NULL,`
	`[FirstName] NVARCHAR (50) NULL,`
	`[LastName] NVARCHAR (50) NOT NULL,`
	`[StreetAddress] NVARCHAR (50) NOT NULL,`
	`[City] NVARCHAR (50) NOT NULL,`
	`[ZipCode] CHAR (5) NOT NULL,`
	`[BirthDate] DATE ENCRYPTED WITH (COLUMN_ENCRYPTION_KEY = [ColumnEncryptionKey1], ENCRYPTION_TYPE = Deterministic, ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256') NOT NULL,`
	`CONSTRAINT [PK_dbo.Students] PRIMARY KEY CLUSTERED ([StudentID] ASC)`
	`);`

References	http://www.oracle.com/technetwork/database/options/oracle-database-vault-external-faq-2032888.pdf https://www.oracle.com/database/database-vault/index.html https://docs.oracle.com/database/121/DVADM/getting_started.htm#DVADM002 https://msdn.microsoft.com/en-us/library/mt163865.aspx

Database Audit

Feature ID	1
Feature	Database Audit
Description	• Auditing facilitates database activity monitoring. It's the recording of selected user database actions. Monitoring statements, privileges, or objects • Security policies can trigger auditing when specified elements in an Oracle database are accessed or altered. • AUDIT_SYS_OPERATIONS initialization parameter- Enables or disables the auditing of top-level operations directly issued by user SYS, and users connecting with SYSDBA or SYSOPER privilege. This parameter should be enabled on ALL production databases. • Oracle Database writes the audit records to the audit trail of the operating system. The database audit trail consists of a single table named SYS.AUD$. Audit trail records contain different types of info, depending on the events audited and the auditing options set. • Oracle Database allows audit trail records to be directed to an operating system audit trail if the operating system makes such an audit trail available to Oracle DB. If not, then audit records are written to a file outside the database. the database will write a trace file of the session actions (for sys or sysdba) to the dump directory location specified by AUDIT_FILE_DEST. • If you set the AUDIT_TRAIL initialization parameter to XML or XML, EXTENDED, it writes the audit records in XML format. AUDIT_TRAIL enables or disables database auditing. • To enable -- alter system set audit_sys_operations=TRUE scope=spfile; -- then restart
Category	Security

To Find Feature Enablement	`show parameter audit_sys_operations;`
	
	`show parameter audit_trail;`
	`select * from dba_stmt_audit_opts union select * from dba_priv_audit_opts;`
	
	-- if a non-container database
	`conn / as sysdba`
	
	-- connect to each PDB in turn and run the following queries
	
	`show parameter audit`
	
	`SELECT MAX(logoff$time)`
	`FROM sys.aud$;`
	
	`SELECT MAX(timestamp#), MAX(ntimestamp#)`
	`FROM sys.fga_log$;`
	
	`SELECT table_name, tablespace_name, num_rows`
	`FROM dba_tables`
	`WHERE table_name IN ('AUD$', 'FGA_LOG$')`

	`ORDER BY 1;` The audit trail is stored in the SYS.AUD$ table. It's contents can be viewed directly or via the following views. • DBA_AUDIT_EXISTS • DBA_AUDIT_OBJECT • DBA_AUDIT_SESSION • DBA_AUDIT_STATEMENT • DBA_AUDIT_TRAIL • DBA_OBJ_AUDIT_OPTS • DBA_PRIV_AUDIT_OPTS • DBA_STMT_AUDIT_OPTS
Recommendation	**Feature Description:** DDL triggers and notifications can aid in auditing SQL Server server-level auditing is resilient, available in all editions, and provides T-SQL call stack frame info SQL Server supports user-defined audit groups and audit filtering Can use T-SQL to enable audit by creating the audit specification for specific database and specific access group. The Audit action items can be individual actions such as SELECT operations on a Table, or a group of actions such as SERVER_PERMISSION_CHANGE_GROUP. SQL Audit Events track the following three categories of Events: – Server Level: These actions include server operations, such as management changes, and logon and logoff operations. – Database Level: These actions include data manipulation languages (DML) and Data Definition Language (DDL). – Audit Level: These actions include actions in the auditing process. You could implement an audit trail quickly in SQL Server by creating shadow table for each table in database and triggers to log every time when a record is inserted, updated or deleted in the table. see last link in the list for Audit Trail Generator Script. The SQL Server Audit feature is built on top of Extended Events to leverage the performance benefits and provide both asynchronous and synchronous write capabilities (by default, SQL Server Audit uses the asynchronous event model). You could use SQL Profiler to see Workload Performance impact of Auditing and turn on audit on specific objects and specific logins. All editions of SQL Server support server level audits. Database level auditing is limited to Enterprise, Developer, and Evaluation editions. **Feature Comparison:**

	Similar to Oracle Audit Vault for DDL and DML statements All actions (DDL and DML) are auditable in SQL Server

Migration Approach	SSMA does not support migrating Auditing configurations.
	
	In SQL server, use T-SQL to enable audit by creating the audit specification for specific database and specific access group.
	
	T-SQL to create a server audit
	`USE master ;`
	`GO`
	`-- Create the server audit.`
	`CREATE SERVER AUDIT Payrole_Security_Audit`
	`TO FILE ( FILEPATH =`
	`'C:\Program Files\Microsoft SQL`
	`Server\MSSQL13.MSSQLSERVER\MSSQL\DATA' ) ;`
	`GO`
	`-- Enable the server audit.`
	`ALTER SERVER AUDIT Payrole_Security_Audit`
	`WITH (STATE = ON) ;`
	
	T-SQL to create a database-level audit specification
	(Following example creates a database audit specification called `Audit_Pay_Tables` that audits SELECT and INSERT statements by the `dbo` user, for the `HumanResources.EmployeePayHistory` table based on the server audit defined above.)
	
	`USE AdventureWorks2012 ;`
	`GO`
	`-- Create the database audit specification.`
	`CREATE DATABASE AUDIT SPECIFICATION Audit_Pay_Tables`
	`FOR SERVER AUDIT Payrole_Security_Audit`
	`ADD (SELECT , INSERT`
	`ON HumanResources.EmployeePayHistory BY dbo )`
	`WITH (STATE = ON) ;`
	`GO`

References	http://docs.oracle.com/cd/B19306_01/network.102/b14266/auditing.htm#DBSEG525 https://oracle-base.com/articles/8i/auditing#AuditOptions http://solutioncenter.apexsql.com/how-to-setup-and-use-sql-server-audit-feature/ https://msdn.microsoft.com/en-us/library/cc280663.aspx (Sql Server Audit Actions) https://msdn.microsoft.com/en-us/library/cc280386.aspx (SQL Server Audit) http://solutioncenter.apexsql.com/sql-server-database-auditing-techniques/

	http://techbrij.com/audit-trail-microsoft-sql-server-quickly

Authentication

Feature ID	8
Feature	Authentication
Description	Authentication the process of verifying that the login ID or username supplied by a user to connect to the database belongs to an authorized user. Oracle allows authentication of user account through the OS or through the database (server). Oracle allows a single database instance to use any or all methods. Oracle requires special authentication procedures for database administrators, because they perform special database operations. Oracle also encrypts passwords during transmission to ensure the security of network authentication. Once authenticated by the operating system, users can connect to Oracle more conveniently, without specifying a user name or password. Oracle Database can authenticate users attempting to connect to a database by using information stored in that database itself. To configure Oracle Database to use database authentication, you must create each user with an associated password.
Category	Security
Find Feature Enablement	an operating-system-authenticated user can invoke SQL*Plus and skip the user name and password prompts by entering the following: `SQLPLUS /`
Recommendation	**Feature Description:** SQL Server has two methods of authentication: – Windows authentication – SQL Server authentication Windows Authentication: When you are accessing SQL Server from the same computer it is installed on, you won't be prompted to type in username and password if you're using Windows Authentication. Authenticating with Windows domain logins, the SQL Server service already knows that someone is logged in into the operating system with the correct credentials, and it uses these credentials to allow the user into its databases. This works as long as the client resides on the same computer as the SQL Server, or as long as the connecting client matches the Windows credentials of the server. Ideally, Windows authentication must be used when working in an Intranet type of an environment. In enterprise environments, these credentials are normally Active Directory domain credentials. Windows Authentication is also a more convenient way to log-in into a SQL Server instance without typing a username and a password, however when more users are involved, or remote connections are being established with the SQL Server, SQL authentication should be used. Mixed authentication mode allows the use of Windows credentials but supplements them with local SQL Server user accounts that the administrator may create and maintain within SQL Server. SQL Server Authentication:

	SQL Authentication is the typical authentication used for various database systems, composed of a username and a password. An instance of SQL Server can have multiple such user accounts (using SQL authentication) with different usernames and passwords. In shared servers where different users should have access to different databases, SQL authentication should be used. Also, when a client (remote computer) connects to an instance of SQL Server on other computer than the one on which the client is running, SQL Server authentication is needed. Even if you don't define any SQL Server user accounts, at the time of installation a root account - sa - is added with the password you provided. Just like any SQL Server account, this can be used to log-in locally or remotely, however if an application is the one that does the log in, and it should have access to only one database, it's strongly recommended that you don't use the sa account, but create a new one with limited access. Microsoft's best practice recommendation is to use Windows authentication mode whenever possible. It allows you to centralize account administration for your entire enterprise in a single place: Active Directory. **Feature Comparison:** Like Oracle, SQL Server has two major methods of authentication: – OS authentication – Database authentication Password Policies can be enforced with authentications in both databases. These policies control password management including account locking, password aging and expiration, password history, and password complexity verification.
Migration Approach	In Oracle, most used Authentication methods are authentication by the database and authentication by the operating system. In SQL Server, the database modes in use are SQL Server Authentication Mode and the Windows Authentication Mode. The database authentication modes in Oracle and SQL Server are closely compatible and use a user name and password pair. The operating system authentication is quite different between Oracle and SQL Server. Oracle's operating system mode can only authenticate users with local accounts on UNIX servers. Windows authentication for SQL Server is actually performed by the domain and not the local account on the Windows server. The Oracle RDBMS also provides password management functions, such as account locking, password lifetime and expiration, password history, and password complexity verification. The SQL Server RDBMS does not provide these services, and Windows security is used to provide these features. Hence the migration of Oracle user names to SQL Server logins and users is dependent on the type of authentication in use as well as the requirements of password management. Migration options for Oracle logins based on authentication mode and the requirements on password management functionality:

Oracle Authentication Mode	Oracle Password Management	SQL Server Authentication Mode
Database	None	Database
Database	Required	Windows
Operating system	N/A	Windows

To add a new Windows authenticated login to a SQL Server instance using T-SQL, use the following syntax:

```
sp_grantlogin [ @loginame = ] 'login_name'
```

where

`login_name`

is of the form

`domain_name\domain_login_name`

To add a new database authenticated login to a SQL Server instance use following T-SQL:

```
sp_addlogin [ @loginame = ] 'login_name
        [ , [ @passwd = ] 'password' ]
        [ , [ @defdb = ] 'database_name' ]
        [ , [ @encryptopt = ] 'encryption_option' ]
```

where database_name specifies the database the login connects to after logging in (default database). While passwords are encrypted in SQL Server by default, the option exists to skip encryption to allow custom password encryption by the application using a different algorithm.

A user account should be created separately for the login in the default database.

```
sp_grantdbaccess [ @loginame = ] 'login_name'[, [ @name_in_db = ] 'user_name'
```

To create a user account to a SQL Server database using T-SQL, use the following syntax:

```
sp_grantdbaccess [ @loginame = ] 'login_name'[, [ @name_in_db = ] 'user_name'
```

The name chosen for the user account can be different from that for the login account.

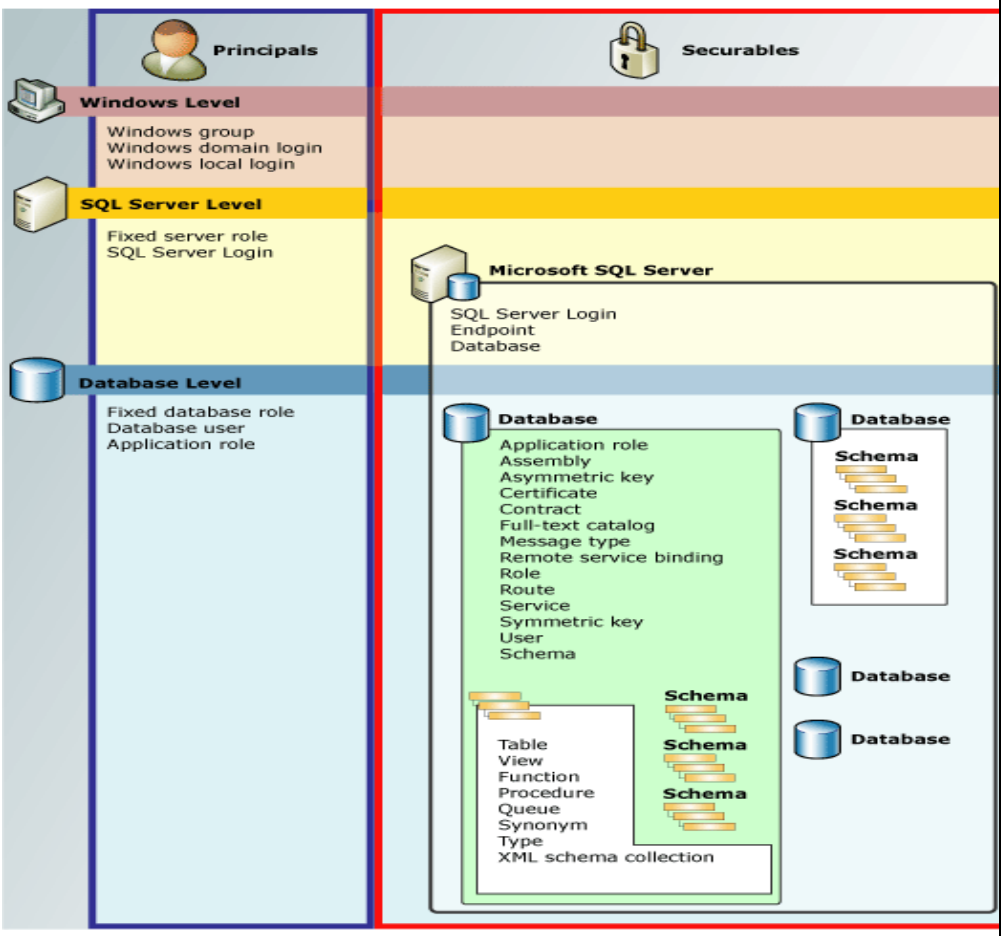
References	https://docs.oracle.com/cd/B19306_01/network.102/b14266/authmeth.htm#BABCGGEB (Oracle Authentication Methods) http://docs.oracle.com/cd/E25054_01/network.1111/e16543/authentication.htm#i1006458 https://msdn.microsoft.com/en-us/library/ms144284.aspx (Authentication modes in SQL Server)

Privileges

Feature ID	39
Feature	Privileges
Description	A privilege is a right to execute an SQL statement or to access another user's object. In Oracle, there are two types of privileges: **system privileges** and **object privileges**. A privilege can be assigned to a user or a role. The set of privileges is fixed, that is, there is no SQL statement like create privilege xyz... **System privileges** In Oracle 9.2, there are 157 system privileges, and 10g has even 173. These are privileges like create job, drop user, alter database, and can be displayed with: `select name from system_privilege_map;` System privileges can be audited. `sysdba` and `sysoper` the most important system privileges. There are five operations on Oracle that require the user to have SYSDBA privileges in order to perform them: – startup a database, – shutdown a database, – backup a database, – recover a database – and create a database `v$pwfile_users` lists all users who have been granted sysdba or sysoper privileges. **Object privileges** While Oracle has several object privileges, the ones commonly granted to users are SELECT, INSERT, DELETE, and UPDATE on tables and EXECUTE on stored programs. Object Privileges can be assigned on following DB objects: • Tables- select, insert, update, delete, alter, debug, flashback, on commit refresh, query rewrite, references, all • Views- select, insert, update, delete, under, references, flashback, debug • Sequence- alter, select • Packages, Procedures, Functions (Java classes, sources...)- execute, debug • Materialized Views- delete, flashback, insert, select, update • Directories- read, write

	• Libraries- execute • User defined types- execute, debug, under • Operators- execute • Indextypes- execute For a user to access an object in another user's schema, they need privilege to the object. Object privileges can be displayed using `all_tab_privs_made` or `user_tab_privs_made`. The `ROLE_ROLE_PRIVS`, `ROLE_SYS_PRIVS`, and `ROLE_TAB_PRIVS` data dictionary views contain information on the privilege domains of roles. The `GRANT ANY OBJECT PRIVILEGE` system privilege allows users to grant and revoke any object privilege on behalf of the object owner. INSERT, UPDATE, or REFERENCES privileges can be granted on individual columns in a table. **Assigning privileges to users and roles** GRANT- assigns a privilege to a user REVOKE- allows to take away such privileges from users and roles. Oracle stores the granted privileges in its data dictionary.
Category	Security
To Find Feature Enablement	Following query returns all system privilege grants made to roles and users: `SELECT count(*) FROM DBA_SYS_PRIVS;`
Recommendation	**Feature Description:** Like Oracle. SQL Server supports system and object level privileges. System and object privileges can be granted to Users directly or via Roles using the GRANT statement and removed using the REVOKE statement. SQL Server also has the additional DENY statement, which prevents users from exercising a privilege even when it has been granted to the user. In SQL Server, the REVOKE statement is used to remove (or cancel out) a previously granted or denied privilege. Conflict in permissions granted directly and through roles is always resolved in favor of the higher-level permission. The only exception to this is if users have been denied permissions (DENY) to an object either explicitly or through their membership in a role. If that is the case, they will not be granted the requested access to the object. Permission Hierarchy • Permissions have a parent/child hierarchy. • If you grant SELECT permission on a database, that permission includes SELECT permission on all (child) schemas in the database. • If you grant SELECT permission on a schema, it includes SELECT permission on all the (child) tables and views in the schema.

- The permissions are transitive; that is, if you grant SELECT permission on a database, it includes SELECT permission on all (child) schemas, and all (grandchild) tables and views.
- Permissions also have covering permissions. The CONTROL permission on an object, normally gives you all other permissions on the object.



Feature Comparison:
The following terminologies relating to privileges in Oracle and SQL Server are equivalent:

Oracle Terminology	SQL Server Terminology
Privilege	Permission
System privilege	Statement permission
Object privilege	Object permission
Predefined role permission (for example: DBA)	Implied permissions (for example: sysadmin)
Grantee	Security account

	Like Oracle, SQL Server has the same database object privileges. Oracle and SQL Server differ a lot in the system privileges that are available. Oracle has very granular (more than 100) system privileges. SQL Server system privileges, called statement permissions, are restricted to the following list: • BACKUP DATABASE • BACKUP LOG • CREATE DATABASE • CREATE DEFAULT • CREATE FUNCTION • CREATE PROCEDURE • CREATE RULE • CREATE TABLE • CREATE VIEW The rest of the Oracle system privileges are bundled into several large fixed roles. For example, the fixed database role `db_datareader` is equivalent to the SELECT ANY TABLE system privilege in Oracle.

Migration Approach	SSMA tool doesn't support automatic migration of privileges.
	
	In SQL Server, these privileges would need to be manually created using T-SQL/SSMS and assigned to principals like user, or roles.
	
	Use T-SQL queries with GRANT, DENY, and REVOKE to manipulate permissions. `sys.server_permissions` and `sys.database_permissions` catalog views provide information on permissions.
	
	You can GRANT and REVOKE privileges on database objects in SQL Server.
	You can grant users various privileges to tables- permissions can be any combination of SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALTER, or ALL.
	REFERENCES- Ability to create a constraint that refers to the table.
	ALTER- Ability to perform ALTER TABLE statements to change the table definition.
	
	`Use <database name>;`
	`Grant <permission name> on <object name> to <username\principle>;`
	
	`GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO smithj;`
	
	ALL does not grant all permissions for the table. Rather, it grants the ANSI-92 permissions which are SELECT, INSERT, UPDATE, DELETE, and REFERENCES
	`GRANT ALL ON employees TO smithj;`
	
	Grant EXECUTE permission on stored procedures to a user
	`GRANT EXECUTE ON dbo.procname TO username;`
	
	SELECT permission on the table (Region) , in a schema (Customers), in a database (SalesDB) can be achieved through any of below statements:
	`GRANT SELECT ON OBJECT::Region TO Ted`
	`GRANT CONTROL ON OBJECT::Region TO Ted`
	`GRANT SELECT ON SCHEMA::Customers TO Ted`

	`GRANT SELECT ON DATABASE::SalesDB TO Ted`
References	http://www.adp-gmbh.ch/ora/misc/users_roles_privs.html https://docs.oracle.com/cd/B19306_01/network.102/b14266/admusers.htm#DBSEG10000 (Administering User Privileges, Roles, and Profiles) https://docs.oracle.com/cd/E21901_01/timesten.1122/e21642/privileges.htm#TTSQL339 https://blogs.msdn.microsoft.com/sqlsecurity/2011/08/25/database-engine-permission-basics/

Roles

Feature ID	114
Feature	Roles
Description	Role-based security, allows you to assign set of permissions to a role, instead of granting them to individual users. This role can then be assigned to group of users. Fixed server and fixed database roles have a fixed set of permissions assigned to them. In Oracle, Single DBA role has database instance wide privileges spanning all schemas. Users with explicit object privileges or those who connect with administrative privileges (SYSDBA) can access objects in the SYS schema. **Predefined Roles** Along with the installation, and creation of an oracle database, Oracle creates many predefined roles: – CONNECT includes the privileges needed to connect to the database. – RESOURCE includes many of the roles a developer might use to create and manage an application, such as creating and altering many types of objects including tables, view, and sequences. – EXP_FULL_DATABASE/IMP_FULL_DATABASE allows the grantee to do logical backups of the database. – RECOVERY_CATALOG_OWNER allows grantee to administer Oracle Recovery Manager catalog. – SCHEDULER_ADMIN allows the grantee to manage the Oracle job scheduler. – DBA gives a user most of the major privileges required to administer a database. These privileges can manage users, security, space, system parameters, and backups. Accessing data dictionary views (v$ views and static dictionary views). exp_full_database, imp_full_database is needed to export objects found in another user's schema. – connect, resource, dba- these might not be created anymore in future versions of Oracle. The `DBA_ROLES` data dictionary view can be used to list all roles of a database and the authentication used for each role.
Category	Security
To Find Feature Enablement	Below query returns all the roles granted to users and other roles: `SELECT count(*) FROM DBA_ROLE_PRIVS;` `USER_ROLE_PRIVS` describes the roles granted to the current user.
Recommendation	**Feature Description:** All versions of SQL Server use role-based security, which allows you to assign permissions to a role, or group of users, instead of to individual users. Fixed server and fixed database roles have a fixed set of permissions assigned to them. SQL Server provides nine fixed server roles.

These roles are security principals that group other principals. Roles are like groups in the Windows operating system.

Server Roles:
Server roles are pre-defined and can't be modified. Nor can you define a new server-wide role. Server roles can be very effective for sharing admin responsibilities among several logins. You don't share the `SA` account password to all logins; rather, you grant the necessary level of admin permissions by adding specific login to a server role. each member of a built-in server role can add other logins to the same role.

Fixed server roles have a fixed set of permissions and server-wide scope. They are intended for use in administering SQL Server and the permissions assigned to them cannot be changed.

Be selective when you add users to fixed server roles. For example, users with `bulkadmin` role can run the BULK INSERT statement, which could jeopardize data integrity.

Fixed SQL Server Roles (8 in total)

Role name	Function
BULKADMIN	BULK INSERT administrators. Can load data into tables using BULK INSERT statement.
DBCREATOR	Alter any database (create, and alter/restore their own). Database creators. Can create and alter, drop and restore databases.
DISKADMIN	Alter resources (manage disk files)
PROCESSADMIN	Process administrators. Can execute KILL statement to disconnect offending sessions.
SECURITYADMIN	Security administrators. Can add and remove logins, add and remove linked servers, Alter any login (grant, deny, revoke server/database permissions, reset passwords).
SERVERADMIN	Server administrators. Can manage server-wide configuration settings and shut down SQL Server service. Members of this role are also allowed to execute `sp_tableoption` system procedure and drop extended procedures.
SETUPADMIN	Setup administrators. Can manage extended stored procedures, linked servers and can mark stored procedures to execute whenever SQL Server starts.

SYSADMIN	System administrators. Can perform any and every activity on server. Members are automatically added to database owner role at creation of every database.

Fixed Database Roles:
Databases too have pre-defined roles that allow role members to perform a certain set of activities within the database. Built-in database roles exist in every database and can't be dropped. At the database level, security is managed by members of the `db_owner` and `db_securityadmin` roles: only members of db_owner can add other users to the db_owner role; db_securityadmin can add users to all other roles except db_owner.

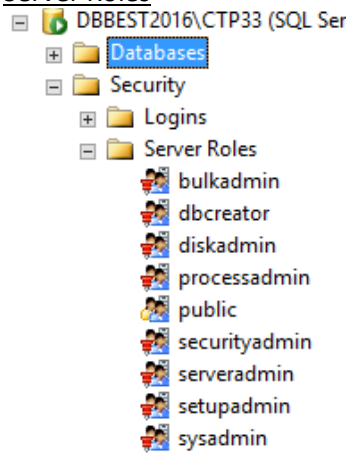
Few Built-in/Fixed SQL Server database roles

Role name	Function
db_owner	Database owner. Users in of `db_owner` role can create, alter or drop the database and perform any other action (read / write / modify objects) within the database. Just as members of SYSADMIN can perform any operation within the server members of db_owner can perform any operation within the database.
public	Built-in role that all logins belong to automatically when they are granted permission to connect to the database. Note that you cannot remove a user from public role. The `public` role is contained in every database including system dbs. It can't be dropped, but you cannot add or remove users from it. Permissions granted to the `public` role are inherited by all other users and roles. Grant `public` only the permissions you want all users to have.
db_securityadmin	Security admins within the database. Members of this role can execute GRANT, REVOKE, DENY statements, add and remove users to roles; add new roles and drop existing roles; change object ownership.
db_accessadmin	Database access administrators can add and remove users from the database, grant and revoke database access from existing users.
db_backupoperator	Members of this role can perform database backups, transaction log backups and can execute CHECKPOINT statement. However, they're not allowed to restore database.
db_datareader	Members of this role can read data from any table in the database

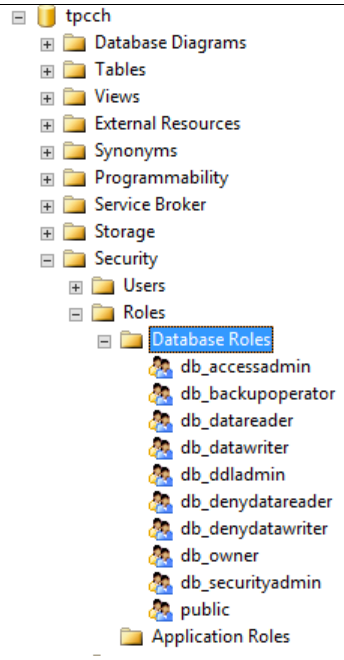
db_datawriter	Data writers can INSERT, UPDATE and DELETE data from any table in the database.
db_ddladmin	DDL administrators can create, alter or drop database objects. Members of this role can set table options, change object ownership, truncate table data, examine index statistics and fragmentation; implement full-text search and reclaim space from a table that was truncated.
db_denydatareader	Members of this role cannot read data from any table in the database.
db_denydatawriter	Members of this role cannot INSERT / UPDATE / DELETE records in any table in the database.

You can view roles in SQL Server via SSMS:

Server Roles



Database Roles

	 User-defined or Application Roles: Users with the CREATE ROLE permission can create new user-defined database roles to represent groups of users with common permissions. User defined roles can be created via T-SQL or SSMS. **Feature Comparison:** Oracle and SQL Server both provide system roles with predefined privileges and user-defined roles.

Migration Approach	SSMA tool doesn't support automatic migration of roles. In SQL Server, Roles can be created manually using T-SQL or SSMA.
	
	In Oracle, roles are available at the instance or server level and can be granted privileges on more than one schema. SQL Server user-defined roles are local to a database and owned by a user. Hence, when migrating a role from Oracle to SQL Server, a role has to be created in each of the databases in which privileges have to be granted.
	
	Two important stored procedures for granting roles in SQL Server
	`sp_addsrvrolemember` can be used for granting fixed system roles and `sp_addrolemember` can be used for granting fixed database roles.
	
	To add a login to a server role
	`EXEC sp_addsrvrolemember 'JohnDoe', 'dbcreator';`
	
	To remove a login from a fixed server role
	`EXEC sp_dropsrvrolemember 'JohnDoe', 'dbcreator';`

	To get server roles list, use `sp_helpsrvrole`.
	To get permissions list each server role has, use `sp_srvrolepermission`.
	
	T-SQL to create user-defined role(s) in SQL Server
	`sp_addrole [ @rolename = ] 'role_name'`
	`[ , [ @ownername = ] 'owner' ]`
	
	To give user read permissions on all tables
	`N'db_datareader` role:
	`EXEC sp_addrolemember N'db_datareader', N'your-user-name'`
	
	To give user all WRITE permissions (INSERT, UPDATE, DELETE) on all tables (use `db_datawriter` role)
	`EXEC sp_addrolemember N'db_datawriter', N'your-user-name'`
	
	The scope of `db_owner` is a database; the scope of `sysadmin` is the whole server.
	
	To add users to a database role
	`exec sp_addrolemember 'db_owner', 'UserName'`
	Users can be assigned to database roles, inheriting any permission sets associated with those roles. `sp_addrolemember` adds a database user, database role, Windows login, or Windows group to a database role in the current database.
	
	To get fixed db roles list
	`sp_helpdbfixedrole`
	
	To get permissions list each database role has
	`sp_dbfixedrolepermission`.

References	https://docs.oracle.com/cd/B10501_01/server.920/a96521/privs.htm https://docs.oracle.com/cd/B19306_01/network.102/b14266/admusers.htm#i1006858 https://msdn.microsoft.com/en-us/library/ms188659.aspx (Server-level roles) https://www.toadworld.com/platforms/sql-server/w/wiki/9764.built-in-server-roles-database-roles http://searchsqlserver.techtarget.com/feature/Create-a-user-defined-server-role-in-SQL-Server-2012-with-T-SQL-SSMS

Data Encryption

Feature ID	10
Feature	Data Encryption
Description	Authentication, authorization, and auditing mechanisms secure data in the database, but not in the operating system data files where data is stored. Oracle introduced Transparent Data Encryption (TDE). TDE provides mechanism to encrypt the data stored in the OS data files. To prevent, unauthorized decryption, TDE stores the encryption keys in a security module outside of the database called Wallet (Keystore in Oracle Database 12c). You can configure Oracle Key Vault as part of the TDE implementation. This enables you to centrally manage TDE keystores (called TDE wallets in Oracle Key Vault) in your enterprise. For example, you can upload a software keystore to Oracle Key Vault and then make the contents of this keystore available to other TDE-enabled databases.
Category	Security
Find Feature Enablement	`SELECT count(*) FROM dba_encrypted_columns;`
Recommendation	**Feature Description:** Encryption is the process of obfuscating data by the use of a key or password. This can make the data useless without the corresponding decryption key or password. Encryption does not solve access control problems. However, it enhances security by limiting data loss even if access controls are bypassed. For example, if the database host computer is misconfigured and a hacker obtains sensitive data, that stolen information might be useless if it is encrypted. You can use encryption in SQL Server for connections, data, and stored procedures. Although encryption is a valuable tool to help ensure security, it should not be considered for all data or connections. Consider how users will access data- If users access data over a public network, data encryption might be required to increase security. However, if all access involves a secure intranet configuration, encryption might not be required. Any use of encryption should also include a maintenance strategy for passwords, keys, and certificates. **Transparent Data Encryption (TDE)** encrypts SQL Server, Azure SQL Database, and Azure SQL Data Warehouse data files, known as encrypting data at rest. TDE performs real-time I/O encryption and decryption of the data and log files. The encryption uses a database encryption key (DEK), which is stored in the database boot record for availability during recovery. The DEK is a symmetric key secured by using a certificate stored in the master database of the server or an asymmetric key protected by an EKM module. TDE protects data "at rest", meaning the data and log files. It provides the ability to comply with many laws, regulations, and guidelines established in various industries. This enables software developers to encrypt data by using AES and 3DES encryption algorithms without changing existing applications. Encryption of the database file is performed

	at the page level. The pages in an encrypted database are encrypted before they are written to disk and decrypted when read into memory. TDE does not increase the size of the encrypted database. Transparent Data Encryption" can help achieve compliancy with Payment Card Industry Data Security Standard. TDE provides strong encryption, but with some shortcomings. First, you must encrypt an entire database. No granularity is offered at a lower level, such as encrypting specific tables or certain data within a table. Second, TDE encrypts only data at rest, in files. Data in memory or in-flight between the application and server are unencrypted. SQL Server 2016 adds a new security feature that helps protect data at rest and in motion, on-premises & cloud: **Always Encrypted** Always Encrypted allows very granular encryption, all the way down to individual columns. Always Encrypted also fully encrypts data at rest, in memory, and in-flight. Always Encrypted allows clients to encrypt sensitive data inside client applications and never reveal the encryption keys to the Database Engine (SQL Database or SQL Server). As a result, Always Encrypted provides a separation between those who own the data (and can view it) and those who manage the data (but should have no access). Always Encrypted enables customers to confidently store sensitive data outside of their direct control. This allows organizations to encrypt data at rest and in use for storage in Azure, to enable delegation of on-premises database administration to third parties, or to reduce security clearance requirements for their own DBA staff. Always Encrypted makes encryption transparent to applications. An Always Encrypted-enabled driver installed on the client computer achieves this by automatically encrypting and decrypting sensitive data in the client application. The driver encrypts the data in sensitive columns before passing the data to the Database Engine, and automatically rewrites queries so that the semantics to the application are preserved. Similarly, the driver transparently decrypts data, stored in encrypted database columns, contained in query results. SQL Server encrypts data with a hierarchical encryption and key management infrastructure. Each layer encrypts the layer below it by using a combination of certificates, asymmetric keys, and symmetric keys. Asymmetric keys and symmetric keys can be stored outside of SQL Server in an Extensible Key Management (EKM) module or external trusted key stores, such as Azure Key Vault, Windows Certificate Store on a client machine, or a hardware security module. **Feature Comparison:**

<table>
<tr><td></td><td>Like Oracle, Encryption at Rest for Data files is supported in SQL Server. Like Oracle, Encryption keys can be stored outside of database in Key Vaults.</td></tr>
<tr><td>Migration Approach</td><td>SSMA does not support migrating encryption configurations.

First, we need to decrypt all the Oracle data; migrate and then set up encryption in SQL Server.

TDE can be set up in SQL Server by using T-SQL to first create master key, certificate, and database encryption key and then enable encryption using T-SQL <code>ALTER DATABASE</code> command.

Configuring Always Encrypted
The initial setup of Always Encrypted in SQL Server involves generating Always Encrypted keys, creating key metadata, configuring encryption properties of selected database columns, and/or encrypting data that may already exist in columns that need to be encrypted.

Please note that some of these tasks are not supported in Transact-SQL and require the use of client-side tools. As Always Encrypted keys and protected sensitive data are never revealed in plaintext to the server, the Database Engine cannot be involved in key provisioning and perform data encryption or decryption operations.

You can use SQL Server Management Studio or PowerShell to accomplish such tasks.

<table>
<tr><th>Task</th><th>SSMS</th><th>PowerShell</th><th>T-SQL</th></tr>
<tr><td>Provisioning column master keys, column encryption keys and encrypted column encryption keys with their corresponding column master keys.</td><td>Yes</td><td>Yes</td><td>No</td></tr>
<tr><td>Creating key metadata in the database.</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Creating new tables with encrypted columns</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Encrypting existing data in selected database columns</td><td>Yes</td><td>Yes</td><td>No</td></tr>
</table>

T-SQL example to enable encryption
<pre>CREATE TABLE [dbo].[Students] (
[StudentID] INT IDENTITY (1, 1) NOT NULL,
[SSN] CHAR (11) COLLATE Latin1_General_BIN2 ENCRYPTED WITH
(COLUMN_ENCRYPTION_KEY = [ColumnEncryptionKey1],
ENCRYPTION_TYPE = Deterministic, ALGORITHM =
'AEAD_AES_256_CBC_HMAC_SHA_256') NOT NULL,
[FirstName] NVARCHAR (50) NULL,
[LastName] NVARCHAR (50) NOT NULL,
[StreetAddress] NVARCHAR (50) NOT NULL,</pre></td></tr>
</table>

	`[City] NVARCHAR (50) NOT NULL,`
	`[ZipCode] CHAR (5) NOT NULL,`
	`[BirthDate] DATE ENCRYPTED WITH (COLUMN_ENCRYPTION_KEY = [ColumnEncryptionKey1], ENCRYPTION_TYPE = Deterministic, ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256') NOT NULL,`
	`CONSTRAINT [PK_dbo.Students] PRIMARY KEY CLUSTERED ([StudentID] ASC)`
	`);`
	`s`
	To access encrypted columns (even if not decrypting them) `VIEW ANY COLUMN` permissions need to be explicitly granted.

References	https://docs.oracle.com/database/121/ASOAG/asotrans.htm#ASOAG10136 (Oracle Transparent Data Encryption) http://www.oracle.com/technetwork/database/security/tde-faq-093689.html https://msdn.microsoft.com/en-us/library/bb510663.aspx (SQL Server Encryption) http://dba.stackexchange.com/questions/137531/in-sql-server-2016-what-is-the-difference-between-always-encrypted-and-transpar http://www.dbta.com/Columns/SQL-Server-Drill-Down/Stronger-Security-Via-Always-Encrypted-in-SQL-Server-2016-106815.aspx

Logins/User Accounts

Feature ID	36
Feature	Login/User Accounts
Description	Oracle provides logins for authorized users to connect to the database. which are referred to as the user or username, and any operation the user can perform is controlled by privileges granted to the login. A user name is database system wide in Oracle, though Oracle 12c pluggable databases can have their own users. ○ In Oracle, users and schemas are essentially same. ○ Consider a user as the account you use to connect to a database, and A schema is a logical container for the database objects (such as tables, views, triggers, and so on) that the user creates. ○ When you create a user, you are also implicitly creating a schema for that user. ○ Schema is owned by a user. A user may be given access to schema objects owned by different Users.
Category	Server
Find Feature Enablement	User accounts can be accessed through a system view called ALL_USERS `SELECT * FROM ALL_USERS;`

Recommendation	**Feature Description:**
	In SQL Server, the privileges at the instance are assigned to the login, and privileges inside a database are given to the related database user. A database user is mapped back to an instance login.
	
	○ In SQL Server, schema and user are separate things. The users are only used to log in and define permissions. One schema is the dbo (or database owner) schema.
	○ In the three-part name 'mydb.dbo.mytable', mydb is a database (physical grouping), while dbo is a schema (logical grouping).
	○ Although the terms login and user are often used interchangeably, they are very different.
	○ A "Login" grants the principal entry into the SERVER.
	  ○ A "User" grants a login entry into a single DATABASE.
	  ○ One "Login" can be associated with many users (one per database).
	○ In SQL Server, DBA add logins to the SQL Server instance, and these logins are mapped to users in individual databases on the SQL Server instance.
	○ Database users who will create tables and feature classes must have privileges necessary to create these objects in the database, and they must have a schema in which they can create them.
	
	Feature Comparison:
	A user name is database system wide in Oracle, but SQL Server uses login IDs to access the instance and user accounts for individual databases.

	Therefore, compared to Oracle; In SQL Server, additionally, a user account must be created in every database that a login needs access to and can be named differently from the login name.

Migration Approach	SSMA doesn't support automatic migration of User Accounts. In SQL Server, use T-SQL to create logins & users and assign permissions.
	
	Below are helpful hints/guidance to migrate Users from Oracle to SQL Server:
	Users of Oracle and SQL Server databases are broadly classified as administrative users, application users, and schema owners.
	• Administrative users are users with special roles, such as database administrator and security administrator.
	• Application users are users who manipulate data in the owning user's tables.
	• Schema owners are users who create and maintain objects related to an application.
	
	The basics for the creation of all the three types of users are the same.
	
	The following query can be run in the source Oracle database to create a list of users that have privileges on any object in a specific schema. The query is constrained to only a specific schema and its users. This aids in situations where only a subset of the schemas and the related users are being migrated:
	`SELECT grantee FROM dba_tab_privs WHERE owner = username`
	`UNION`
	`SELECT grantee FROM dba_col_privs WHERE owner = username;`
	
	The grantee could be a user or a role. Obtain the characteristics of user accounts in Oracle to be migrated:
	`SELECT du.username,`
	`DECODE(du.password,'EXTERNAL','EXTERNAL','DB')`
	`"AUTHENTICATION MODE",`
	`du.default_tablespace, du.temporary_tablespace,`
	`dp.resource_name, dp.limit`
	`FROM dba_users du, dba_profiles dp`
	`WHERE du.profile = dp.profile`
	`AND dp.resource_type = 'PASSWORD'`
	`AND du.username = 'OE';`
	where OE is the name of the user that is being migrated.
	
	Create SQL Server login accounts that provide access to the SQL Server instance, and Create a user account in each of the databases in which the schema's objects have been migrated.
	
	The system stored procedure `sp_grantlogin` is used to create a SQL Server login for a domain-authenticated account. `sp_addlogin` is used

	to create a SQL Server authenticated account. The procedure `sp_grantdbaccess` is used to create user accounts in the individual databases for these logins. User accounts should be created in a database only if there are objects in the database the user needs to access.
	
	T-SQL DDL commands:
	To create login
	`CREATE LOGIN AbolrousHazem WITH PASSWORD = '340$Uuxwp7Mcxo7Khy';`
	
	Following creates a database user for the login created above: `CREATE USER AbolrousHazem FOR LOGIN AbolrousHazem;`
	
	To retrieve all Logins in SQL Server, you can execute the following SQL statement:
	`SELECT * FROM master.sys.sql_logins;`
	
	For a list of SQL Users:
	`SELECT * FROM sys.database_principals`
	
	After user migration is done, make sure to reproduce the privileges they possess in the Oracle database.

References	https://msdn.microsoft.com/en-us/library/aa337545.aspx https://www.techonthenet.com/sql_server/users/create_login.php

Row-Level Security

Feature ID	60
Feature	Row-Level Security
Description	Protect data privacy by ensuring the right access across rows Fine-grained access control over specific rows in a database table Help prevent unauthorized access when multiple users share the same tables, or to implement connection filtering in multitenant applications. Oracle Label Security (OLS) enables you to enforce row-level security for your tables. Hides rows and data depending on user access grants. You can accomplish this by assigning one or more security labels that define the level of security you want for the data rows of the table. You then create a security authorization for users based on the OLS labels. For example, rows that contain highly sensitive data can be assigned a label entitled HIGHLY SENSITIVE; rows that are less sensitive can be labeled as SENSITIVE, and so on. Rows that all users can have access to can be labeled PUBLIC. You can create as many labels as you need, to fit your site's security requirements. In a multitenant environment, the labels apply to the local pluggable database (PDB) and the session labels apply to local users. After you create and assign the labels, you can use Oracle Label Security to assign specific users authorization for specific rows, based on these labels. Afterward, Oracle Label Security automatically compares the label of the data row with the security clearance of the user to determine whether the user is allowed access to the data in the row. You can create Oracle Label Security labels and policies in Enterprise Manager, or you can create them using the SA_SYSDBA, SA_COMPONENTS, and SA_LABEL_ADMIN PL/SQL packages.
Category	Security
To Find Feature Enablement	Check if Oracle Label Security is enabled: `SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Label Security';`
Recommendation	**Feature Description:** In SQL Server, Implement RLS by using the CREATE SECURITY POLICY Transact-SQL statement, and predicates created as inline table valued functions.

	It is highly recommended to create a separate schema for the RLS objects (predicate function and security policy). RLS supports two types of security predicates. FILTER silently filters the rows available to read operations (SELECT, UPDATE, and DELETE). BLOCK explicitly blocks write operations (AFTER INSERT, AFTER UPDATE, BEFORE UPDATE, BEFORE DELETE) that violate the predicate. Access to row-level data in a table is restricted by a security predicate defined as an inline table-valued function. The function is then invoked and enforced by a security policy. For filter predicates, there is no indication to the application that rows have been filtered from the result set; if all rows are filtered, then a null set will be returned. For block predicates, any operations that violate the predicate will fail with an error. Administer via SQL Server Management Studio or SQL Server Data Tools Enforcement logic inside the database and schema bound to the table. **Feature Comparison:** RLS feature is supported by SQL Server as well. The access restriction logic is located in the database tier rather than away from the data in another application tier. The database system applies the access restrictions every time that data access is attempted from any tier.

Migration Approach	SSMA can't migrate Row Level Security directly.
	
	In SQL Server, Row-Level Security can be implemented manually by using the CREATE SECURITY POLICY Transact-SQL statement, and predicates defining filtering criteria created as inline table valued functions.
	
	Step 1: Create a new inline table valued function. The function returns 1 when a row in the SalesRep column is the same as the user executing the query (@SalesRep = USER_NAME()) or if the user executing the query is the Manager user (USER_NAME() = 'Manager').
	
	`CREATE TABLE Sales`
	`(`
	`OrderID int,`
	`SalesRep sysname,`
	`Product varchar(10),`
	`Qty int`
	`);`
	
	`CREATE FUNCTION Security.fn_securitypredicate(@SalesRep AS sysname)`
	`RETURNS TABLE`

	WITH SCHEMABINDING AS RETURN SELECT 1 AS fn_securitypredicate_result WHERE @SalesRep = USER_NAME() OR USER_NAME() = 'Manager'; Step 2: Create a security policy adding the function as a filter predicate. The state must be set to ON to enable the policy. CREATE SECURITY POLICY SalesFilter ADD FILTER PREDICATE Security.fn_securitypredicate(SalesRep) ON dbo.Sales WITH (STATE = ON);
References	https://docs.oracle.com/database/121/TDPSG/GUID-72D524FF-5A86-495A-9D12-14CB13819D42.htm#TDPSG94446 (Enforcing Row-Level Security with Oracle Label Security) https://msdn.microsoft.com/en-us/library/dn765131.aspx (Row-Level Security) https://www.datavail.com/blog/row-level-security-never-got-this-easy-with-sql-server-2016/ http://searchsqlserver.techtarget.com/feature/Put-row-level-security-to-work-in-Azure-SQL-databases

Data Masking

Feature ID	111
Feature	Data Masking
Description	Data masking (also known as data scrambling and data anonymization) is the process of replacing sensitive information copied from production databases to test non-production databases with realistic, but scrubbed, data based on masking rules. To mask data, the Data Masking Pack provides two main features: Masking format library- The format library contains a collection of ready-to-use masking formats. Masking definitions- A masking definition defines a data masking operation to be implemented on one or more tables in a database. Masking definitions associate table columns with formats to use for masking the data.
Category	Security
To Find Feature Enablement	
Recommendation	**Feature Description:** Dynamic data masking limits (DDM) sensitive data exposure by masking it to non-privileged users. It can be used to greatly simplify the design and coding of security in your application. DDM can be configured on the database to hide sensitive data in the result sets of queries over designated database fields, while the data in the database is not changed. Dynamic data masking is easy to use with existing applications, since masking rules are applied in the query results. Many applications can mask sensitive data without modifying existing queries. DDM features full masking and partial masking functions, as well as a random mask for numeric data. Dynamic Data Masking is applied when running SQL Server Import and Export. A database containing masked columns will result in a backup file with masked data (assuming it is exported by a user without UNMASK privileges), and the imported database will contain statically masked data. Dynamic data masking is available in SQL Server 2016 and Azure SQL Database, and is configured by using Transact-SQL commands. **Feature Comparison:** Like Oracle, both full and partial Data Masking is supported in SQL Server.

Migration Approach	SSMA does not support migrating Data Masking directly.
	
	Based on Masking requirements, SQL Server Dynamic Data Masking can be configured manually by using CREATE or ALTER Transact-SQL commands:
	
	`CREATE TABLE Membership`
	`  (MemberID int IDENTITY PRIMARY KEY,`

	<pre> FirstName varchar(100) MASKED WITH (FUNCTION = 'partial(1,"XXXXXXX",0)') NULL,
	   LastName varchar(100) NOT NULL,
	   Phone# varchar(12) MASKED WITH (FUNCTION = 'default()') NULL,
	   Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL);
	
	Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL
	ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()')</pre>
	Use the sys.masked_columns view to query for table-columns that have a masking function applied to them:
	<pre>SELECT c.name, tbl.name as table_name, c.is_masked, c.masking_function
	FROM sys.masked_columns AS c
	JOIN sys.tables AS tbl
	    ON c.[object_id] = tbl.[object_id]
	WHERE is_masked = 1;</pre>
	Dropping a Dynamic Data Mask:
	<pre>ALTER TABLE Membership
	ALTER COLUMN LastName DROP MASKED;
	
	GRANT UNMASK TO TestUser;
	-- Removing the UNMASK permission
	REVOKE UNMASK TO TestUser;</pre>

References	https://docs.oracle.com/database/121/DMKSB/GUID-E3C164DC-0004-4857-A038-54EC7B1A5118.htm (Oracle Data Masking) http://www.oracle.com/technetwork/database/options/data-masking-subsetting/overview/ds-security-dms-2245926.pdf https://msdn.microsoft.com/en-us/library/mt130841.aspx (Dynamic Data Masking)

Case Sensitive Password

Feature ID	Case Sensitive Password
Feature	104
Description	Case sensitive user passwords in Oracle: Oracle by default force case sensitivity of user passwords. The users must provide passwords in the same case (upper, lower or mixed) they created the password with. This behavior is controlled with an initialization parameter SEC_CASE_SENSITIVE_LOGON. By default, it has a value TRUE. Oracle releases before 11g didn't have case sensitivity on password Case sensitive password in Password File in Oracle ignorecase=n is the default with the orapwd command in oracle 11g i.e. you mention it or not it will force the password to be case sensitive when users log in as SYSDBA remotely. To turn off password case sensitivity in password file we need to explicitly mention ignorecase=y while creating the password file.
Category	Security
Find Feature Enablement	`show parameter sec_case_sensitive_logon;`
Recommendation	**Feature Description:** If you selected a case-sensitive collation when you installed SQL Server, your SQL Server login is also case sensitive. Since SQL server is not case sensitive. By default, `SELECT * FROM SomeTable is the same as SeLeCT * frOM soMetaBLe`. **Feature Comparison:** case-sensitive password is not configurable option in SQL Server but can be implemented by applying case-sensitive collation.

Migration Approach	SSMA doesn't support automated migration for case-sensitive migration.
	
	To enable password case-sensitivity, Select a case-sensitive collation when you install SQL Server, your SQL Server login will then become case sensitive.
	
	For case sensitive passwords you need to use a case-sensitive collation:
	`SELECT * FROM dbo.TableName WHERE Password = @ password COLLATE SQL_Latin1_General_CP1_CS_AS;`
	
	`ALTER DATABASE { database_name  | CURRENT }  COLLATE Latin1_General_100_CI_AS;`
	
	The Oracle RDBMS also provides password management functions, such as account locking, password lifetime and expiration, password

	history, and password complexity verification. The SQL Server RDBMS does not provide these services, and Windows security is used to provide these features.
References	http://www.oracleflash.com/37/Oracle-11g-Case-Sensitive-Passwords.html https://www.mindstick.com/blog/360/check-case-sensitive-password-in-sql-server-using-collate-clause http://stackoverflow.com/questions/1411161/sql-server-check-case-sensitivity https://www.webucator.com/how-to/how-check-case-sensitivity-sql-server.cfm

Total Database Size

Feature ID	4
Feature	Total Database size
Description	• DB size = the size of (data files + temp files + online/offline redo log files + control files)- Overall DB size includes used space and free space. • Maximum DB Size in Oracle: There are limits, which vary depending on operating system. Example: if you have 8k bigfile tablespaces and 65,533 files the upper limit is somewhere around 2,047 petabytes!
Category	Performance

Find Feature Enablement	`select`
	`( select sum(bytes)/1024/1024/1024 data_size from sys.dba_data_files ) +`
	`( select nvl(sum(bytes),0)/1024/1024/1024 temp_size from sys.dba_temp_files ) +`
	`( select sum(bytes)/1024/1024/1024 redo_size from sys.v_$log ) +`
	`( select sum(BLOCK_SIZE*FILE_SIZE_BLKS)/1024/1024/1024 controlfile_size from v$controlfile) "Size in GB"`
	`from`
	`dual`

Recommendation	**Feature Description:** SQL Server supports a maximum single database size of nearly 525 petabytes. SQL Server database can be further expanded by either increasing the size of an existing data or log file or by adding a new file to the database. Please follow referred links below for actions recommended for expanding size of database. **Feature Comparison:** Like Oracle, SQL Server size can be expanded.

Migration Approach	SQL Server supports a maximum single DB size of nearly 525 petabytes. If required to migrate bigger data size, it can easily be expanded by adding additional data files. Size is effectively limited only by disk size or windows limitations.
	
	The database is expanded by either increasing the size of an existing data or log file or by adding a new file to the database.
	
	To increase the size of a database using T-SQL:
	`USE master;`
	`GO`
	`ALTER DATABASE AdventureWorks2012`
	`MODIFY FILE`
	`(NAME = test1dat3,`
	`SIZE = 20MB);`
	`GO`
	
	To add data or log files to a database using T-SQL:
	`USE master`
	`GO`
	`ALTER DATABASE AdventureWorks2012`
	`ADD FILEGROUP Test1FG1;`
	`GO`

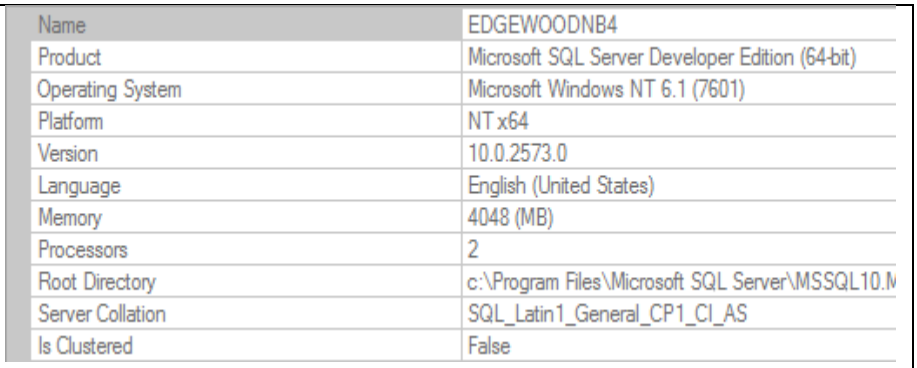
	``
	ALTER DATABASE AdventureWorks2012
	ADD FILE
	(
	    NAME = test1dat3,
	    FILENAME = 'C:\Program Files\Microsoft SQL
	Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\t1dat3.ndf',
	    SIZE = 5MB,
	    MAXSIZE = 100MB,
	    FILEGROWTH = 5MB
	),
	(
	    NAME = test1dat4,
	    FILENAME = 'C:\Program Files\Microsoft SQL
	Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\t1dat4.ndf',
	    SIZE = 5MB,
	    MAXSIZE = 100MB,
	    FILEGROWTH = 5MB
	)
	TO FILEGROUP Test1FG1;
	GO
	``
	
	To increase the size of a database using SQL Server Management Studio:
	1. In Object Explorer, connect to an instance of the SQL Server Database Engine, and then expand that instance.
	2. Expand Databases, right-click the database to increase, and then click Properties.
	3. In Database Properties, select the Files page.
	4. To increase the size of an existing file, increase the value in the Initial Size (MB) column for the file. You must increase the size of the database by at least 1 megabyte.
	5. To increase the size of the database by adding a new file, click Add and then enter the values for the new file. For more information, see Add Data or Log Files to a Database.
	6. Click OK.

References	https://msdn.microsoft.com/en-us/library/ms175890.aspx (Increase the Size of a Database) https://msdn.microsoft.com/en-us/library/ms143432.aspx http://docs.oracle.com/cd/B19306_01/server.102/b14237/limits002.htm#i287915 (physical DB limits) http://docs.oracle.com/cd/B19306_01/server.102/b14237/limits003.htm#i288032 (logical DB limits) http://awads.net/wp/2010/02/15/oracle-database-limits-you-may-not-know-about/ (data type limits)

Oracle Database Version

Feature ID	9
Feature	Oracle Database Version
Description	The version information is retrieved in a table called v$version. It returns detailed version number of the database components.
Category	General
Find Feature Enablement	`SELECT * FROM SYS.PRODUCT_COMPONENT_VERSION;` `SELECT * from V$VERSION;`
Recommendation	**Feature Description:** The most up-to-date version of Microsoft's RDBMS is SQL Server 2016, released in June 2016. Microsoft offers four different editions of SQL Server 2016, plus a web edition for web hosting providers. Express Edition is a lightweight SQL Server database that can support up to 10 GB of data, while Developer Edition is licensed exclusively for development and test environments. The other SQL Server versions include Enterprise, Standard and Web. Enterprise Edition comes with the full suite of features suitable for mission-critical databases and advanced analytics workloads, while Standard Edition comes with a more limited set of features suited to a smaller-scale setup. Web Edition is for use with public websites and is available exclusively to third-party hosting service providers, who set the price.

Migration Approach	You can find version of SQL Server running by T-SQL query:
	`SELECT @@VERSION`
	
	`SQL Server 2016`
	`Microsoft SQL Server 2016 (RTM) - 13.0.1601.5 (X64)`
	`Apr 29 2016 23:23:58`
	`Copyright (c) Microsoft Corporation`
	`Developer Edition (64-bit) on Windows 10 Pro 6.3 <x64>`
	`(Build 14393: )`
	
	In SQL Server Management Studio, right click on the instance name and selecting properties. The "Product version" or "Version" gives you a number of the version that is installed:

	 The first digits refer to the version of SQL Server such as: 8.0 for SQL Server 2000 9.0 for SQL Server 2005 10.0 for SQL Server 2008 10.5 for SQL Server 2008 R2 11.0 for SQL Server 2012 12.0 for SQL Server 2014 13.0 for SQL Server 2016 **SQL Server 2016**: RTM (no SP) 13.0.1601.5; SP1 **13.0.4001.0** or 13.1.4001.0 **SQL Server 2014**: RTM (no SP) 12.0.2000.8; SP1 **12.0.4100.1** or 12.1.4100.1; SP2 **12.0.5000.0** or 12.2.5000.0 **SQL Server 2012** codename Denali: RTM (no SP) 11.0.2100.60; SP1 **11.0.3000.0** or 11.1.3000.0; SP2 **11.0.5058.0** or 11.2.5058.0; SP3 **11.0.6020.0** or 11.3.6020.0
References	https://www.techonthenet.com/oracle/questions/version.php https://www.mssqltips.com/sqlservertip/1140/how-to-tell-what-sql-server-version-you-are-running/ https://support.microsoft.com/en-us/help/321185/how-to-determine-the-version,-edition-and-update-level-of-sql-server-and-its-components http://sqlserverbuilds.blogspot.com/

Set Schema Statement

Feature ID	34
Feature	Set Schema
Description	• A "database" in Oracle typically refers to the complete instance. • You can consider that a user is the account you use to connect to a database, and A schema is a logical container for the database objects (such as tables, views, triggers, and so on) that the user creates. The CREATE USER command automatically creates a schema for same name. A USER owns its SCHEMA. A user may be given access to schema objects owned by different Users as well. • The SET SCHEMA statement sets the default schema for a connection's session to the designated schema. The default schema is used as the target schema for all statements issued from the connection that do not explicitly specify a schema name.
Category	General
Find Feature Enablement	• Query database names o `select * from v$database;` o `select ora_database_name from dual;` • use select instance_name from v$instance; to find out which instance are you currently connected to • TNSNAMES.ora also has the details about which database you are connecting to

Recommendation	**Feature Description:**
	• When a login connects to SQL Server
	o the login is automatically connected to its default database and acquires the security context of a database user.
	o If no database user has been created for the SQL Server login, the login connects as guest.
	o If no default database has been assigned to the login, its default database will be set to master.
	• USE is executed at both compile and execution time and takes effect immediately. Therefore, statements that appear in a batch after the USE statement are executed in the specified database. If the database user does not have CONNECT permission on the database, the USE statement will fail.
	Feature Comparison:
	• As there is typically only a single instance/installation there is no sense in "switching a database" in Oracle. The closest thing to switch the current schema in Oracle is to "USE mydatabase" in SQL Server.
	`Show databases;`
	`Use databaseName;`

Migration Approach	You need to choose how to map the Oracle schemas to the target. In SQL Server, schemas are not necessarily linked to a specific user or a login, and one server contains multiple databases. Using SSMA tool for migration, you can follow one of two typical approaches to schema mapping:

	– By default, in SSMA, every Oracle schema becomes a separate SQL Server database. The target SQL Server schema in each of these databases is set to dbo—the predefined name for the database owner. Use this method if there are few references between Oracle schemas. – Another approach is to map all Oracle schemas to one SQL Server database. In this case, an Oracle schema becomes a SQL Server schema with the same name. To use this method, you change the SSMA default settings. Use this method if different source schemas are deeply linked with each other (for instance if there are cross-references between Oracle tables in different schemas, when trigger is on the table and the tables itself are in different schemas…). SSMA applies the selected schema-mapping method consistently when it converts both database objects and the references to them. A schema is separate entity within the database. It is created by using the CREATE SCHEMA statement. A schema can be owned by a user, a role, or a group (for more information about possible schema owners, see the "Principals" section in this document). A user executing CREATE SCHEMA can be the owner of the schema or it can allocate another user as the schema owner (with appropriate IMPERSONATE permissions). A schema only has one owner, but a user can own many schemas. **Default Schema** Users can be defined with a default schema. The default schema is the first schema that is searched when it resolves the names of objects it references. The default schema for a user can be defined by using the DEFAULT_SCHEMA option of CREATE USER or ALTER USER. If no default schema is defined for a user account, SQL Server will assume dbo is the default schema. It is important note that if the user is authenticated by SQL Server as a member of a group in the Windows operating system, no default schema will be associated with the user. If the user creates an object, a new schema will be created and named the same as the user, and the object will be associated with that user schema.
References	http://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj32268.html https://msdn.microsoft.com/en-us/library/ms188366.aspx http://www.baezaconsulting.com/index.php?option=com_content&view=article&id=46:use-database-command-in-oracle&catid=7:oracle-answers&Itemid=12

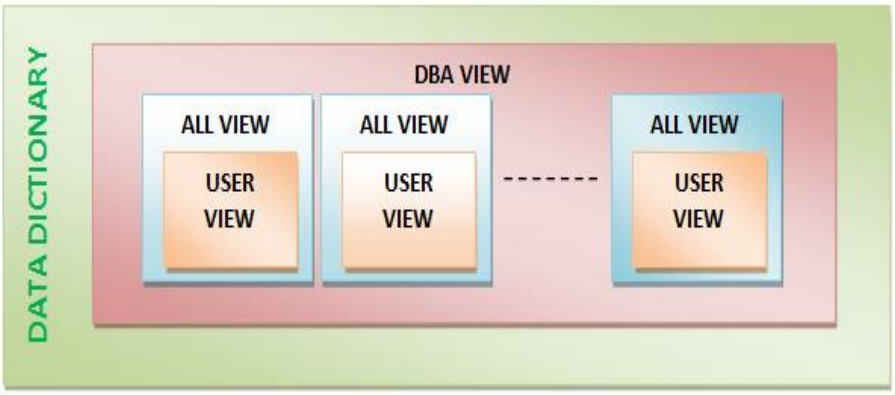
Admin Accounts

Feature ID	113
Feature	Admin Accounts

Description	The SYS user is automatically granted the SYSDBA privilege upon installation. When you log in as user SYS, you must connect to the database as SYSDBA or SYSOPER. Connecting as a SYSDBA user invokes the SYSDBA privilege; connecting as SYSOPER invokes the SYSOPER privilege. EM Express allows you to log in as user SYS and connect as SYSDBA or SYSOPER. SYS or SYSTEM, are Oracle's internal data dictionary accounts. You set the SYS account password upon installation (Windows) or configuration (Linux).
	
	`CONNECT SYSTEM/<password>`
	You set the SYS and SYSTEM account password upon installation (Windows) or configuration (Linux).
	
	`CONNECT SYS/<password> AS SYSDBA`
	To connect as SYSDBA you must supply the SYS user name and password.
	
	`CONNECT / AS SYSDBA`
	The slash (/) indicates that the database should authenticate you with operating system (OS) authentication. when you connect with OS authentication, you are effectively logging in to the database as user SYS. An administrator who is authenticated through OS authentication does not need to know the SYS or SYSTEM account password.
	
	SYSTEM user is also automatically created when Oracle database is installed & is automatically granted the DBA role
	It's used to create additional tables and views that display administrative information. This account can perform all administrative functions except Backup and recovery, and Database upgrade.
	
	While this account can be used to perform day-to-day administrative tasks, Oracle strongly recommends creating named users account for administering the Oracle database to enable monitoring of database activity.
	
	When you connect with the SYSDBA or SYSOPER privilege, you connect with a default schema, not with the schema that is generally associated with your user name. For SYSDBA this schema is SYS; for SYSOPER the schema is PUBLIC. The SYSDBA role is like "root" on Unix or "Administrator" on Windows.
	
	SYSDBA and SYSOPER are administrative privileges to perform high-level admin operations such as creating, starting up, shutting down, backing up, or recovering the database.
	
	The SYSDBA system privilege is for fully empowered database administrators and the SYSOPER system privilege allows a user to perform basic operational tasks, but

	without the ability to look at user data. The SYSDBA and SYSOPER system privileges allow access to a database instance even when the database is not open, and allow to connect to the database instance to start the database.
Category	Admin
Find Feature Enablement	
Recommendation	**Feature Description:** SA is a SQL login administrator account that can be used if mixed authentication is enabled on SQL Server. Like sa, dbo is the most powerful user in a database and no permissions can be denied to him. dbo is a member of the db_owner database role. mapping of dbo is not done at user creation – dbo always exists since the database was created; instead, the login to which dbo maps is determined by what login is the owner of the database – dbo will always map to the login that is marked as the database owner. **Feature Comparison:** Like Oracle, root account for SQL server gets created at time of installation, and is assigned all admin privileges.
Migration Approach	Admin Accounts for SQL Server are set up independently. There is no migration of Admin accounts from Oracle database. The SA account gets automatically created on every new SQL Server installation. But this account is disabled by default if you select Windows Authentication during setup. sa is the primary admin login name, which is, by default, mapped to the dbo user in all the databases. The sa login is hardcoded to be a member of the sysadmin server role. sa/sysadmin pair represent ownership of the server system. "sa" is a SQL Server login, and "BUILTIN\Administrators" is an Integrated Windows Group login. `sa` admin is well-known, so target for hackers. Our recommendation would be to disable `sa` admin created as part of installation; instead create a separate user & assign admin privileges.
References	https://docs.oracle.com/database/121/ADMQS/GUID-2033E766-8FE6-4FBA-97E0-2607B083FA2C.htm#ADMQS12004 https://docs.oracle.com/cd/B25329_01/doc/admin.102/b25107/users_secure.htm#CHDJIEBA https://docs.oracle.com/cd/B28359_01/server.111/b28337/tdpsg_user_accounts.htm#TDPSG20303 https://blogs.msdn.microsoft.com/lcris/2007/09/12/basic-sql-server-security-concepts-permissions-and-special-principals-sa-dbo-guest/ https://www.mssqltips.com/sqlservertip/3695/best-practices-to-secure-the-sql-server-sa-account/

Data Dictionary

<table>
<tr><td>Feature ID</td><td>38</td></tr>
<tr><td>Feature</td><td>Data Dictionary</td></tr>
<tr><td>Description</td><td>The data dictionary is structured in tables and views & store information about the database.
– In Oracle, the data dictionary is stored in the SYSTEM tablespace. The Oracle user SYS owns all base tables and user-accessible views of the data dictionary. When you connect as user SYS, although you have unlimited privileges on data dictionary tables; but you don't modify any data dictionary tables.
– A data dictionary contains:
o The definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on)
o How much space has been allocated for, and is currently used by, the schema objects
o Default values for columns
o Integrity constraint information
o The names of Oracle users
o Privileges and roles each user has been granted
o Auditing information, such as who has accessed or updated various schema objects
– Oracle creates public synonyms for many data dictionary views so users can access them conveniently.
– In Oracle the instance and the database are closely related, there are not system databases like in SQL Server.</td></tr>
<tr><td>Category</td><td>Admin</td></tr>
<tr><td>To Find Feature Enablement</td><td> 

<table>
<tr><td>Data Dictionary View Prefix</td><td>Scope</td></tr>
<tr><td>USER</td><td>User's view (what is in the user's schema)</td></tr>
<tr><td>ALL</td><td>Expanded user's view (what the user can access)</td></tr>
</table></td></tr>
</table>

DBA	Database administrator's view (what is in all users' schemas)

- View Data dictionary: `SELECT * from DICT;`
- this query returns all the objects contained in your schema:

```
SELECT object_name, object_type FROM USER_OBJECTS;
```

- this query returns all the objects to which you have access:

```
SELECT owner, object_name, object_type FROM ALL_OBJECTS;
```

- to query the DBA views, administrators must prefix the view name with its owner, SYS, as in the following:

```
SELECT owner, object_name, object_type FROM SYS.DBA_OBJECTS;
```

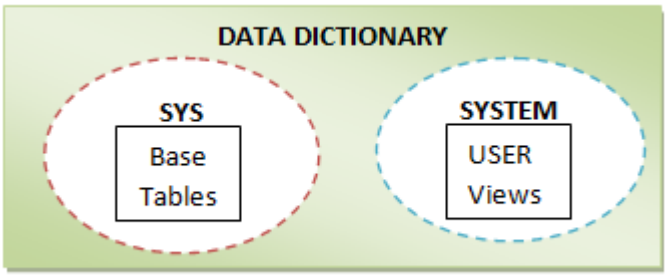
- Oracle recommends that you protect the data dictionary to prevent users that have the ANY system privilege from using those privileges on the data dictionary. To enable data dictionary protection, following initialization parameter set to FALSE (which is default) in the initsid.ora control file: O7_DICTIONARY_ACCESSIBILITY = FALSE. This restricts access to objects in the SYS schema (dictionary objects) to users with the SYS schema. These users are SYS and those who connect as SYSDBA.

```
SELECT * from DICTIONARY;
```

DICT	
TABLE_NAME	**COMMENTS**
USER_RESOURCE_LIMITS	Display Resouce limit to the User
USER_PASSWORD_LIMITS	Displays password limits to the User
USER_CATALOG	All tables, views,synonyms, Sequences owned by the users
ALL_CATALOG	All tables, views,synonyms, Sequences accessible to the users
USER_CLUSTURES	Description of user's own Clustures
ALL_CLUSTERS	Description of clustures accessible to other users
DBA_TABLES	Description of all relational tables in the database
DBA_ALL_TABLES	Description of all object and relational tables in the database
USER_TABLES	Description of the user's own relational tables
ALL_TABLES	Description of relational tables accessible to the user
ALL_INDEXES	Descriptions of indexes on tables accessible to the user
DICT	Synonym for DICTIONARY
COLUMN_PRIVILEGES	Grants on columns for which the user is the grantor, grantee, owner

Recommendation	**Feature Description:** ○ SQL Server System Catalog contain information about all the objects, data types, constraints, configuration options, and resources available to SQL Server. Each database has a system catalog and the structure of the database is defined by this catalog. ○ System catalog is stored in the system tables. All SQL Server system tables are under sys schema so have names prefixed with "sys".

	○ System catalog consists of the following: ○ Catalog View: This is the best way to access system metadata. ○ Backward Compatibility Views: This contains all system tables from previous releases ○ Dynamic Management Views: These enable insight into the current state of the SQL Server system and provide real-time snapshots of internal memory structures indicating the server state. ○ INFORMATION_SCHEMA Views: For SQL-99 standards compliance; allow to view system metadata. ○ In SQL Server, each instance has the system databases which includes: ▪ the master database which stores the system information, ▪ the model database which contains a configuration template for new databases created, ▪ the tempdb database used for temporary storage or temporary results, ▪ the msdb database which contains the SQL Server Agent configuration and ▪ the resource database which contains system objects included in SQL Server. **Feature Comparison:** Similar to Oracle, SQL Server provides system views and table for metadata on database objects.

Migration Approach	SSMA for Oracle V6.0 can convert Oracle system views, which are frequently used. It does not convert columns that are too closely linked with Oracle physical structures or have no equivalent in SQL Server 2014.
	
	Please refer SSMA Guide- Emulating Oracle System Objects page 21
	
	SQL Server's `resource` database contains the metadata for system stored procedures
	`SELECT * FROM sys.columns WHERE object_id = object_id('myTable');`
	
	`SELECT * FROM mydb.INFORMATION_SCHEMA.TABLES;`
	
	 
	
	The following table maps the system tables to their corresponding system views or functions in SQL Server 2016.

System table	System views or functions	Type of view or function
sysaltfiles	sys.master_files	Catalog view
syscacheobjects	sys.dm_exec_cached_plans sys.dm_exec_plan_attributes (Transact-SQL) sys.dm_exec_sql_text sys.dm_exec_cached_plan_dependent_objects	Dynamic management view
syscharsets	sys.syscharsets	Compatibility view
sysconfigures	sys.configurations	Catalog view
syscurconfigs	sys.configurations	Catalog view
sysdatabases	sys.databases	Catalog view
sysdevices	sys.backup_devices	Catalog view
syslanguages	sys.syslanguages	Compatibility view
syslockinfo	sys.dm_tran_locks	Dynamic management view
syslocks	sys.dm_tran_locks	Dynamic management view
syslogins	sys.server_principals sys.sql_logins	Catalog view
sysmessages	sys.messages	Catalog view
sysoledbusers	sys.linked_logins	Catalog view
sysopentapes	sys.dm_io_backup_tapes	Dynamic management view
sysperfinfo	sys.dm_os_performance_counters	Dynamic management view
sysprocesses	sys.dm_exec_connections sys.dm_exec_sessions sys.dm_exec_requests	Dynamic management view
sysremotelogins	sys.remote_logins	Catalog view

	sysservers | sys.servers | Catalog view
References	https://docs.oracle.com/database/121/CNCPT/datadict.htm#CNCPT002 (Oracle 12c Data Dictionary Views) http://www.dummies.com/programming/databases/how-to-use-oracle-12cs-data-dictionary/ https://msdn.microsoft.com/en-us/library/ms345522.aspx (Querying the SQL Server System Catalog FAQ) https://docs.oracle.com/cd/B19306_01/gateways.102/b14270/apc.htm

Diagnostics and Performance views

Feature ID	112
Feature	Diagnostics and Performance views
Description	• When Oracle is running, set of tables is continually updated with current system metrics. Access is limited to DBA users by default These are in-memory virtual tables, read only views used for performance tuning, session monitoring, etc. • The prefix used is V$. Also known as Oracle Dynamic Performance Views (V$ Views). • It shows current state of database. If you are running clusters, you have V$ views, but you also have GV$ views (global views).- you will have Instance_Id for each node in cluster. • Its system generated views, DBA can't change, remove or modify them. These views are used internally by DB or can be used for monitoring by users. SYS owns V$ views. Oracle's V$ objects are actually public synonyms
Category	Admin
To Find Feature Enablement	`select * from v$sql_plan_statistics` `select * from v$session_wait_class`
Recommendation	**Feature Description:** • SQL Server provides Dynamic Management Views (DMVs), and Dynamic management functions (DMFs). They are prefixed with 'dm_'. **Feature Comparison:** • Like Oracle, SQL Server also has system views and functions that give insight into the current state of the system and provide real-time snapshots of internal memory structures indicating the server state.
Migration Approach	Dynamic Management Views and Functions are available in SQL Server, and are enabled by default. There are two types of dynamic management views and functions: Server-scoped dynamic management views and functions. To access them a user requires SELECT permission on object and require VIEW SERVER STATE permission on the server. Database-scoped dynamic management views and functions require VIEW DATABASE STATE permission on the database. • All dynamic management views and functions exist in the sys schema and follow this naming convention dm_*. When you use a dynamic management view or function, you must prefix the name of the view or function by using the sys schema. • Dynamic management views can be referenced in Transact-SQL statements by using two-part, three-part, or four-part names. Dynamic management functions on the other hand can be referenced in Transact-SQL statements by using either two-part or three-part names. Dynamic management views and

functions cannot be referenced in Transact-SQL statements by using one-part names.

- You could Monitor database and instance activity, using DMVs and, monitor performance and scalability using DMFs

```
select * from sys.dm_exec_query_stats
select * from sys.dm_exec_session_wait_stats
```

- Retrieving connection information: Use `sys.dm_exec_connections` to view information about the current connections to SQL Server. It helps you find which processes are currently connected to the instance. The following are the columns commonly used by sys.dm_exec_connections: session_id, most_recent_session_id, connection_time, client_net_address, last_read, last_write, auth_scheme, and most_recent_sql_handle. For example, the following query shows the most recent SQL text executed for each session connected to the SQL Server:

```
SELECT  ec.[session_id]
      ,ec.[connect_time]
      ,ec.[client_net_address]
      ,ec.[last_read]
      ,ec.[last_write]
      ,ec.[auth_scheme]
      ,qt.[text]
FROM [sys].[dm_exec_connections] ec
CROSS APPLY
[sys].[dm_exec_sql_text](ec.[most_recent_sql_handle]) AS qt
```

- Retrieving currently executing query and blocking information. It can be useful to find out what requests are currently executing on SQL Server at any given time. For this you can use the sys.dm_exec_requests dynamic management view. This SQL DMV includes detailed information about the query and query plan, the status of the request and information about the amount of time the query has been executing. The columns you are most likely to use are:
    - blocking_session_id: The Service Profile Identifier of the blocking session
    - wait_type: type of wait
    - wait_time: length of time request has been waiting (in milliseconds)
    - last_wait_type: if a wait has ended, its type is listed here
    - wait_resource: name of resource the request is waiting for
    - lock_timeout: length of time a lock can exist before timing out

The DMV is ideal for troubleshooting blocking, as the following example shows:

```
SELECT  [session_id]
       ,[blocking_session_id]
       ,[status]
       ,[wait_time]
       ,[wait_type]
       ,[wait_resource]
       ,[transaction_id]
       ,[lock_timeout]
FROM [sys].[dm_exec_requests]
WHERE [blocking_session_id] <> 0
```

The following figure shows example results for blocking sessions:

	session_id	blocking_session_id	status	wait_time	wait_type	wait_resource
1	59	58	suspended	430303	LCK_M_S	KEY: 6:72057594045136896 (4
2	60	59	suspended	9554	LCK_M_S	KEY: 6:72057594045136896 (4

References	https://docs.oracle.com/database/121/CNCPT/datadict.htm#CNCPT88897 https://msdn.microsoft.com/en-us/library/ms188754.aspx http://searchsqlserver.techtarget.com/tip/Manage-your-SQL-Server-databases-with-SQL-DMVs

Feature Usage Statistics

Feature ID	109
Feature	Feature Usage Statistics
Description	DBA_FEATURE_USAGE_STATISTICS view is to display information about database feature usage statistics. Some of the information tracked are: ○ Name of the feature ○ # of times the system has detected usage for the feature ○ First sample time the system detected usage of the feature ○ Last sample time the system detected usage of the feat
Category	
To Find Feature Enablement	• Functions: select name c1, detected_usages c2, first_usage_date c3, currently_used c4 from dba_feature_usage_statistics where first_usage_date is not null; • Below sql gives the detail on oracle services being used: `select NAME, VERSION, DETECTED_USAGES, CURRENTLY_USED, FIRST_USAGE_DATE, LAST_USAGE_DATE from DBA_FEATURE_USAGE_STATISTICS where CURRENTLY_USED = 'TRUE' order by 1,2;`
Recommendation	**Feature Description:** Tracking SQL Server object usage can be done with the Audit feature. To track object use with the SQL Server Audit feature, it's necessary to set up the auditing. To do this, an audit object must be created first. This can be done using SQL Server Management Studio or T-SQL. To continue setting up the auditing, it's required to create a database level audit specification. Such database level audit specification will belong to the audit object previously created. Although SQL Server provides a built-in feature (the View Audit Logs context menu option of an audit object) to view captured information, this is not a convenient way for creating comprehensive reports, and it provides basic filtering only. So, in order to provide tracked information for any deeper analysis or documenting, use the `fn_get_file_audit` SQL Server function to read repository `.sqlaudit` files used by the audit object. **Feature Comparison:** Feature usage tracking via System view is NOT available in SQL Server; but tracking database object usage can be done with the Audit feature in SQL Server.
Migration Approach	Feature Usage Statistics support can't be migrated through SSMA tool. **Tracking SQL Server object usage with the Audit feature**

	The database level auditing is available in SQL Server Enterprise and Developer editions only.
	
	To track object use with the SQL Server Audit feature, it's necessary to set up the auditing. In order to do so, an audit object must be created first. This can be done using SQL Server Management Studio or T-SQL
	
	The following T-SQL creates and enables the AuditObjectUsage audit object:
	`USE [master];`
	`GO`
	`CREATE SERVER AUDIT [AuditObjectUsage] TO FILE (`
	`FILEPATH = N'C:\AUDITs\'`
	`, MAXSIZE = 15 MB`
	`, MAX_FILES = 10`
	`, RESERVE_DISK_SPACE = OFF`
	`)`
	`WITH (`
	`QUEUE_DELAY = 1000`
	`, ON_FAILURE = CONTINUE`
	`);`
	`ALTER SERVER AUDIT [AuditObjectUsage]`
	`WITH (STATE = ON);`
	`GO`
	
	With above T-SQL audited info will be stored in maximum 10 files (each 15 MB in size), located in the AUDITs sub-folder on the local drive. This can be modified per requirements. The next step is to set up the auditing in the particular database on specific objects.
	
	To continue setting up the auditing, it's required to create a database level audit specification. Such database level audit specification will belong to the audit object (AuditObjectUsage) we previously created. The following T-SQL creates and enables the database level audit specification:
	`USE [ACMEDBNEW];`
	`GO`
	`CREATE DATABASE AUDIT SPECIFICATION [ObjectUseSpecification]`
	`FOR SERVER AUDIT [AuditObjectUsage]`
	`ADD (DELETE ON OBJECT::dbo.Customers BY [public]),`
	`ADD (INSERT ON OBJECT::dbo.Customers BY [public]),`
	`ADD (SELECT ON OBJECT::dbo.Customers BY [public]),`
	`ADD (UPDATE ON OBJECT::dbo.Customers BY [public]),`
	`ADD (EXECUTE ON OBJECT::dbo.Customers BY [db_owner]),`
	`ADD (EXECUTE ON OBJECT::dbo.Invoices BY [db_securityadmin])`
	`WITH (STATE = ON);`
	`GO`
	
	Within the T-SQL we specified that the Customers table will be audited for particular actions (SELECT/INSERT/UPDATE/DELETE/EXECUTE), while the

	Invoices table will be audited for EXECUTE operations only. Note that it's possible to specify only one object and one principal per event.
	
	Although SQL Server provides a built-in feature (the View Audit Logs context menu option of an audit object) to view captured information, this is not a convenient way for creating comprehensive reports, and it provides basic filtering only.
	
	So, in order to provide tracked information for any deeper analysis or documenting, use the fn_get_file_audit SQL Server function to read repository .sqlaudit files used by the audit object. The following T-SQL script queries the information tracked by the AuditObjectUsage server level audit object:
	`SELECT`
	`event_time AS [Event time],`
	`session_server_principal_name AS [User name] ,`
	`server_instance_name AS [Server name],`
	`database_name AS [Database name],`
	`object_name AS [Audited object],`
	`statement AS [T-SQL statement]`
	`FROM`
	`sys.fn_get_audit_file('C:\AUDITs\AuditObjectUsage*.sqlaudit',`
	`DEFAULT,`
	`DEFAULT);`
	
	The SQL Server Audit feature is native, but when it comes to tracking database level objects, it is supported by SQL Server Enterprise and Developer editions only.
	
	ApexSQL Audit is a compliance tool for SQL Server that features a range of auditing and documenting captured information options via a user-friendly GUI. It helps ensuring SQL Server security and requirements for compliance regulations by tracking changes and access to objects on one or more SQL Server instances

References	https://oracle-base.com/articles/misc/tracking-database-feature-usage http://solutioncenter.apexsql.com/tracking-sql-server-object-usage/ http://dba.stackexchange.com/questions/24294/tracking-object-use-using-sql-auditing Editions and Supported Features for SQL Server 2016 https://msdn.microsoft.com/en-us/library/cc645993.aspx

Oracle Component Installed

Feature ID	35
Feature	Oracle Component Installed
Description	Oracle provides several views (dba_registry and v$option) that display the installed features within the database.
Category	General
To Find Feature Enablement	• `select comp_name, version from dba_registry where status = 'VALID';` • `select parameter from v$option where value = 'TRUE' order by parameter;`
Recommendation	**Feature Description:** The Discovery Report feature is included in the SQL Server Installation Center under Configuration Tools. this feature can be launched from the Start menu. This will produce the report of all discovered versions/components of SQL Server that exist on your machine. The SQL Server Discovery Report is saved to %ProgramFiles%\Microsoft SQL Server\110\Setup Bootstrap\Log\<last Setup Session>\SqlDiscoveryReport.htm. **Feature Comparison:** Like Oracle, SQL Server supports finding out Installed database components.
Migration Approach	SSMA tool doesn't support migrating installed components information. However, In SQL Server, licensing is simple, because every feature and capability is already built into edition itself. There's no extra add-ons to run. SQL Server Discovery Report will produce the report of all discovered versions/components of SQL Server that exist on your machine. The SQL Server discovery report can be used to verify the version of SQL Server and the SQL Server features installed on the computer. The Installed SQL Server features discovery report displays a report of all SQL Server 2000, SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, SQL Server 2014, and SQL Server 2016 products and features that are installed on the local server. The SQL Server features discovery report is available on the Tools page on the SQL Server Installation center. The SQL Server discovery report is saved to `%ProgramFiles%\MicrosoftSQL Server\130\Setup Bootstrap\Log\<last Setup Session>` You can also generate the discovery report through the command line. Run "`Setup.exe /Action=RunDiscovery`" from a command prompt If you add "`/q`" to the command line above no UI will be shown, but the report will still be created in `%ProgramFiles%\MicrosoftSQL Server\130\Setup Bootstrap\Log\<last Setup Session>`

Microsoft SQL Server 2016 Setup Discovery Report

Product	Instance	Instance ID	Feature	Language	Edition	Version	Clustered	Configured
Microsoft SQL Server 2016	MSSQLSERVER	MSSQL13.MSSQLSERVER	Database Engine Services	1033	Enterprise Edition: Core-based Licensing	13.0.1711.0	No	Yes
Microsoft SQL Server 2016	MSSQLSERVER	MSSQL13.MSSQLSERVER	SQL Server Replication	1033	Enterprise Edition: Core-based Licensing	13.0.1711.0	No	Yes
Microsoft SQL Server 2016	MSSQLSERVER	MSSQL13.MSSQLSERVER	Full-Text and Semantic Extractions for Search	1033	Enterprise Edition: Core-based Licensing	13.0.1601.5	No	Yes
Microsoft SQL Server 2016	MSSQLSERVER	MSSQL13.MSSQLSERVER	Data Quality Services	1033	Enterprise Edition: Core-based Licensing	13.0.1601.5	No	Yes
Microsoft SQL Server 2016	MSSQLSERVER	MSAS13.MSSQLSERVER	Analysis Services	1033	Enterprise Edition: Core-based Licensing	13.0.1711.0	No	Yes
Microsoft SQL Server 2016	MSSQLSERVER	MSRS13.MSSQLSERVER	Reporting Services - Native	1033	Enterprise Edition: Core-based Licensing	13.0.1711.0	No	Yes
Microsoft SQL Server 2016	SOFTWARE	MSSQL13.SOFTWARE	Database Engine Services	1033	Express Edition	13.0.1711.0	No	Yes
Microsoft SQL Server 2016			SSMS	1033	Enterprise Evaluation Edition	13.0.15900.1	No	Yes
Microsoft SQL Server 2016			Data Quality Client	1033	Enterprise Edition: Core-based Licensing	13.0.1601.5	No	Yes
Microsoft SQL Server 2016			Integration Services	1033	Enterprise Edition: Core-based Licensing	13.0.1601.5	No	Yes
Microsoft SQL Server 2016			Master Data Services	1033	Enterprise Evaluation Edition	13.0.1601.5	No	Yes

References

http://www.dba-oracle.com/t_list_installed_components.htm

Editions and Supported Features for SQL Server 2016
https://msdn.microsoft.com/en-us/library/cc645993.aspx

https://technet.microsoft.com/en-us/library/bb510455(v=sql.130).aspx

Shut down

Feature ID	18
Feature	Shut Down

Description	To shut down Oracle database and instance, you must first connect as SYSOPER or SYSDBA. modes for shutting down a database:
	• shutting down with the NORMAL clause (default): Oracle will close all sessions, close the database, un-mount the data files and then shut down the instance in two steps, first issuing a "free" the SGA RAM heap and finally, terminating the background processes. Since it waits for all in-flight work to be complete, this could take hours.
	• IMMEDIATE clause: terminates all sessions and does a rollback on all uncommitted transactions
	• TRANSACTIONAL clause,
	• ABORT clause.
	
	Some shutdown modes wait for certain events to occur (such as transactions completing or users disconnecting) before actually bringing down the database. There is a one-hour timeout period for these events
	
	`SQL> shutdown`
	`SQL> shutdown immediate`
	`SQL> shutdown abort`

Category	Admin

Find Feature Enablement	This query shows current instance info- time instance was started, current status, & if any shutdown is pending:
	`SELECT TO_CHAR(STARTUP_TIME,'MON-DD-RR HH24:MI:SS') AS "Inst Start Time",`
	`  SHUTDOWN_PENDING,`
	`  DATABASE_STATUS`
	`FROM V$INSTANCE;`

Recommendation	**Feature Description:** SQL Server is among the most reliable database systems; you may still occasionally need to shut it down or stop SQL Server for a planned maintenance or relocation. SQL Server supports Shutdown in multiple modes via T-SQL or stopping Windows Services with dba privileges. **Feature Comparison:** Like Oracle, SQL Server supports Shutdown in multiple modes with dba privileges.
Migration Approach	SHUTDOWN permissions are assigned to members of the sysadmin and serveradmin fixed server roles, and they are not transferable. SHUTDOWN can be performed by following methods: using T-SQL commands:

	`SHUTDOWN`
	Immediately stops SQL Server. performs an orderly shutdown of the server, with SQL Server checkpointing all databases and flushing all committed data to disk.
	
	`SHUTDOWN WITH NOWAIT`
	Shuts down SQL Server without performing checkpoints in every database. SQL Server exits after attempting to terminate all user processes. When the server restarts, a rollback operation occurs for uncompleted transactions.
	
	using the Windows Services from Control Panel:
	stop the `MSSQLServer` service (or the MSSQL$InstanceName service, if you have a named instance), to stop the instance that you've selected.
	
	using the SQL Server Configuration Manager:
	This issues a checkpoint in all databases. You can flush committed data from the data cache and stop the server.
	
	using command prompt:
	run `net stop mssqlserver` for a default instance,
	
	run `net stop mssql$instancename` for a named instance.
	
	If `sqlservr.exe` was started from the command prompt, pressing CTRL+C shuts down SQL Server. However, pressing CTRL+C does not insert a checkpoint.
	
	Using any of above methods to stop SQL Server sends the `SERVICE_CONTROL_STOP` message to SQL Server.

References	https://docs.oracle.com/cd/B28359_01/server.111/b28310/start003.htm#ADMIN11156 http://www.dba-oracle.com/t_oracle_shutdown_immediate_abort.htm http://sqlmag.com/t-sql/shutting-down-sql-server https://msdn.microsoft.com/en-us/library/ms188767.aspx http://codingsight.com/database-checkpoints-in-sql-server

Constraints

Feature ID	28
Feature	Constraints
Description	○ Constraint is a way of enforcing rules in the database, and it maintains the integrity of the database. ○ Constraints are defined on the columns of a table or table itself to enforce certain business rules. ○ `ALTER TABLE table_name ENABLE CONSTRAINT constraint_name;` ○ `ALTER TABLE table_name DISABLE CONSTRAINT constraint_name;`
Category	SQL
To Find Feature Enablement	○ `select * from all_constraints;` ○ `select * from user_constraints;`
Recommendation	**Feature Comparison:** Both Oracle & SQL Server support same six constraints- ○ PRIMARY KEY ○ UNIQUE ○ FOREIGN KEY ○ CHECK ○ NOT NULL ○ DEFAULT
Migration Approach	SSMA tool performs Constraints Migration as part of converting Oracle Tables to SQL Server Tables. (Refer Feature ID 42 Migration Approach section)
References	http://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj13590.html https://www.techonthenet.com/oracle/check.php https://technet.microsoft.com/en-us/library/ms189862(v=sql.105).aspx

Column-level check constraint

Feature ID	50
Feature	Column-level check constraint
Description	Oracle check constraint insures that updated or inserted values meet a specific condition. The Oracle check constraint check condition must return a TRUE or FALSE, much Like the WHERE clause. If the Oracle check constraint condition returns as TRUE when you use Oracle check constraint, the value is accepted by the constraint. If Oracle check constraint returns the condition as FALSE, the value is rejected. Any column level constraint (exception: not null) can be expressed at the table level - but the opposite is not true. Column Level constraint is checked when the value of the column changed. Oracle check constraint has some limitations. For one, subqueries cannot be used within your Oracle check constraints. Also, an Oracle check constraint is able to reference another column. Sysdate, currval, nextval, level, rowid, uid, user or userenv cannot be referenced with Oracle check constraint. Oracle check constraint does have some limitations in its ability to validate data. If more than one Oracle check constraint is needed, triggers must be implemented.
Category	SQL

Find Feature Enablement	`SELECT constraint_name,`
	`constraint_type,`
	`search_condition`
	`FROM DBA_CONSTRAINTS where constraint_type='C';`

Recommendation	**Feature Description:**
	You don't have to create constraints that only check the values of a single column. You can create constraints that check values in multiple columns at the same time.
	For instance, if I wanted to create a single constraint that checked both the Salary, and SalaryType constraints, I could use the following code:
	`ALTER TABLE dbo.Payroll`
	`WITH NOCHECK ADD CONSTRAINT CK_Payroll_Salary_N_SalaryType CHECK (SalaryType IN ('Hourly','Monthly','Annual')`
	`AND Salary > 10.00`
	`AND Salary < 150000.00);`
	
	While migrating, keep in mind:
	• CHECK constraint causes validation logic overhead. The amount of overhead is determined by its complexity of evaluation (comparisons good; function calls not so much)
	• Disabling unneeded CHECK constraints will reduce the load time.
	• CHECK constraints reject values that evaluate to FALSE. Because null values evaluate to UNKNOWN, their presence in expressions may override a constraint. For example, suppose you place a constraint on an int column specifying that MyColumn can contain only the value 10 (MyColumn=10). If you insert the value NULL into MyColumn, the

	Database Engine inserts NULL and does not return an error. A CHECK constraint returns TRUE when the condition it is checking is not FALSE • CHECK constraints are not validated during DELETE statements. **Feature Comparison:** CHECK constraints are supported in SQL Server as well.
Migration Approach	SSMA tool performs column-level check constraints Migration as part of converting Oracle Tables to SQL Server Tables. (Refer Feature ID 42 Migration Approach section) While migrating, keep in mind: • CHECK constraints reject values that evaluate to FALSE. Because null values evaluate to UNKNOWN, their presence in expressions may override a constraint. For example, suppose you place a constraint on an int column specifying that MyColumn can contain only the value 10 (MyColumn=10). If you insert the value NULL into MyColumn, the Database Engine inserts NULL and does not return an error. A CHECK constraint returns TRUE when the condition it is checking is not FALSE CHECK constraints are not validated during DELETE statements.
References	http://www.databasejournal.com/features/mssql/article.php/3811831/Using-Check-Constraints-to-Validate-Data-in-SQL-Server.htm https://msdn.microsoft.com/en-us/library/ms187550.aspx

Views

<table>
<tr><td>Feature ID</td><td>20</td></tr>
<tr><td>Feature</td><td>Views</td></tr>
<tr><td>Description</td><td>Supported View Types in Oracle & SQL Server:
<table>
<tr><th>Feature</th><th>Oracle</th><th>SQL Server</th></tr>
<tr><td>Simple view</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Join view</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Partitioned view</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Updateable view</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Inline view</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Object view</td><td>Row/statement</td><td>Row</td></tr>
<tr><td>Indexed view</td><td>Yes</td><td>Yes</td></tr>
</table></td></tr>
<tr><td>Category</td><td>SQL</td></tr>
<tr><td>To Find Feature Enablement</td><td>for all views (you need dba privileges for this query):
<code>select view_name from sys.dba_views</code>

for all accessible views (accessible by logged user):
<code>select view_name from sys.all_views</code>

for views owned by logged user:
<code>select view_name from sys.user_views</code>

<code>SELECT view_name, owner
FROM sys.all_views
ORDER BY owner, view_name</code></td></tr>
<tr><td>Recommendation</td><td>Feature Comparison:
○ Both Oracle and SQL Server offer views based on simple queries involving a single table and complex queries based on multiple tables.
○ Indexed views in SQL Server are the only type of view that actually stores data, similar to materialized views in Oracle. (Please refer Feature ID 82 for details on Materialized View) Everything else is run on top of the tables.
○ Oracle and SQL Server offer updatable views with INSTEAD OF triggers and WITH CHECK OPTION constraints.</td></tr>
<tr><td>Migration Approach</td><td>SSMA for Oracle V6.0 converts Oracle system objects including views It does not convert columns that are too closely linked with Oracle physical structures or have no equivalent in SQL Server 2014. The following views can be migrated automatically to SQL Server views:
• ALL_INDEXES
• DBA_INDEXES
• ALL_OBJECTS
• DBA_OBJECTS</td></tr>
</table>

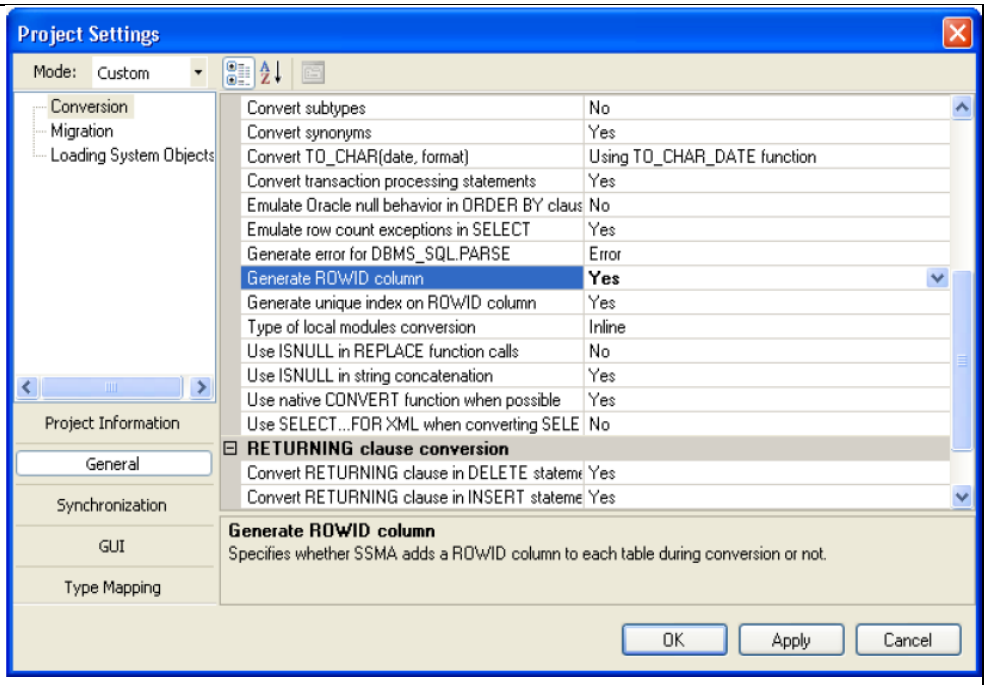
	• ALL_SYNONYMS • DBA_SYNONYMS • ALL_TAB_COLUMNS • DBA_TAB_COLUMNS • ALL_TABLES • DBA_TABLES • ALL_CONSTRAINTS • DBA_ CONSTRAINTS • ALL_SEQUENCES • DBA_SEQUENCES • ALL_VIEWS • DBA_VIEWS • ALL_USERS • DBA _USERS • ALL_SOURCE • DBA_SOURCE • GLOBAL_NAME • ALL_JOBS • DBA_ JOBS • V$SESSION There are ways to manually convert the following views: (Please Refer SSMA Migration Guide V6.0 Page 21) • ALL_EXTENTS • V$LOCKED_OBJECT • DBA_FREE_SPACE DBA_SEGMENTS
References	http://www.dba-oracle.com/concepts/views.htm http://www.oratable.com/oracle-views-features/ http://www.dotnettricks.com/learn/sqlserver/different-types-of-sql-server-views https://www.mssqltips.com/sql-server-tip-category/50/views/

Triggers

Feature ID	37
Feature	Triggers
Description	A trigger is an exceptional sort of stored procedure which functions when we try to amend the data in a table like inserting, deleting or updating data. It is a database object, executed automatically and is bound to a table. Trigger types supported in both Oracle and SQL Server: ○ DML – Insert ○ DML – Update ○ DML – Delete ○ Timing – Before ○ Timing – After ○ Level ○ Views – INSTEAD OF ○ Multiple triggers per actions ○ DDL triggers ○ Login triggers ○ Single trigger for multiple actions
Category	SQL

Find Feature Enablement	`select * from all_triggers;`
	`select * from DBA_TRIGGERS`
	`select * from USER_TRIGGERS`

Recommendation	**Feature Comparison:** ○ Note that SQL Server does not have an exact equivalent of Oracle's 'Before' trigger. During migrations, Oracle's 'Before' trigger is replaced by SQL Server's "Instead Of" trigger. ○ Also note that many triggers in Oracle databases assign only default values – this is not efficient and overcomplicates the implementation. For default values use SQL Server's built-in database default value capability. Don't use (or continue to use) triggers to assign default values.
Migration Approach	SSMA tool performs Trigger Migration as part of converting Oracle Tables to SQL Server Tables. Refer Feature ID 42 Migration Approach section, Also Refer Migrating Oracle Triggers section in SSMA Guide for specific detailed information on migrating different types of Trigger. Using SSMA, you can migrate Oracle Row-level triggers, if SSMA generates a special ROWID column for the SQL Server table. Therefore, if you are converting tables with UPDATE triggers, we recommend setting the Generate ROWID column option to Yes or Add ROWID column for tables with triggers in the SSMA project settings (See Figure below). To emulate row-level triggers, SSMA processes each row in a cursor loop.



Since, SQL Server does not have an exact equivalent of Oracle's 'Before' trigger. To emulate this in SQL Server, you must create INSTEAD OF triggers. That means you must incorporate the triggering statement into the target trigger's body. Because multiple rows can be affected, SSMA puts the statement in a separate cursor loop.

In some cases, you cannot convert Oracle triggers to SQL Server triggers with one-to-one correspondence. If an Oracle trigger is defined for several events at once (for example, INSERT or UPDATE), you must create two separate target triggers, one for INSERT and one for UPDATE. In addition, because SQL Server supports only one INSTEAD OF trigger per table, SSMA combines the logic of all BEFORE triggers on that table into a single target trigger. This means that triggers are not converted independently of each other; SSMA takes the entire set of triggers belonging to a table and converts them into another set of SQL Server triggers so that the general relation is many-to-many.

In brief, the conversion rules are:

- BEFORE triggers for a table are converted into one INSTEAD OF trigger.
- AFTER triggers remain AFTER triggers in SQL Server.
- INSTEAD OF triggers are converted to INSTEAD OF triggers. Multiple INSTEAD OF triggers defined on the same operation are combined into one trigger.
- Row-level triggers are emulated using cursors.
- Cascading triggers are converted into multiple individual triggers.

Triggers that are defined for multiple events are split into separate target triggers.

References	https://technet.microsoft.com/en-us/library/ms179288(v=sql.105).aspx

	http://www.way2tutorial.com/plsql/plsql_triggers.php http://blog.sqlauthority.com/2013/01/24/sql-server-use-instead-trigger/

Indexes

<table>
<tr><td>Feature ID</td><td>21</td></tr>
<tr><td>Feature</td><td>Indexes</td></tr>
<tr><td>Description</td><td>Oracle uses indexes for query performance.</td></tr>
<tr><td>Category</td><td>SQL</td></tr>
<tr><td>To Find Feature Enablement</td><td><code>select dbms_metadata.get_ddl('INDEX', index_name, owner) from all_indexes;</code></td></tr>
<tr><td>Recommendation</td><td>Feature Description:
○ During migration, make sure you understand the index capabilities of SQL Server and use those capabilities.
○ Before migrating indexes it is important to ensure that they are actually used. Do not migrate unused indexes as this can result in a lot of unnecessary processing.
○ Also note that since SQL is more “set-based” than Oracle, it may need fewer indexes due to advanced query optimizations.
○ We recommend that you use the index tuning wizard after migration to recommend new indexes. Do not run this wizard during peak load times, however.
○ Oracle’s invisible indexes are maintained like any other index, but they are ignored by the optimizer unless the OPTIMIZER_USE_INVISIBLE_INDEXES parameter is set to TRUE at the instance or session level.
○ Disabled indexes in SQL Server prevent user access to the index, and for clustered indexes, to the underlying table data. The index definition remains in metadata and index statistics are kept on nonclustered indexes. Disabling a clustered index on a table prevents access to the data; the data still remains in the table, but is unavailable for DML operations until the index is dropped or rebuilt.
○ Clustered Columnstore indexes in the SQL Server Database Engine, with improved data compression, can be used to significantly speed-up the processing time of common data warehousing workloads that primarily perform bulk loads and read-only queries. If most of your queries are small lookup queries, seeking into a B-tree index may be faster and you may not find a columnstore index to be beneficial. If you test a columnstore index and it does not benefit your workload, you can drop or disable the index.

Feature Comparison:
Following table highlights comparative support:
<table>
<tr><th>Feature</th><th>Oracle</th><th>SQL Server</th></tr>
<tr><td>B-tree unique</td><td>Yes</td><td>Yes</td></tr>
<tr><td>B-tree non-unique</td><td>Yes</td><td>Yes</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>B-tree composite</td><td>Yes (32 cols)</td><td>Yes (16 cols)</td></tr>
<tr><td>B-tree ascending</td><td>Yes</td><td>Yes</td></tr>
<tr><td>B-tree cluster</td><td>Yes</td><td>Yes</td></tr>
<tr><td>B-tree reverse key</td><td>Yes</td><td>No</td></tr>
<tr><td>B-tree key compressed</td><td>Yes</td><td>No</td></tr>
<tr><td>B-tree function-based</td><td>Yes</td><td>No</td></tr>
<tr><td>B-tree index organized table</td><td>Yes</td><td>Yes (clustered)</td></tr>
<tr><td>B-tree partitioned</td><td>Yes</td><td>No</td></tr>
<tr><td>Bitmap</td><td>Yes</td><td>No</td></tr>
<tr><td>Bitmap join</td><td>Yes</td><td>No</td></tr>
<tr><td>Columnstore index</td><td>No</td><td>Yes</td></tr>
<tr><td>In-Memory OLTP table index</td><td>No</td><td>Yes</td></tr>
<tr><td>Invisible index</td><td>Yes</td><td>No*</td></tr>
</table>
</td></tr>
<tr><td>Migration Approach</td><td></td></tr>
<tr><td>References</td><td>https://www.techonthenet.com/oracle/indexes.php

http://viralpatel.net/blogs/invisible-indexes-in-oracle-11g/

https://www.simple-talk.com/sql/performance/14-sql-server-indexing-questions-you-were-too-shy-to-ask/

https://www.brentozar.com/sql/index-all-about-sql-server-indexes/

http://logicalread.solarwinds.com/sql-server-2016-columnstore-pd01/#.WCcA_vorKM8</td></tr>
</table>

Trace Files

Feature ID	26
Feature	Trace Files

Description	• Trace File are trace (or dump) file that Oracle Database creates to help you diagnose and troubleshoot operating problems, and are a useful mechanism for developers and DBAs to performance tune applications. With trace data, you can track the execution of a given set of SQL statements of a session.
	• Each server and background process writes to a trace file. When a process detects an internal error, it writes information about the error to its trace file.
	• The file name format of a trace file is sid_processname_unixpid.trc.
	• The following is a sample trace file name: `$ORACLE_BASE/diag/rdbms/mydb/mydb/trace/test_lgwr_1237.trc` where ORACLE_BASE is the Oracle Base Directory. For security reasons and to prevent unauthorized access to the ORACLE_HOME area, the trace file area (identified by the USER_DUMP_DEST init.ora parameter) is separate from the ORACLE_HOME area.
	• MAX_DUMP_FILE initialization parameter is used to set the size of trace file.
	• Use SQL Developer to see a formatted display of a SQL Trace file (*.trc file) in Oracle. You can also use The Oracle Enterprise Manager to monitor the active sessions, with the the query that are being executed, its execution plan, locks, some statistics and even a progress bar for the longer tasks.
	• To gather trace at your own session, you can issue the following commands:
	`ALTER SESSION SET SQL_TRACE TRUE;`
	`ALTER SESSION SET timed_statistics = TRUE;`
	`ALTER SESSION SET statistics_level=ALL;`
	`ALTER SESSION SET EVENTS '10046 trace name context forever,level 12';`

Category	Admin

To Find Feature Enablement	To find the trace file for your current session:
	`SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Default Trace File';`
	
	The full path to the trace file is returned.
	
	To find all trace files for the current instance:
	`SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Diag Trace';`
	
	The path to the ADR trace directory for the current instance is returned.

	To determine the trace file for each Oracle Database process, submit the following query: `SELECT PID, PROGRAM, TRACEFILE FROM V$PROCESS;`
Recommendation	**Feature Description:** In SQL Server, use Profiler tool in SQL Server to view trace files. For best practices, to use SQL Server Profiler, pls refer below link. **Feature Comparison:** Similar to Oracle, Tracing is available in SQL Server as well. Tracing can be disabled or enabled as needed.

Migration Approach	Tracing configurations can't be migrated directly via SSMA.
	
	In SQL Server, tracing can be configured easily. Use the **default trace enabled** option in SQL Server to enable or disable the default trace log files.
	
	Default Trace in SQL Server can be enabled or disabled using the sp_configure system stored procedure.
	Set the 'default trace enabled' advanced option to 1 (which is default setting for this option) to enable the default trace or set it to 0 to disable the default trace.
	To change the settings for any of the advanced options (like 'default trace enabled' option) using the sp_configure system stored procedure, 'show advanced options' must be set to 1.
	
	Default location of trace files: `C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\LOG\log.trc`
	
	Use the default trace enabled option in SQL Server to enable or disable the default trace log files.
	
	To enable the default trace
	`EXEC sp_configure 'show advanced options', 1;`
	`GO`
	`RECONFIGURE;`
	`GO`
	`EXEC sp_configure 'default trace enabled', 1;`
	`GO`
	`RECONFIGURE;`
	`GO`
	`EXEC sp_configure 'show advanced options', 0;`
	`GO`
	`RECONFIGURE;`
	`GO`
	
	To disable the default trace
	`EXEC sp_configure 'show advanced options', 1;`
	`GO`
	`RECONFIGURE;`
	`GO`
	`EXEC sp_configure 'default trace enabled', 0;`

	GO RECONFIGURE; GO EXEC sp_configure 'show advanced options', 0; GO RECONFIGURE; GO **To check whether the default trace is ON (1), or OFF (0)** EXEC sp_configure 'show advanced options', 1; GO RECONFIGURE; GO EXEC sp_configure 'default trace enabled'; GO EXEC sp_configure 'show advanced options', 0; GO RECONFIGURE; GO **To get information for all traces in the instance of SQL Server** SELECT * FROM :: fn_trace_getinfo(default) This will give you a list of all of the traces that are running on the server. The property of the trace as represented by the following integers: 1 – Trace Options (@options in sp_trace_create) 2 – FileName 3 – MaxSize 4 – StopTime 5 – Current Trace status (1 = On and 0 = Off)
References	https://oracle-base.com/articles/misc/sql-trace-10046-trcsess-and-tkprof https://msdn.microsoft.com/en-us/library/ms187929.aspx https://www.mssqltips.com/sqlservertutorial/3501/sql-server-profiler-best-practices/

Tables

Feature ID	42
Feature	Tables
Description	Tables are the basic unit of data storage in an Oracle Database. Data is stored in rows and columns. You define a table with a table name, such as employees, and a set of columns. You give each column a column name, such as employee_id, last_name, and job_id; a datatype, such as VARCHAR2, DATE, or NUMBER; and a width. The width can be predetermined by the datatype, as in DATE. If columns are of the NUMBER datatype, define precision and scale instead of width. A row is a collection of column information corresponding to a single record. You can specify rules for each column of a table. These rules are called integrity constraints. One example is a NOT NULL integrity constraint. This constraint forces the column to contain a value in every row.
Category	Platform
To Find Feature Enablement	`Select * from dba_tables where owner in ('select user from dual');`

Feature Usage	`To list all tables owned by the current user, type`
	`Select tablespace_name, table_name from user_tables;`
	
	`To list all tables in a database`
	`Select tablespace_name, table_name from dba_tables;`
	
	`To list all tables accessible to the current user, type:`
	`Select tablespace_name, table_name from all_tables;`
	
	`Relational Tables:`
	`Select * FROM all_tables`
	
	`Temporary Tables`
	`Select * from dba_tables where temporary='Y'`
	`Select * from SYS.TTABLES`
	`Select * from SYS.TTBL_STATS`

Recommendation	**Feature Description -** In SQL Server Tables are database objects that contain all the data in a database. In tables, data is logically organized in a row-and-column format like a spreadsheet. Each row represents a unique record, and each column represents a field in the record • The number of tables in a database is limited only by the number of objects allowed in a database (2,147,483,647). • You can assign properties to the table and to each column in the table to control the data that is allowed and other properties. • The data in the table can be compressed either by row or by page. Data compression can allow more rows to be stored on a page. **Feature Comparison -**

Below tables lists the tables present in Oracle and SQL Server.

Oracle	SQL Server
Heap-organized table	Heap
Clustered table	Indexed views
Partition table	Partitioned table
Temporary table	Temporary table
External table	Linked server
Object table	Table type
Index-organized table	Clustered index
N/A	In-Memory OLTP table
Hybrid columnar compressed table	Columnstore clustered index

Migration Approach

In SSMA, each Oracle table is converted to a SQL Server table. During the conversion, all indexes, constraints, and triggers defined for a table are also converted. When determining the target table's structure, SSMA uses type mapping definitions.

Below are the steps to migrate your Oracle Schema to SQL Server

- Start SSMA
- **Change Default Project Setting** - By default, SSMA loads only basic Oracle system schemas and packages. You need to customize project settings to allow loading of the schema you want to migrate. Click on Tools from the menu and select Default Project Settings. On the Default Project Setting menu, click on Loading System Objects and check **'<Schema Name You Want To Migrate>'** system object.
- **Create a new project**- Once the program is running, click on the **New Project** icon in the upper left corner to get started. Specify the name of the project and the location of the file to save the project information:
- **Connect to Oracle -** Click on the **Connect to Oracle** icon from the menu toolbar and provide connection information to your Oracle database.
- **Create a schema migration report** - Select the schema, then right-click the schema then select ***Create Report***:
- **Connect to SQL Server -** Click on *the Connect to SQL Server* icon from the File Menu. Specify the server name (e.g. *localhost* if SSMA is running on the SQL server machine) and port number (if using other than default 1433 SQL Server port number). Type the name of the database you are migrating to (e.g. **HR**). If the database does not exist, SSMA will create a new database using the default setting. Specify authentication information and click *Connect* to continue.
- **Map Schema and Type -** In the **Oracle Metadata Explorer**, check the schema and expand. You can select (or deselect) objects to

be migrated as well as map schema. Schema mapping can be done at the Oracle schema level or at the individual object (such as specific table in Oracle) to SQL Server schema.

- **Convert the schema -** In the **Oracle Metadata Explorer**, right-click the **schema** and select ***Convert Schema***:
- **Review conversion report and resolve error as necessary**.
- **Synchronize the SQL Server database**. To deploy the changes to the SQL server, right-click the database in the SQL Server metadata explorer and select **Synchronize with Database**.
- **Migrate the data**. From Oracle Metadata Explorer window, right-click on the schema and select ***Migrate Data***. Provide connection information to both the Oracle source database and the target SQL server.
- **Review Migration Report.** After the data is migrated, a report will be displayed with migration statistics

Migrating tables to Memory Optimized Tables

- SQL Server 2014 introduced In-Memory OLTP database concept which improves OLTP database performance.
- The In-Memory OLTP feature includes memory-optimized tables, table types and native compilation of stored procedures for efficient access to these tables.
- AS memory-optimized tables reside in memory, rows in the table are read from and written to memory.
- A second copy of the table data is maintained on disk, but only for durability purposes. Each row in the table potentially has multiple versions. This row versioning is used to allow concurrent reads and writes on the same row.

DDL syntax for creating memory-optimized table is as follows:

```
CREATE TABLE database_name.schema_name.table_name
(
column_name data_type
[COLLATE collation_name] [NOT] NULL
[DEFAULT constant_expression]
[IDENTITY]
[PRIMARY KEY NONCLUSTERED [HASH WITH (BUCKET_COUNT =
bucket_count)]]
[INDEX index_name
[NONCLUSTERED [HASH WITH (BUCKET_COUNT = bucket_count)]]]
[,…]
[PRIMARY KEY
{
NONCLUSTERED HASH (column [,…]) WITH (BUCKET_COUNT = bucket_count) |
NONCLUSTERED (column [ASC|DESC] [,…] ) }
}]
[INDEX index_name
{
```

NONCLUSTERED HASH (column [,…]) WITH (BUCKET_COUNT = bucket_count) | NONCLUSTERED (column [ASC|DESC] [,…]) }
}] [,…]
)
WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);

Restrinctions for conversion to memory-optimized tables:

- When converting an Oracle table that uses sequence to SQL Server table using IDENTITY property, it must be created only with SEED equal to 1 and INCREMENT equal to 1.
- If this condition is not met, SSMA generates a warning message like this: “Cannot create identity with seed 1000 and increment 5 for Memory optimized table. Allowed only Identity(1,1)”. There are two ways to solve this issue.

First one is to convert the table with IDENTITY (1, 1) and add a corresponding seed to the identity column value and multiplying this value into the corresponding increment. For example, if Oracle sequence has seed value equal to 10 and increment value equal to 2:

SQL Server

```
CREATE TABLE imt(
id INT NOT NULL IDENTITY(1,1) PRIMARY KEY NONCLUSTERED,
name VARCHAR(50) NOT NULL)
WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA)
SELECT 10 + (id – 1) * 2
FROM imt;
```

The second way is to use SQL Server SEQUENCE objects instead of IDENTITY property when inserting new records:

SQL Server

```
CREATE TABLE imt(
id INT NOT NULL PRIMARY KEY NONCLUSTERED,
name VARCHAR(50) NOT NULL)
WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);
GO
CREATE SEQUENCE imt_seq AS INT START WITH 10 INCREMENT BY 2
GO
INSERT INTO imt(id, name)
SELECT NEXT VALUE FOR imt_seq, 'New Name';
```

The next restriction is that uniqueidentifier column default is not supported for memory-optimized tables. Besides, column defaults support only constant expressions. SSMA issues warning about that and removes the column default. A workaround for this can be defining the column that uses uniqueidentifier default as varchar column that can contain at least 36 characters (this is the

	length of uniqueidentifier value in SQL Server). Insert the value to this column explicitly every time when inserts to the table are performed: *Oracle* CREATE TABLE IMT (ID RAW(32) DEFAULT sys_guid(), NAME VARCHAR2(50)); *SQL Server* CREATE TABLE [dbo].[IMT] ([ID] varchar(36) NULL, [NAME] nvarchar(50) NULL, [PKCol] int IDENTITY(1, 1) NOT NULL, PRIMARY KEY NONCLUSTERED ([PKCol] ASC)) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA) GO Conversion to memory-optimized tables is not supported on Azure SQL DB.
Performance Recommendation	• Create user tables on a non-primary filegroup; reserve the primary file group for system objects. This way the system supplied and user-defined objects do not compete for disk resources • Create commonly accessed tables on the same filegroup. You can expect performance benefits if the data of commonly joined tables resides on the same disk. • Create a clustered index on every table. Each table can only have a single clustered index. If a table has a clustered index, its data is physically sorted as per the clustered index key. Clustered indexes in SQL Server have numerous benefits. For example, if you retrieve data from a table using an ORDER BY clause referencing the clustered index key, the data does not need to be sorted at query execution time. • Ensure the clustered index is built on a column that contains distinct values in each row. This makes the clustered index also a unique index. If the clustered index key(s) contains non-unique values, SQL Server will add a hidden column to your table to make clustered index keys unique. • In addition to the clustered index, create non-clustered indexes, particularly on those columns used for joining the table to other tables or for filtering the data set to be retrieved. • If you have numerous "lookup" tables with very few rows in each, consider combining them into a single "master lookup" table. For example, you could have numerous "type_lookup" and "category_lookup" tables, each with a dozen of rows. Instead of having

	to maintain 30 different lookup tables, you can combine them in a single table that has row_identifier, row_type and row_value columns. • If a table contains millions of rows and you have multiple disks (or disk arrays) at your disposal, take advantage of table and index partitioning. Partitioning can provide considerable query performance improvements. It can also make loading and purging large data sets from a table very fast. • If the table is partitioned, make sure its indexes are aligned; this means indexes are using the same partitioning scheme as the table.
References	https://docs.oracle.com/database/121/CNCPT/tablecls.htm#GUID-F845B1A7-71E3-4312-B66D-BC16C198ECE5 https://msdn.microsoft.com/en-us/library/ms189084.aspx https://www.dbbest.com/blog/ssma-oracle-7-0-whats-new-using-memory-oltp-sql-server-2016/

Cluster

Feature ID	49
Feature	Cluster
Description	A cluster provides an optional method of storing table data. A cluster is made up of a group of tables that share the same data blocks. The tables are grouped together because they share common columns and are often used together. Because clusters store related rows of different tables together in the same data blocks, properly used clusters offer two primary benefits: • Disk I/O is reduced and access time improves for joins of clustered tables. • The cluster key is the column, or group of columns, that the clustered tables have in common. You specify the columns of the cluster key when creating the cluster. You subsequently specify the same columns when creating every table added to the cluster. Each cluster key value is stored only once each in the cluster and the cluster index, no matter how many rows of different tables contain the value.
Category	Platform
To Find Feature Enablement	`Select count(*) from DBA_CLUSTERS`

Feature Usage	`Select * from USER_CLUSTERS`
	
	`Select * from DBA_CLUSTERS`
	
	`Select * from ALL_CLUSTERS`
	
	`Select * from V_$ACTIVE_INSTANCES`

Recommendation	Feature Description: • To provide high availability for your SQL Server databases, it is recommended that you store them on server clusters using Cluster Service. • Servers configured using Cluster Service share common data and work as a single system. • Each server can operate independently of other servers in the cluster. So, if one server fails, another server takes over its functions. • If the current server is a clustered server, fn_virtualservernodes returns a list of failover clustered instance nodes on which this instance of SQL Server has been defined. If the current server instance is not a clustered server, fn_virtualservernodes returns an empty rowset. • As part of the SQL Server Always On offering, Always On Failover Cluster Instances leverages Windows Server Failover Clustering (WSFC) functionality to provide local high availability through redundancy at the server-instance level—a failover cluster instance (FCI). An FCI is a single instance of SQL Server that is installed across Windows Server Failover Clustering (WSFC) nodes and, possibly, across multiple subnets. On the network, an FCI appears to be an instance of SQL Server running on a

	single computer, but the FCI provides failover from one WSFC node to another if the current node becomes unavailable. Feature Comparison: • Oracle Clusters do not have an equivalent in SQL Server. • The tables in the cluster must be created as regular heaps. • If rows are frequently accessed using a range search, a clustered index may be created on such column(s). Only one clustered index can be created on a table. • This migration of clustered tables in Oracle to heaps in SQL Server will be transparent to the application and users. • Always On Failover Cluster Instance and Mirroring can be used as a workaround for Oracle clusters.
Migration Approach	As migrating from Oracle Clusters to SQL Server Clustering is not a straight forward migration, we need to study the clusters present on the Oracle applications and accordingly we can create Clusters using the workaround of Always On Failover Cluster Instances or Database Mirroring. This application consists of 11 clusters which are created under the SYSTEM tablespace. We can configure Always on Failover Cluster Instances which leverages Windows Server Failover Clustering (WSFC) functionality to provide local high availability through redundancy at the server-instance level—a *failover cluster instance* (FCI). Windows Server Failover Clustering (WSFC) is a feature of the Windows Server platform for improving the high availability (HA) of applications and services. With Windows Server Failover Clustering, each active server has another server identified as its standby server. For a failover cluster to work, each server's hardware specifications must be the same and the servers must share storage. The two servers communicate through a series of "heartbeat" signals over a dedicated network. SQL Server 2016 takes advantage of WSFC services and capabilities to support Always On Availability Groups and SQL Server Failover Cluster Instances. Windows Server Failover Clustering provides infrastructure features that support the high-availability and disaster recovery scenarios of hosted server applications such as Microsoft SQL Server and Microsoft Exchange. If a cluster node or service fails, the services that were hosted on that node can be automatically or manually transferred to another available node in a process known as failover.
Performance Recommendation	• A two-node Active/Passive configuration is preferred over an Active/Active configuration. • If you build a 3-node or larger cluster, one node should be dedicated to failover, while the rest of the nodes run a single instance of SQL Server.

	• Running multiple instances of SQL Server on the same node is not recommended. • SQL Server clusters should be dedicated to SQL Server. Don't run other software on these nodes. • Cluster nodes should be member servers, not domain controllers. • Cluster nodes must be all in the same domain and have access to multiple domain controllers. • Before you begin to install Cluster Services or SQL Server Clustering, determine your virtual names and IP addresses. • After each major step, check all logs to look for potential problems, before moving onto the next step. • Before installing SQL Server Clustering, ensure that the Windows Cryptographic Service Provider and the Task Scheduler are on. • Once Clustering Services and SQL Server Clustering have been installed, give the domain controllers some time to replicate the virtual server names before clients try to access them.
References	https://docs.oracle.com/cd/B28359_01/rac.111/b28255/intro.htm#BABCHEEE https://docs.oracle.com/cd/B28359_01/server.111/b28310/clustrs001.htm#ADMIN11739 https://msdn.microsoft.com/en-us/library/ee784936(v=cs.20).aspx https://www.brentozar.com/archive/2012/02/introduction-sql-server-clusters. https://blogs.msdn.microsoft.com/clustering/2014/05/08/deploying-sql-server-2014-with-cluster-shared-volumes/ https://msdn.microsoft.com/en-us/library/ms189134.aspx

Packages

Feature ID	51
Feature	Packages
Description	A package is a schema object that groups logically related PL/SQL types, variables, constants, subprograms, cursors, and exceptions. Oracle supports encapsulating variables, types, stored procedures, and functions into a package. A package is compiled and stored in the database, where many applications can share its contents. A package always has a specification, which declares the public items that can be referenced from outside the package. If the public items include cursors or subprograms, then the package must also have a body. The body must define queries for public cursors and code for public subprograms.
Category	Platform
To Find Feature Enablement	`SELECT count(*) FROM DBA_OBJECTS WHERE OBJECT_TYPE IN ('PACKAGE')`
Feature Usage	`SELECT * FROM DBA_OBJECTS WHERE OBJECT_TYPE IN ('PACKAGE')`
Recommendation	**Feature Description**: SQL Server does not support objects with functionality like that of a ORACLE Packages. **Feature Comparison:** Some Oracle object categories, such as packages, do not have direct SQL Server equivalents. SSMA converts each packaged procedure or function into separate target subroutines and applies rules for stand-alone procedures or functions. When you convert Oracle packages, you need to convert: o Packaged procedures and functions (both public and private). o Packaged variables. o Packaged cursors. o Package initialization routines.

Migration Approach	As studied above, SQL Server does not provide packages as a direct feature, but the below workaround can be used to convert Oracle packages to its SQL Server equivalent.
	
	In SQL Server 2014, you can group procedures and functions by their names. Suppose that you have the following Oracle package:
	
	`CREATE OR REPLACE PACKAGE MY_PACKAGE`
	` IS`
	`space varchar(1) := ' ';`

	``
	unitname varchar(128) := 'My Simple Package';
	curd date := sysdate;
	procedure MySimpleProcedure;
	procedure MySimpleProcedure(s in varchar);
	function MyFunction return varchar2;
	END;
	
	CREATE OR REPLACE PACKAGE BODY MY_PACKAGE
	IS
	procedure MySimpleProcedure
	is begin
	dbms_output.put_line(MyFunction);
	end;
	procedure MySimpleProcedure(s in varchar)
	is begin
	dbms_output.put_line(s);
	end;
	function MyFunction return varchar2
	is begin
	return 'Hello, World!';
	end;
	``
	
	In SQL Server we can emulate Oracle Packages, you can group procedures and functions by their names. such as:
	`Scott.MY_PACKAGE$MySimpleProcedure and`
	`Scott.MY_PACKAGE$MyFunction.`
	
	The naming pattern is:
	`<schema name>.<package name>$<procedure or function name>`

References	https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6006.htm

Sequences

Feature ID	53
Feature	Sequences
Description	An ORACLE sequence is a user-defined object that generates a series of numeric values based on the specification with which the sequence was created. The most common purpose of a sequence is to provide unique values for the primary key column of a table. ORACLE sequences are not associated with any tables. Applications refer to a sequence object to get the current or next value of that sequence. ORACLE keeps the set of generated values of a sequence in a cache, and a unique set of cached values is created for each session. In ORACLE, the NEXTVAL expression generates and returns the next value for the specified sequence. The ORACLE CURRVAL expression returns the most recently generated value of the previous NEXTVAL expression for the same sequence within the current application process. In ORACLE, the value of the CURRVAL expression persists until the next value is generated for the sequence, the sequence is dropped, or the application session ends.
Category	Platform
To Find Feature Enablement	Select count(*) from DBA_SEQUENCES
Feature Usage	Select * from DBA_SEQUENCES;
Recommendation	**Feature Description:** • A sequence is a user-defined schema bound object that generates a sequence of numeric values according the specification with which the sequence was created. • The sequence of numeric values is generated in an ascending or descending order at a defined interval and can be configured to restart (cycle) when exhausted. Sequences, unlike identity columns, are not associated with specific tables. • Applications refer to a sequence object to retrieve its next value. The relationship between sequences and tables is controlled by the application. User applications can reference a sequence object and coordinate the values across multiple rows and tables. **Feature Comparison:** SQL Server 2014 supports objects with functionality similar as ORACLE sequence.
Migration Approach	Below are the steps to migrate your Oracle Sequences to SQL Server • Start SSMA • **Connect to Oracle -** Click on the **Connect to Oracle** icon from the menu toolbar and provide connection information to your Oracle database.

- **Create a schema migration report** - Select the schema, then right-click the schema then select ***Create Report***:
- **Connect to SQL Server -** Click on *the Connect to SQL Server* icon from the File Menu. Specify the server name (e.g. *localhost* if SSMA is running on the SQL server machine) and port number (if using other than default 1433 SQL Server port number). Type the name of the database you are migrating to (e.g. **HR**). If the database does not exist, SSMA will create a new database using the default setting. Specify authentication information and click *Connect* to continue.
- **Map Schema and Type -** In the **Oracle Metadata Explorer**, check the schema and expand. You can select (or deselect) objects to be migrated as well as map schema. Select Sequences. Schema mapping can be done at the Oracle schema level or at the individual object (such as specific table in Oracle) to SQL Server schema.
- **Convert the schema -** In the **Oracle Metadata Explorer**, right-click the **schema** and select ***Convert Schema***:
- **Review conversion report and resolve error as necessary**.
- **Synchronize the SQL Server database**. To deploy the changes to the SQL server, right-click the database in the SQL Server metadata explorer and select **Synchronize with Database**.
- **Migrate the data**. From Oracle Metadata Explorer window, right-click on the schema and select ***Migrate Data***. Provide connection information to both the Oracle source database and the target SQL server.
- **Review Migration Report.** After the data is migrated, a report will be displayed with migration statistics

In many cases if you use sequence only for getting NEXTVAL you can convert it to SQL Server sequence.

ORACLE

```
CREATE SEQUENCE seq1;
...
INSERT INTO t1 (id, name)
VALUES (seq1.NEXTVAL, 'name');
INSERT INTO t2 (id, name)
VALUES (seq1.CURRVAL, 'name');
...
```

SQL Server

```
CREATE SEQUENCE seq1
...
declare @newid int;
select @newid = NEXT VALUE FOR seq1;
INSERT INTO t1 (id, name)
VALUES (@newid, 'name');
INSERT INTO t2 (id, name)
VALUES (@newid, 'name');
```

However, some features of ORACLE sequences (e.g. CURRVAL) are not supported in SQL Server.

	Workaround for Resolving the issue Two distinct scenarios of ORACLE sequence CURRVAL usage exist: a variable that saves sequence value, and an auxiliary table that represents an ORACLE sequence. In this scenario, an ORACLE sequence is used in a way that is incompatible with SQL Server sequence. For example, NEXTVAL and CURRVAL of sequence can be used in different procedures or application modules. In this case, you can create an auxiliary table to represent the ORACLE sequence object. This table contains a single column declared as IDENTITY. When you need to get a new sequence value, you insert a row in this auxiliary table and then retrieve the automatically assigned value from the new row. create table MY_SEQUENCE (id int IDENTITY(1 /* seed */, 1 /* increment*/)) go To maintain such emulation of NEXTVAL, you must clean up the added rows to avoid unrestricted growth of the auxiliary table. The fastest way to do this in SQL Server is to use a transactional approach. declare @tran bit, @nextval int set @tran = 0 if @@trancount > 0 begin save transaction seq set @tran = 1 end else begin transaction insert into MY_SEQUENCE default values set @nextval = SCOPE_IDENTITY() if @tran=1 rollback transaction seq else rollback In SQL Server, IDENTITY is generated in a transaction-independent way and, as in ORACLE, rolling back the transaction does not affect the current IDENTITY value. In this scenario, we can emulate CURRVAL by using SQL Server @@IDENTITY or SCOPE_IDENTITY() functions. @@IDENTITY returns the value for the last INSERT statement in the session, and SCOPE_IDENTITY() gets the last IDENTITY value assigned within the scope of current Transact-SQL module. Note that the values returned by these two functions can be overwritten by next INSERT statement in the current session, so we highly recommend that you save the value in an intermediate variable, if CURRVAL is used afterwards in the source

	code. Both @@IDENTITY and SCOPE_IDENTITY() are limited to the current session scope, which means that as in ORACLE, the identities generated by concurrent processes are not visible.
References	https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_6015.htm#SQLRF01314 https://msdn.microsoft.com/en-us/library/ff878091.aspx https://blogs.msdn.microsoft.com/ssma/2011/07/12/converting-oracle-sequence-to-sql-server-denali/

Snapshot

Feature ID	54
Feature	Snapshot
Description	A snapshot is a replica of a target master table from a single point-in-time. Whereas in multimaster replication tables are continuously being updated by other master sites, snapshots are updated by one or more master tables via individual batch updates, known as a *refresh*, from a single master site. Oracle offers a variety of snapshots to meet the needs of many different replication (and non-replication) situations. You might use a snapshot to achieve one or more of the following: • Ease Network Loads • Mass Deployment • Data Subsetting • Disconnected Computing
Category	Platform
To Find Feature Enablement	`Select count(*) from DBA_HIST_SNAPSHOT`
Feature Usage	`Select * from DBA_HIST_SNAPSHOT`
Recommendation	**Feature Description:** • A database snapshot is a read-only, static view of a SQL Server database (the *source database*). • Database snapshots operate at the data-page level. Before a page of the source database is modified for the first time, the original page is copied from the source database to the snapshot. The snapshot stores the original page, preserving the data records as they existed when the snapshot was created. The same process is repeated for every page that is being modified for the first time. To the user, a database snapshot appears never to change, because read operations on a database snapshot always access the original data pages, regardless of where they reside. • The database snapshot is transitionally consistent with the source database as the moment of the snapshot's creation. • A database snapshot always resides on the same server instance as its source database. • As the source database is updated, the database snapshot is updated. Therefore, the longer a database snapshot exists, the more likely it is to use up its available disk space. • Multiple snapshots can exist on a given source database. Each database snapshot persists until it is explicitly dropped by the database owner. • Database snapshots are unrelated to snapshot backups, snapshot isolation of transactions, or snapshot replication. • Snapshots can be used for reporting purposes.

	• In the event of a user error on a source database, you can revert the source database to the state it was in when a given database snapshot was created. Data loss is confined to updates to the database since the snapshot's creation. • Database snapshots are dependent on the source database. Therefore, using database snapshots for reverting a database is not a substitute for your backup and restore strategy. Performing all your scheduled backups remains essential. If you must restore the source database to the point in time at which you created a database snapshot, implement a backup policy that enables you to do that. • sparse file- A file provided by the NTFS file system that requires much less disk space than would otherwise be needed. A sparse file is used to store pages copied to a database snapshot. When first created, a sparse file takes up little disk space. As data is written to a database snapshot, NTFS allocates disk space gradually to the corresponding sparse file. **Feature Comparison:** Oracle and SQL Server has defined Snapshots differently, and hence we can migrate or rewrite Snapshots in SQL Server. We can use Snapshot Replication, Transactional Replication or Merge replication to solve the purpose of Oracle Snapshots.

Migration Approach	We need to rewrite the Oracle Snapshots in SQL Server and look for a workaround depending on the type of Snapshot we are migrating.
	
	Creating a snapshot on the AdventureWorks database
	This example creates a database snapshot on the AdventureWorks database. The snapshot name, AdventureWorks_dbss_1800, and the file name of its sparse file, AdventureWorks_data_1800.ss, indicate the creation time, 6 P.M (1800 hours).
	
	`CREATE DATABASE AdventureWorks_dbss1800 ON`
	`( NAME = AdventureWorks_Data, FILENAME =`
	`'C:\Program Files\Microsoft SQL`
	`Server\MSSQL13.MSSQLSERVER\MSSQL\Data\AdventureWorks_data_1800.ss`
	`' )`
	`AS SNAPSHOT OF AdventureWorks;`
	`GO`
	
	To drop database snapshot
	`DROP DATABASE SalesSnapshot0600 ;`
	
	Reverting a snapshot on the AdventureWorks database
	`USE master;`
	`-- Reverting AdventureWorks to AdventureWorks_dbss1800`
	`RESTORE DATABASE AdventureWorks from`
	`DATABASE_SNAPSHOT = 'AdventureWorks_dbss1800';`

	GO
References	https://docs.oracle.com/cd/F49540_01/DOC/server.815/a67791/mview.htm https://msdn.microsoft.com/en-us/library/ms175158.aspx https://blogs.msdn.microsoft.com/dbrowne/2013/07/25/how-to-move-a-database-that-has-database-snapshots/

Built-In Functions

<table>
<tr><td>Feature ID</td><td>56</td></tr>
<tr><td>Feature</td><td>Built-In Functions</td></tr>
<tr><td>Description</td><td>A built-in function is an expression in which an SQL keyword or special operator executes some operation. Built-in functions use keywords or special built-in operators. Built-ins are SQL92Identifiers and are case-insensitive.</td></tr>
<tr><td>Category</td><td>Platform</td></tr>
<tr><td>To Find Feature Enablement</td><td>select count(*)
from
  all_arguments
where
  package_name = 'STANDARD';</td></tr>
<tr><td>Feature Usage</td><td>select distinct object_name from all_arguments
    where package_name = 'STANDARD';</td></tr>
<tr><td>Recommendation</td><td><u>Feature Description:</u>
• SQL Server provides many built-in functions and lets you create user-defined functions
• SQL Server built-in functions are either deterministic or nondeterministic.
• Functions are deterministic when they always return the same result any time they are called by using a specific set of input values.
• Functions are nondeterministic when they could return different results every time they are called, even with the same specific set of input values.

<u>Feature Comparison:</u>
SQL Server supports all the built-in functions which are present in Oracle.
Below table will list the Oracle built in functions and their equivalent SQL Server function.

<table>
<tr><th colspan="2">Oracle</th><th colspan="2">SQL Server</th></tr>
<tr><td>1</td><td>ADD_MONTHS</td><td>Add specified number of months</td><td>DATEADD</td></tr>
<tr><td>2</td><td>CAST</td><td>Convert one built-in data type into another</td><td>CAST</td></tr>
<tr><td>3</td><td>DECODE</td><td>Evaluate a list of conditions</td><td>CASE Expression</td></tr>
<tr><td>4</td><td>EMPTY_BLOB</td><td>Create an empty BLOB value</td><td>0x Constant (Empty binary string)</td></tr>
</table>
</td></tr>
</table>

5	EMPTY_CLOB	Create an empty CLOB or NCLOB value	'' (Empty string)	
6	EXTRACT for Datetime	Extract day, month, year etc from datetime		
7	INITCAP	Capitalize the first letter of each word	User-defined function	
8	INSTR	Find position of substring in string	CHARINDEX	First occurrence only, different parameter order ⚠
9	LAST_DAY	Get last date of month	EOMONTH	Since SQL Server 2012 ⚠
10	LENGTH	Get string length in characters	LEN	CHAR handled differently, excludes trailing spaces ⚠
11	LOWER	Convert string to lowercase	LOWER	
12	LPAD	Left-pad string to the specified length	Expression using REPLICATE, RIGHT and LEFT	
13	MOD	Get the remainder of division of one number by another	% Operator	
14	MONTHS_BETWEEN	Get number of months between two dates		
15	NVL	Replace NULL with expression	ISNULL	
16	REPLACE	Replaces all occurrences of string with another string	REPLACE	
17	SIGN	If value is positive return 1, if negative then -1, if zero then 0	SIGN	

18	SUBSTR	Return a substring from string	SUBSTRING	Negative start position is not allowed, length must be specified ⚠
19	TO_CHAR for Datetime	Convert datetime to string	CONVERT	
20	TO_DATE	Convert string to datetime	CONVERT	
21	TRANSLATE	One-to-one single-character substitution	Expressions using REPLACE or User-defined function	
22	TRIM	Trim leading or trailing characters	LTRIM and RTRIM	
23	TRUNC for Datetime	Truncate datetime	Expressions using CONVERT	
24	UNISTR	Convert Unicode code points to characters	Expressions using NCHAR	

Arithmetic Functions

	Oracle	SQL Server	
1	MOD	Get the remainder of division of one number by another	% Operator
2	SIGN	If value is positive return 1, if negative then -1, if zero then 0	SIGN

String Functions

	Oracle	SQL Server		
1	INITCAP	Capitalize the first letter of each word	User-defined function	
2	INSTR	Find position of substring in string	CHARINDEX	First occurrence only, different parameter order ⚠
3	LENGTH	Get string length in characters	LEN	CHAR handled differently, excludes trailing spaces ⚠
4	LOWER	Convert string to lowercase	LOWER	

5	LPAD	Left-pad string to the specified length	Expression using REPLICATE, RIGHT and LEFT	
6	REPLACE	Replaces all occurrences of string with another string	REPLACE	
7	SUBSTR	Return a substring from string	SUBSTRING	Negative start position is not allowed, length must be specified ⚠
8	TO_CHAR for Datetime	Convert datetime to string	CONVERT	
9	TRANSLATE	One-to-one single-character substitution	Expressions using REPLACE or User-defined function	
10	TRIM	Trim leading or trailing characters	LTRIM and RTRIM	
11	UNISTR	Convert Unicode code points to characters	Expressions using NCHAR	

Datetime Functions:

	Oracle	**SQL Server**	
1	ADD_MONTHS	Add specified number of months	DATEADD
2	EXTRACT for Datetime	Extract day, month, year etc from datetime	
3	LAST_DAY	Get last date of month	EOMONTH
4	MONTHS_BETWEEN	Get number of months between two dates	
5	TO_CHAR for Datetime	Convert datetime to string	CONVERT
6	TO_DATE	Convert string to datetime	CONVERT
7	TRUNC for Datetime	Truncate datetime	Expressions using CONVERT

Conversion and Format Functions:

	Oracle	SQL Server	
1	CAST	Convert one built-in data type into another	
2	TO_CHAR for Datetime	Convert datetime to string	CONVERT
3	TO_DATE	Convert string to datetime	CONVERT
4	TRANSLATE	One-to-one single-character substitution	Expressions using REPLACE or User-defined function
5	UNISTR	Convert Unicode code points to characters	Expressions using NCHAR

Case and Decode Functions:

	Oracle	SQL Server	
1	DECODE	Evaluate a list of conditions	CASE Expression
2	NVL	Replace NULL with expression	ISNULL
3	SIGN	If value is positive return 1, if negative then -1, if zero then 0	SIGN

NULL Functions:

	Oracle	SQL Server	
1	NVL	Replace NULL with expression	ISNULL

LOB Functions

	Oracle	SQL Server

	1	EMPTY_BLOB	Create an empty BLOB value	0x Constant (Empty binary string)
	2	EMPTY_CLOB	Create an empty CLOB or NCLOB value	'' (Empty string)

Migration Approach

SSMA converts Oracle system functions to either SQL Server system functions or to user-defined functions from the Microsoft Extension Library for SQL Server.
The library is created in the SSMA oracle schema when you convert your database.
Below are the steps to migrate your Oracle Sequences to SQL Server

Below are the steps to migrate your Oracle Functions to SQL Server

- Start SSMA
- **Connect to Oracle -** Click on the **Connect to Oracle** icon from the menu toolbar and provide connection information to your Oracle database.
- **Create a schema migration report** - Select the schema, then right-click the schema then select ***Create Report***:
- **Connect to SQL Server -** Click on *the Connect to SQL Server* icon from the File Menu. Specify the server name (e.g. *localhost* if SSMA is running on the SQL server machine) and port number (if using other than default 1433 SQL Server port number). Type the name of the database you are migrating to (e.g. **HR**). If the database does not exist, SSMA will create a new database using the default setting. Specify authentication information and click *Connect* to continue.
- **Map Schema and Type -** In the **Oracle Metadata Explorer**, check the schema and expand. You can select (or deselect) objects to be migrated as well as map schema. Select **Functions**. Schema mapping can be done at the Oracle schema level or at the individual object (such as specific table in Oracle) to SQL Server schema.
- **Convert the schema -** In the **Oracle Metadata Explorer**, right-click the **schema** and select ***Convert Schema***:
- **Review conversion report and resolve error as necessary**.
- **Synchronize the SQL Server database**. To deploy the changes to the SQL server, right-click the database in the SQL Server metadata explorer and select **Synchronize with Database**.
- **Migrate the data**. From Oracle Metadata Explorer window, right-click on the schema and select ***Migrate Data***. Provide connection information to both the Oracle source database and the target SQL server.
- **Review Migration Report.** After the data is migrated, a report will be displayed with migration statistics

Note: The prefix [ssma_oracle] is placed before functions in the ssma_oracle schema, as required for SQL Server functions that are part of the SSMA conversion.

	Note that the following functions are not supported on Azure SQL DB: CUME_DIST, LAG, LEAD, FIRST_VALUE, LAST_VALUE, PERCENTILE_DISC, PERCENTILE_RANK, PERCENTILE_COST.
References	http://docs.oracle.com/javadb/10.6.2.1/ref/rrefsqlj29026.html https://msdn.microsoft.com/en-us/library/ms174318.aspx http://www.sqlines.com/oracle-to-sql-server

Data Concurrency and Locking Concepts

Feature ID	57
Feature	Locking Concepts and Data Concurrency Issue
Description	In a single-user database, the user can modify data in the database without concern for other users modifying the same data at the same time. However, in a multiuser database, the statements within multiple simultaneous transactions can update the same data. Transactions executing at the same time need to produce meaningful and consistent results. Therefore, control of data concurrency and data consistency is vital in a multiuser database. • **Data concurrency** means that many users can access data at the same time. • **Data consistency** means that each user sees a consistent view of the data, including visible changes made by the user's own transactions and transactions of other users. In general, multiuser databases use some form of data locking to solve the problems associated with data concurrency, consistency, and integrity. Locks are mechanisms that prevent destructive interaction between transactions accessing the same resource. Resources include two general types of objects: • User objects, such as tables and rows (structures and data) • System objects not visible to users, such as shared data structures in the memory and data dictionary rows
Category	Platform

To Find Feature Enablement	`select username,`
	`v$lock.sid,`
	`trunc(id1/power(2,16)) rbs,`
	`bitand(id1, to_number('ffff', 'xxxx'))+0 slot,`
	`id2 seq,`
	`lmode,`
	`request`
	`from v$lock, v$session`
	`where v$lock.type = 'TX'`
	`and v$lock.sid = v$session.sid`
	`and v$session.username = USER;`

Feature Usage	--Identify locks and Transaction ID's
	`select username,`
	`v$lock.sid,`
	`trunc(id1/power(2,16)) rbs,`
	`bitand(id1, to_number('ffff', 'xxxx'))+0 slot,`
	`id2 seq,`
	`lmode,`
	`request`
	`from v$lock, v$session`
	`where v$lock.type = 'TX'`

	`and v$lock.sid = v$session.sid`
	`and v$session.username = USER;`
	
	--Identify who is blocking whom
	`select (select username from v$session where sid=a.sid) blocker,`
	`a.sid,`
	`' is blocking ',`
	`(select username from v$session where sid=b.sid) blockee,`
	`b.sid`
	`from v$lock a, v$lock b`
	`where a.block = 1`
	`and b.request > 0`
	`and a.id1 = b.id1`
	`and a.id2 = b.id2;`

Recommendation	**Feature Description:**
	In a multiple-user environment, there are two models for updating data in a database:
	• Pessimistic concurrency involves locking the data at the database when you read it. You exclusively lock the database record and don't allow anyone to touch it until you are done modifying and saving it back to the database. You have 100 percent assurance that nobody will modify the record while you have it checked out. Another person must wait until you have made your changes. Pessimistic concurrency complies with ANSI-standard isolation levels as defined in the SQL-99 standard. Microsoft SQL Server 2014 has four pessimistic isolation levels:
	o READ COMMITTED
	o READ UNCOMMITTED
	o REPEATABLE READ
	o SERIALIZABLE
	
	Optimistic concurrency means that you read the database record but don't lock it. Anyone can read and modify the record at any time, so the record might be modified by someone else before you modify and save it. If data is modified before you save it, a collision occurs. Optimistic concurrency is based on retaining a view of the data as it is at the start of a transaction.
	
	Locking serves as a control mechanism for concurrency. Locking is a necessity in a multi-user environment because more than one user at a time may be working with the same data.
	
	SQL Query For getting the current execution
	Querying this DMV, at any moment, gives you a quick view of everything executing right then. The wait_type, wait_time and last_wait_type columns will give you an immediate feel of what is 'runnig' vs. what is 'waiting' and what is being waited for:
	
	`select session_id,`
	`status,`
	`command,`
	`blocking_session_id,`
	`wait_type,`

```
        wait_time,
        last_wait_type,
        wait_resource
from sys.dm_exec_requests
where r.session_id >= 50
and r.session_id <> @@spid;
```

Execute the following query to view wait stats for all block processes on SQL Server

```
USE [master]
GO
SELECT   w.session_id
 ,w.wait_duration_ms
 ,w.wait_type
 ,w.blocking_session_id
 ,w.resource_description
 ,s.program_name
 ,t.text
 ,t.dbid
 ,s.cpu_time
 ,s.memory_usage
FROM sys.dm_os_waiting_tasks w
INNER JOIN sys.dm_exec_sessions s
ON w.session_id = s.session_id
INNER JOIN sys.dm_exec_requests r
ON s.session_id = r.session_id
OUTER APPLY sys.dm_exec_sql_text (r.sql_handle) t
WHERE s.is_user_process = 1
GO
```

Refer this link https://www.mssqltips.com/sqlservertip/2732/different-techniques-to-identify-blocking-in-sql-server/ for further information on similar queries .

Feature Comparison:

Oracle	SQL Server
Fully automatic and does not require intervention by users.	Fully automatic and does not require intervention by users.
Data locks (DML locks) to protect data. The "table locks" lock the entire table and "row locks" lock individual rows. DML operations can acquire data locks at two different levels; one for specific rows and one for entire tables	Applies exclusive locks for INSERT, UPDATE, and DELETE operations. When an exclusive lock is set, no other transaction can obtain any type of lock on those objects until the original lock is in place
Dictionary locks (DDL locks) to protect the structure of objects.	For non-update or read operations, a shared lock is applied. If a shared lock is applied to a table or a page, other transactions can also obtain a shared lock on that table or page. However, no transaction can obtain

		an exclusive lock. Therefore, Microsoft SQL Server reads block the modifications to the data.
	Oracle has a row-locking feature. Only one row is locked when a DML statement is changing the row.	Does not have a row-level locking feature. SQL Server applies a page-level lock, which effectively locks all rows on the page, whenever any row in the page is being updated. This is an exclusive lock whenever the data is being changed by DML statements. Implements a form of row level lock.

- In Microsoft SQL Server, SELECT statements obtain shared locks on pages/rows.
- This prevents other statements from obtaining an exclusive lock on those pages/rows.
- All statements that update the data need an exclusive lock. This means that the SELECT statement in Microsoft SQL Server blocks the UPDATE statements as long the transaction that includes the SELECT statement does not commit or rollback.
- This also means that two transactions are physically serialized whenever one transaction selects the data and the other transaction wants to change the data first and then select the data again.
- In Oracle, however, SELECT statements do not block UPDATE statements, since the rollback segments are used to store the changed data before it is updated in the actual tables. Also, the reader of the data is never blocked in Oracle. This allows Oracle transactions to be executed simultaneously.

Logical Transaction Handling

- In Microsoft SQL Server, logical transactions are automatically translated to Oracle logical transactions, the preceding transactions that execute properly in Microsoft SQL Server as they are serialized cause a deadlock in Oracle.
- These transactions should be identified and serialized to avoid the deadlock.
- These transactions are serialized in Microsoft SQL Server as INSERT, UPDATE, and DELETE statements block other statements.
- Transactions are not implicit in Microsoft SQL Server. Therefore, applications expect that every statement they issue is automatically committed it is executed.
- Oracle transactions are always implicit, which means that individual statements are not committed automatically.
- In Microsoft SQL Server, transactions may also be explicitly begun by a client application by issuing a BEGIN TRAN statement during the conversion process.

Migration Approach	• Because SQL Server 2014 and above has completely controllable isolation-level models, you can choose the most appropriate isolation level. • To control a row-versioning isolation level, use the SET TRANSACTION ISOLATION LEVEL command.

- SNAPSHOT is the isolation level that is similar to Oracle and does optimistic escalations.

Make Transaction Behavior Look Like Oracle

- For complete transaction management emulation in SQL Server 2014, and using a row-versioning isolation level, set the ALLOW_SNAPSHOT_ISOLATION option to ON for each database that is referenced in the Transact-SQL object (view, procedure, function, or trigger).
- In addition, either each Transact-SQL object must be started with a SNAPSHOT isolation level; otherwise, this level must be set on each client connection.
- Alternatively, the autonomous block must be started with the READ COMMITTED isolation level with the READ_COMMITTED_SNAPSHOT database option set to ON.
- Both the READ_COMMITTED_SNAPSHOT and ALLOW_SNAPSHOT_ISOLATION database options are set to ON in Azure SQL DB by default and cannot be changed

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SELECT es.session_id AS session_id
,COALESCE(es.original_login_name, '') AS login_name
,COALESCE(es.host_name,'') AS hostname
,COALESCE(es.last_request_end_time,es.last_request_start_time) AS
last_batch
,es.status
,COALESCE(er.blocking_session_id,0) AS blocked_by
,COALESCE(er.wait_type,'MISCELLANEOUS') AS waittype
,COALESCE(er.wait_time,0) AS waittime
,COALESCE(er.last_wait_type,'MISCELLANEOUS') AS lastwaittype
,COALESCE(er.wait_resource,'') AS waitresource
,coalesce(db_name(er.database_id),'No Info') as dbid
,COALESCE(er.command,'AWAITING COMMAND') AS cmd
,sql_text=st.text
,transaction_isolation =
CASE es.transaction_isolation_level
    WHEN 0 THEN 'Unspecified'
    WHEN 1 THEN 'Read Uncommitted'
    WHEN 2 THEN 'Read Committed'
    WHEN 3 THEN 'Repeatable'
    WHEN 4 THEN 'Serializable'
    WHEN 5 THEN 'Snapshot'
END
,COALESCE(es.cpu_time,0)
    + COALESCE(er.cpu_time,0) AS cpu
,COALESCE(es.reads,0)
    + COALESCE(es.writes,0)
    + COALESCE(er.reads,0)
+ COALESCE(er.writes,0) AS physical_io
,COALESCE(er.open_transaction_count,-1) AS open_tran
,COALESCE(es.program_name,'') AS program_name
,es.login_time
FROM sys.dm_exec_sessions es
```

	LEFT OUTER JOIN sys.dm_exec_connections ec ON es.session_id = ec.session_id LEFT OUTER JOIN sys.dm_exec_requests er ON es.session_id = er.session_id LEFT OUTER JOIN sys.server_principals sp ON es.security_id = sp.sid LEFT OUTER JOIN sys.dm_os_tasks ota ON es.session_id = ota.session_id LEFT OUTER JOIN sys.dm_os_threads oth ON ota.worker_address = oth.worker_address CROSS APPLY sys.dm_exec_sql_text(er.sql_handle) AS st where es.is_user_process = 1 and es.session_id <> @@spid and es.status = 'running' ORDER BY es.session_id
Performance Recommendation	Although we cannot completely avoid deadlocks but they can be minimised by following the tips below: • Ensure the database is normalized properly because bad database design can increase the number of deadlocks to occur inside database. • Deadlocks can also occur if the resources are not acquired in some well-defined order because if all concurrent transactions access objects in the same order, deadlocks are less likely to occur. Using stored procedures for all data modifications can standardize the order of accessing objects. We need to define some programming policy which governs the order in which database objects can be accessed. • We must release locks in the opposite order to that in which we acquired them, and should release them in a finally clause of TRY/CATCH block. Properly analyses the situation and ensure that all resources within your code are acquired in some well-defined order. • Do not allow users to input the data during transactions. Update all the data before the transaction begins. • Avoid cursors if possible because same transaction locking rules will apply to SELECT statement within a cursor definition that applies to any other SELECT statement. • Keeping the transaction in one batch minimizes network roundtrips during a transaction, reducing possible delays in completing the transaction and releasing locks. • Reduce the time a transaction takes to complete by making sure you are not performing the same reads repeatedly. • If your application does need to read the same data more than once, cache it by storing it in a variable or an array, and then re-reading it from there, not from SQL Server. • Reduce lock time. Try to develop your application so that it grabs locks at the latest possible time, and then releases them at the very earliest time. • Lock escalation reduces the overall number of locks being held on the SQL Server instance, reducing the lock memory usage. • Consider using the NOLOCK hint where possible, which will override locking of the whole table and allow access to it to other queries. • Using a lower isolation level, such as read committed, holds shared locks for a shorter duration than a higher isolation level, such as SERIALIZABLE. This reduces locking contention.

	• Use bound connections because two or more connections opened by the same application can cooperate with each other. Any locks acquired by the secondary connections are held as if they were acquired by the primary connection, and vice versa. Therefore, they do not block each other.
References	https://docs.oracle.com/cd/B19306_01/server.102/b14220/consist.htm https://technet.microsoft.com/en-us/library/ms187101(v=sql.105).aspx

Change Data Capture

Feature ID	58
Feature	Change Data Capture
Description	Change Data Capture efficiently identifies and captures data that has been added to, updated in, or removed from, Oracle relational tables and makes this change data available for use by applications or individuals. Often, data warehousing involves the extraction and transportation of relational data from one or more production databases into a data warehouse for analysis. Change Data Capture quickly identifies and processes only the data that has changed and makes the change data available for further use.
Category	SQL
To Find Feature Enablement	Select count(*) from ALL_CHANGE_TABLES
Feature Usage	Select * from ALL_CHANGE_TABLES
Recommendation	**Feature Description:** • Change data capture records insert, update, and delete activity that is applied to a SQL Server table. • This makes the details of the changes available in an easily consumed relational format. • Column information and the metadata that is required to apply the changes to a target environment is captured for the modified rows and stored in change tables that mirror the column structure of the tracked source tables. • Table-valued functions are provided to allow systematic access to the change data by consumers. **Feature Comparison:** Oracle and SQL Server has defined Change Data Capture in its own ways, but the purpose both meet is the same.
Migration Approach	We can implement CDC in SQL server differently and hence it is not a part of the migration in SSMA. However, to create a capture instance for individual table you must enable change data capture for a database. For that, a member of the **sysadmin** fixed server role must first enable the database for change data capture. This is done by running the stored procedure sys.sp_cdc_enable_db (Transact-SQL) in the database context. To determine if a database is already enabled, query the **is_cdc_enabled** column in the **sys.databases** catalog view. • When a database is enabled for change data capture, the **cdc** schema, **cdc** user, metadata tables, and other system objects are created for the database. The **cdc** schema contains the change data capture metadata tables and, after source tables are enabled for change data capture, the individual change tables serve as a repository for change data. The **cdc** schema also contains associated system functions used to query for change data. • Change data capture requires exclusive use of the **cdc** schema and **cdc** user. If either a schema or a database user named *cdc* currently

	exists in a database, the database cannot be enabled for change data capture until the schema and or user are dropped or renamed.
	Transact-SQL code to enable change data capture:
	`-- ====`
	`-- Enable Database for CDC template`
	`-- ====`
	`USE MyDB`
	`GO`
	`EXEC sys.sp_cdc_enable_db`
	`GO`
	And to disable change data capture for a database a member of the **sysadmin** fixed server role can run the stored procedure sys.sp_cdc_disable_db (Transact-SQL).
	Transact-SQL code to disable change data capture:
	`-- =======`
	`-- Disable Database for Change Data Capture template`
	`-- =======`
	`USE MyDB`
	`GO`
	`EXEC sys.sp_cdc_disable_db`
	`GO`

References	http://docs.oracle.com/javadb/10.6.2.1/ref/rrefsqlj29026.html https://msdn.microsoft.com/en-us/library/ms174318.aspx http://www.sqlines.com/oracle-to-sql-server

File Groups

<table>
<tr><td>Feature ID</td><td>61</td></tr>
<tr><td>Feature</td><td>File Groups</td></tr>
<tr><td>Description</td><td>A file group repository can contain multiple file groups and multiple versions of a particular file group. A tablespace repository is a collection of tablespace sets in a file group repository. Tablespace repositories are built on file group repositories, but tablespace repositories only contain the files required to move or copy tablespaces between databases. This chapter provides sample queries that you can use to monitor file group repositories and tablespace repositories.</td></tr>
<tr><td>Category</td><td>Platform</td></tr>
<tr><td>To Find Feature Enablement</td><td>Select count(*) from dba_file_groups;</td></tr>
<tr><td>Feature Usage</td><td>select * from dba_file_group_versions;

select * from dba_file_groups;</td></tr>
<tr><td>Recommendation</td><td><u>Feature Description:</u>

At a minimum, every SQL Server database has two operating system files: a data file and a log file. Data files contain data and objects such as tables, indexes, stored procedures, and views. Log files contain the information that is required to recover all transactions in the database. Data files can be grouped together in filegroups for allocation and administration purposes.

Every database has a primary filegroup. This filegroup contains the primary data file and any secondary files that are not put into other filegroups. User-defined filegroups can be created to group data files together for administrative, data allocation, and placement purposes.

For example, three files, Data1.ndf, Data2.ndf, and Data3.ndf, can be created on three disk drives, respectively, and assigned to the filegroup fgroup1. A table can then be created specifically on the filegroup fgroup1. Queries for data from the table will be spread across the three disks; this will improve performance. The same performance improvement can be accomplished by using a single file created on a RAID (redundant array of independent disks) stripe set. However, files and filegroups let you easily add new files to new disks.

<table><tr><th>Filegroup</th><th>Description</th></tr><tr><td>Primary</td><td>The filegroup that contains the primary file. All system tables are allocated to the primary filegroup.</td></tr><tr><td>User-defined</td><td>Any filegroup that is specifically created by the user when the user first creates or later modifies the database.</td></tr></table>
Default Filegroup</td></tr>
</table>

<table>
<tr><td></td><td>When objects are created in the database without specifying which filegroup they belong to, they are assigned to the default filegroup. At any time, exactly one filegroup is designated as the default filegroup. The files in the default filegroup must be large enough to hold any new objects not allocated to other filegroups.
The primary data file is in the primary filegroup and the user-defined filegroup has two secondary data files.

<u>Feature Comparison:</u>

<table>
<tr><th>Oracle</th><th>SQL Server</th></tr>
<tr><td>Datafile of Tablespace</td><td>Data Files (Primary File Group And Secondary File Group)</td></tr>
<tr><td>Online Redo Log Files</td><td>Transaction Log Files</td></tr>
</table>
</td></tr>
<tr><td>Migration Approach</td><td>

Filegroups in SQL Server are like tablespaces in Oracle. They are used to logically group storage.
When a database is created in SQL Server, it has one filegroup by default.
Based on the number of tablespaces that the schema uses, additional filegroups must be created in the newly created database.
Every database is created with a default primary filegroup that cannot be renamed or dropped.
Thus, one of the tablespaces must map to the primary filegroup during migration. Additional secondary or user filegroups can be created with user-specified names.
In Oracle, tablespaces are created as locally managed or dictionary managed.
The type of tablespace determines what storage parameters can be used. Creating filegroups is like creating tablespaces under Oracle except that a filegroup is added to a specific SQL Server database.
Filegroups added to a database are called secondary filegroups, and they can be added to any database. Datafiles are added separately from the definition of a filegroup.

In SQL Server, you can use <code>CREATE DATABASE</code> statement to create a database and assign <code>PRIMARY</code> filegroup to store the database data.
A filegroup in SQL Server is like tablespaces in Oracle, it is a logical storage for table and index data that can contain one or multiple OS files.

1) Create database and define the default PRIMARY filegroup to store the data
<pre>CREATE DATABASE sales
   ON PRIMARY
    (NAME = sales_data, FILENAME =
'C:\MSSQLData\sales_data.ndf', SIZE = 3MB);</pre>
2) When a table is created in the sales database, it is stored in the PRIMARY filegroup (sales_data) by default.
</td></tr>
</table>

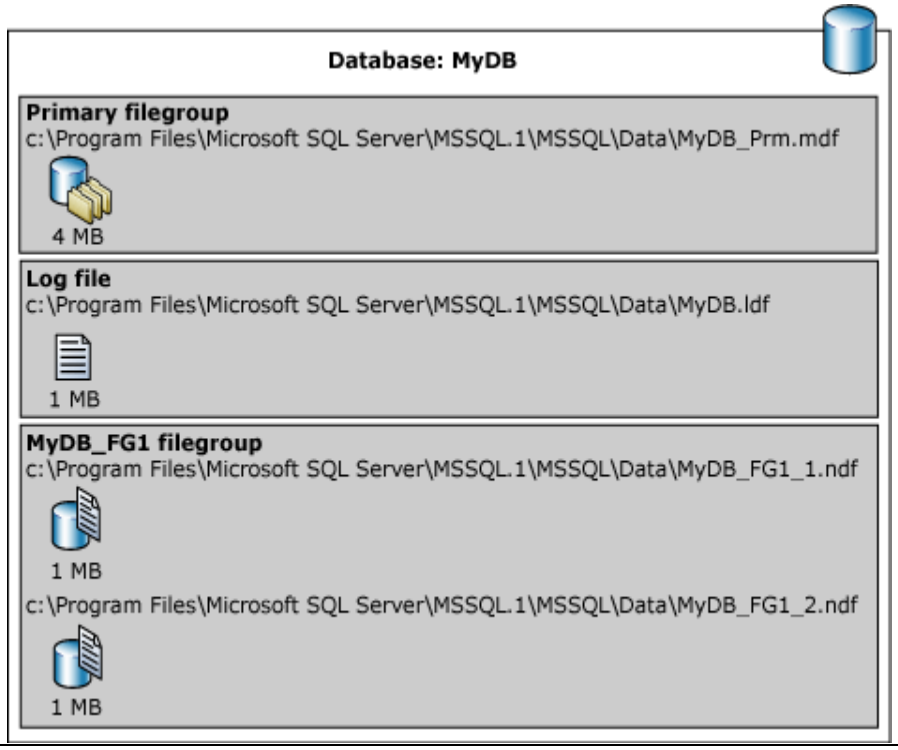
```
CREATE TABLE sales..regions   (
       id INT,
  name VARCHAR(90)
   );
```

In SQL Server, you may also have a separate filegroup to store indexes and LOB data.

```
-- Add a filegroup to the database
  ALTER DATABASE sales ADD FILEGROUP sales_idx;

  -- Add a OS file to the filegroup
  ALTER DATABASE sales ADD FILE (NAME = sales_idx, FILENAME =
'C:\MSSQLData\sales_idx.mdf', SIZE = 3MB)
      TO FILEGROUP sales_idx;

  -- Create an index and store it in sales_idx filegroup
   CREATE UNIQUE INDEX regions_id_idx ON sales..regions (id)
ON sales_idx;
```



Performance Recommendations	• As a best practice, you should place the data and log files on different drives. • Because SQL Server writes all the database transactions into the transaction log, the log files benefit from being on drives with high write performance. • Although not everyone agrees with this recommendation, another best practice for SQL Server databases is to enable AutoGrow. When you enable this property for a database, the database's data and log files will automatically grow if more space is required. This setting prevents the system from stopping if no space is available.

	• With that said, AutoGrow should be considered a last ditch safety mechanism. It shouldn't be used as your primary method to manage database growth. You should manually manage the growth of all data and log files. • Databases activity halts while the database experiences AutoGrow events. Frequent AutoGrow events can lead to disk fragmentation and reduced performance. Nevertheless, AutoGrow is a good safety measure for unexpected data growth. • Disable AutoShrink property for a database. • AutoShrink events cause all database activity to halt. Plus, you can't control when AutoShrink runs. • Another storage best practice is to enable Instant File Initiation. • Unlike most of the other configuration settings discussed here, Instant File Initialization is controlled with a Windows Server policy. • Instant File Initialization avoids zeroing out allocated space for a file. It simply allocates the required space. SQL Server uses Instant File Initialization during database creation, AutoGrow, and database restore operations. • The RAID levels have a big impact on both performance and availability. As you might expect, the more expensive options typically offer the best performance and availability. The most common RAID levels that you'll encounter are: • RAID 0 (sometimes called disk striping). This RAID level spreads all the data across all the available disks. You often see this RAID level used in different database benchmarks. RAID 0 provides good performance, but you should never use it on a production server because one disk failure will result in data loss. • RAID 1 (sometimes called disk mirroring). With RAID 1, data is mirrored on the disks. Read and write performance is good, but the overall disk capacity is cut in half. RAID 1 is often used for SQL Server log files. It can sustain one disk failure. • RAID 5 (sometimes called disk striping with parity). RAID 5 stripes data across multiple disks and uses a disk for data redundancy. It's often used for data files. This RAID level provides good read performance and can sustain the failure of one disk. However, it's known for slower write performance. • RAID 10 (sometimes called disk mirroring with striping). RAID 10 combines the performance of striping with the protection of mirroring. RAID 10 provides the highest levels of performance and availability out of the different RAID levels. RAID 10 uses twice as many disks as RAID 5, but it can sustain multiple disk failures. A RAID 10 array can sustain failures for as many as half of the disks in the set. RAID 10 is good for both data and log files.

References	https://docs.oracle.com/cd/B28359_01/server.111/b28321/strms_fgmon.htm#CFFFFHAH https://msdn.microsoft.com/en-us/library/ms189563.aspx http://www.sqlines.com/oracle_schema_sql_server_database

Functions

<table>
<tr><td>Feature ID</td><td>63</td></tr>
<tr><td>Feature</td><td>Functions</td></tr>
<tr><td>Description</td><td>SQL functions are built into Oracle Database and are available for use in various appropriate SQL statements. Do not confuse SQL functions with user-defined functions written in PL/SQL.

If you call a SQL function with an argument of a datatype other than the datatype expected by the SQL function, then Oracle attempts to convert the argument to the expected datatype before performing the SQL function. If you call a SQL function with a null argument, then the SQL function automatically returns null. The only SQL functions that do not necessarily follow this behavior are CONCAT, NVL, REPLACE, and REGEXP_REPLACE.</td></tr>
<tr><td>Category</td><td>Platform</td></tr>
<tr><td>To Find Feature Enablement</td><td>SELECT count(*) FROM ALL_OBJECTS WHERE OBJECT_TYPE IN ('FUNCTION')</td></tr>
<tr><td>Feature Usage</td><td>SELECT * FROM DBA_OBJECTS WHERE OBJECT_TYPE IN ('FUNCTION')</td></tr>
<tr><td>Recommendation</td><td><u>Feature Description:</u>
These functions are created by user in system database or in user defined database. There are three types of user-defined functions.
Scalar Functions.
Inline Table-Valued Function
Multi-Statement Table-Valued Function.
<u>Feature Comparison:</u>
While Oracle functions closely resemble Transact-SQL functions, significant differences do exist. The main difference is that Transact-SQL functions cannot contain DML statements and cannot invoke stored procedures.
In addition, Transact-SQL functions do not support transaction-management commands. These are stiff restrictions.
A workaround implements a function body as a stored procedure and invokes it within the function by means of an extended procedure. Note that some Oracle function features, such as output parameters, are not currently supported.
<table>
<tr><th></th><th>Oracle</th><th>SQL</th></tr>
<tr><td>PL/SQL user defined type is converted to user defined table type</td><td>CREATE TYPE person_ot AS
OBJECT (
firstname VARCHAR(100),
lastname VARCHAR(100),
hiredate DATE
);</td><td>CREATE TYPE person_ot AS
TABLE
(
rowid uniqueidentifier
DEFAULT NEWID(),
firstname VARCHAR(100),
lastname VARCHAR(100),
hiredate DATETIME2,</td></tr>
</table>
</td></tr>
</table>

			PRIMARY KEY (rowid))
	Oracle variable declared as user defined type is converted to sql server variable of user defined table type	DECLARE person_var person_ot;	DECLARE @person_var person_ot

	Input argument as user defined type will be converted to sql server table value parameter (TVP)	CREATE PROCEDURE showname(person_in IN person_ot, fullname OUT VARCHAR2)
		IS
		BEGIN
		Fullname := person_in.firstname || ' ' person_in.lastname;
		END;
			CREATE PROCEDURE showname(@person_in person_ot READONLY, @fullname VARCHAR(200))
			AS
			SELECT @fullname = firstname + ' ' + lastname FROM @person_in;

	Output argument as user defined type is converted to retuned data set	CREATE OR REPLACE PROCEDURE createperson (firstname IN VARCHAR2, lastname in VARCHAR2, person_out OUT person_ot)
		IS
		BEGIN
		person_out := person_ot(firstname, lastname, SYSDATE);
		END;
			CREATE PROCEDURE createperson (@firstname VARCHAR(100), @lastname VARCHAR(100))
			AS
			DECLARE @person_out person_ot
			INSERT INTO @person_out (firstname, lastname, hiredate) VALUES (@firstname, @lastname, GETDATE())
			— return the object type output as result set
			SELECT * FROM @person_out
			GO

	Object table is created out of schema definition of the user defined table type	CREATE TABLE obtblperson OF person_ot;	DECLARE @person_ot person_ot SELECT * INTO obtblperson FROM @person_ot
	Oracle table column with user defined type is converted into a seperate table	CREATE TABLE tblemployee_ot (id NUMBER, employee person_ot, role VARCHAR2(100), CONSTRAINT tblemployee_ot_pk PRIMARY KEY (id));	CREATE TABLE tblemployee_ot (id INT, employee uniqueidentifier, role VARCHAR(100), CONSTRAINT tblemployee_ot_pk PRIMARY KEY (id));
	Member method is converted into procedure or function	CREATE OR REPLACE TYPE BODY person_ot AS MEMBER PROCEDURE update_hiredate (SELF IN	ALTER TYPE person_ot ADD MEMBER PROCEDURE update_hiredate (SELF IN OUT NOCOPY person_ot);

<table>
<tr><td></td><td><table>
<tr><td></td><td>OUT NOCOPY person_ot) IS
BEGIN
SELF.hiredate :=
SYSDATE;
END;
END;</td><td>CREATE PROCEDURE
person_ot$proc_update_hiredate
(@person_ot person_OT
READONLY)
AS
DECLARE @person_ot_out
person_ot
INSERT INTO @person_ot_out
SELECT * FROM @person_ot
UPDATE @person_ot_out SET
hiredate = getdate()
SELECT * FROM
@person_ot_out
GO</td></tr>
<tr><td>Constructor method is converted into procedure</td><td>CONSTRUCTOR FUNCTION
person_ot
(
firstname IN VARCHAR2,
lastname IN VARCHAR2
)
RETURN SELF AS RESULT
IS
BEGIN
SELF.firstname := firstname;
SELF.lastname := lastname;
SELF.hiredate := SYSDATE;
END;</td><td>CREATE PROCEDURE
person_ot$constructor
(@firstname VARCHAR(100),
@lastname VARCHAR(100))
DECLARE @self person_ot
INSERT INTO @self (firstname,
lastname, hiredate)
VALUES (@firstname,
@lastname, GETDATE())
SELECT * FROM @self;
GO</td></tr>
</table></td></tr>
<tr><td>Migration Approach</td><td>The general format of an Oracle user-defined function is:

FUNCTION [schema.]name [({@parameter_name [IN | OUT | IN OUT]
[NOCOPY] [type_schema_name.] parameter_data_type [:= | DEFAULT]
default_value } [,...n]
)]
RETURN <return_data_type>
[AUTHID {DEFINER | CURRENT_USER}]
[DETERMINISTIC]
[PARALLEL ENABLE ...]
[AGGREGATE | PIPELINED]
{ IS | AS } { LANGUAGE { Java_declaration | C_declaration } | {
[<declaration statements>]
BEGIN
<executable statements>
RETURN <return statement>
[EXCEPTION
exception handler statements]
END [name]; }}

The proper Transact-SQL format of a scalar function is:
CREATE FUNCTION [schema_name.] function_name</td></tr>
</table>

	([{ @parameter_name [AS][type_schema_name.] parameter_data_type [= default_value] } [,...n]]) RETURNS <return_data_type> [WITH { EXEC | EXECUTE } AS { CALLER | OWNER }] [AS] BEGIN <function_body> RETURN <scalar_expression> END [;] The following clauses and arguments are not supported by SSMA and are ignored during conversion: • AGGREGATE • DETERMINISTIC • LANGUAGE • PIPELINED • PARALLEL_ENABLE For the remaining function options, the following rules are applied during conversion: • The OUT qualifier is used when a function is implemented as a procedure. • The [:= | DEFAULT] option of a function parameter is converted to an equals sign (=). • The AUTHID clause is converted to an EXECUTE AS clause. • The CURRENT_USER argument is converted to a CALLER argument. The DEFINER argument is converted to an OWNER argument.
References	http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions001.htm https://msdn.microsoft.com/en-in/library/ms186755.aspx

Instead of Triggers

Feature ID	74
Feature	Instead of Triggers
Description	INSTEAD OF triggers provide a transparent way of modifying views that cannot be modified directly through DML statements (INSERT, UPDATE, and DELETE). These triggers are called INSTEAD OF triggers because, unlike other types of triggers, Oracle fires the trigger instead of executing the triggering statement. You can write normal INSERT, UPDATE, and DELETE statements against the view and the INSTEAD OF trigger is fired to update the underlying tables appropriately. INSTEAD OF triggers are activated for each row of the view that gets modified.
Category	Platform
To Find Feature Enablement	--To view all the triggers present on a database Select * from DBA_TRIGGERS where Trigger_Type='INSTEAD OF' --To view triggers present on a table SELECT * FROM USER_TRIGGERS WHERE TABLE_NAME = 'NAME_OF_YOUR_TABLE';
Feature Usage	Select * from DBA_TRIGGERS where Trigger_Type='INSTEAD OF'
Recommendation	**Feature Description:** INSTEAD OF triggers override the standard actions of the triggering statement: an INSERT, UPDATE, or DELETE. An INSTEAD OF trigger can be defined to perform error or value checking on one or more columns, and then perform additional actions before inserting the record. INSTEAD OF triggers can be defined on either tables or views; however, INSTEAD OF triggers are most useful for extending the types of updates a view can support. For example, INSTEAD OF triggers can provide the logic to modify multiple base tables through a view or to modify base tables that contain the following columns: • **timestamp** data type • Computed columns • Identity columns **Feature Comparison:** • The SQL Server **INSTEAD OF** triggers are equivalent to Oracle's **BEFORE** triggers. • Combine multiple INSTEAD OF triggers that are defined on the same event into one trigger. INSTEAD OF trigger statements are implicitly activated for each row. • INSTEAD OF triggers on Oracle views remain INSTEAD OF triggers

Functionality of Oracle Triggers Mapped to SQL Server

Trigger Feature	Oracle	SQL Server
DML – INSERT	Yes	Yes
DML – UPDATE	Column/Row	Row
DML – DELETE	Yes	Yes
Timing – BEFORE	Yes	Yes (INSTEAD OF)
Timing – AFTER	Yes	Yes
Level	Row/Statement	Row
Views – INSTEAD OF	Yes	Yes
Multiple triggers per action	Yes	Yes (first/last specified)

Migration Approach

Pattern for INSTEAD OF UPDATE triggers and INSTEAD OF DELETE triggers

```
CREATE
TRIGGER [schema. ]INSTEAD_OF_UPDATE_ON_VIEW_<table> ON <table>
INSTEAD OF {UPDATE | DELETE}
AS
/* beginning of trigger implementation */
SET NOCOUNT ON
/* column variables declaration */
DECLARE
/*if the trigger has no references to :OLD that define one variable to store first column. Else define only columns that have references to :OLD*/
@column_old_value$1 <COLUMN_1_TYPE>
@column_old_value$X <COLUMN_X_TYPE>,
@column_old_value$Y <COLUMN_Y_TYPE>,
...
/*define columns to store references to :NEW*/
@column_new_value$A <COLUMN_A_TYPE>,
@column_new_value$B <COLUMN_B_TYPE>,
...
/* iterate for each for from inserted/updated table(s) */
/* For trigger for UPDATE event that has references to :NEW define and open cursor from inserted as well*/
DECLARE ForEachInsertedRowTriggerCursor CURSOR LOCAL FORWARD_ONLY READ_ONLY FOR
SELECT <COLUMN_A_NAME>, <COLUMN_B_NAME> ... FROM inserted
OPEN ForEachInsertedRowTriggerCursor
```

	FETCH NEXT FROM ForEachInsertedRowTriggerCursor INTO @column_new_value$A, @column_new_value$B ... DECLARE ForEachDeletedRowTriggerCursor CURSOR LOCAL FORWARD_ONLY READ_ONLY FOR SELECT <COLUMN_X_NAME>, <COLUMN_Y_NAME> ... FROM deleted OPEN ForEachDeletedRowTriggerCursor FETCH NEXT FROM ForEachDeletedRowTriggerCursor INTO /* trigger has no references to :OLD*/ @column_old_value$1 /* trigger has references to :OLD*/ @column_old_value$X, @column_old_value$Y ... WHILE @@fetch_status = 0 BEGIN ---------- /* Oracle-trigger INSTEAD OF UPDATE/DELETE trigger_1 implementation: begin */ BEGIN < INSTEAD OF UPDATE/DELETE trigger_1 BODY> END /* Oracle-trigger INSTEAD OF UPDATE/DELETE trigger_1 implementation: end */ /* Oracle-trigger INSTEAD OF UPDATE/DELETE trigger_2 implementation: begin */ BEGIN < INSTEAD OF UPDATE/DELETE trigger_1 BODY> END /* Oracle-trigger INSTEAD OF UPDATE/DELETE trigger_2 implementation: end */ ... ---------- /*Only for trigger for UPDATE event that has references to :NEW*/ FETCH NEXT FROM ForEachInsertedRowTriggerCursor INTO @column_new_value$A, @column_new_value$B ... OPEN ForEachDeletedRowTriggerCursor FETCH NEXT FROM ForEachDeletedRowTriggerCursor INTO /* trigger has no references to :OLD*/ @column_old_value$1 /* trigger has references to :OLD*/ @column_old_value$X, @column_old_value$Y ... END /*Only for trigger for UPDATE event that has references to :NEW*/ CLOSE ForEachInsertedRowTriggerCursor DEALLOCATE ForEachInsertedRowTriggerCursor CLOSE ForEachDeletedRowTriggerCursor DEALLOCATE ForEachDeletedRowTriggerCursor /* end of trigger implementation */ <u>Pattern for INSTEAD OF INSERT triggers</u> INSTEAD OF triggers are converted in the same way as DELETE and UPDATE triggers, except the iteration for each row is made with the inserted table.

	CREATE TRIGGER [schema.]INSTEAD_OF_INSERT_ON_VIEW_<table> ON <table> INSTEAD OF INSERT AS /* beginning of trigger implementation */ SET NOCOUNT ON /* column variables declaration */ DECLARE /*if the trigger has no references to :NEW that define one variable to store first column. Else define only columns that have references to :NEW*/ @column_new_value$1 <COLUMN_1_TYPE> @column_new_value$X <COLUMN_X_TYPE>, @column_new_value$Y <COLUMN_Y_TYPE>, ... /*define columns to store references to :OLD */ @column_old_value$A <COLUMN_A_TYPE>, @column_old_value$B <COLUMN_B_TYPE>, ... /* iterate for each for from inserted/updated table(s) */ DECLARE ForEachInsertedRowTriggerCursor CURSOR LOCAL FORWARD_ONLY READ_ONLY FOR SELECT <COLUMN_X_NAME>, <COLUMN_Y_NAME> ... FROM inserted OPEN ForEachInsertedRowTriggerCursor FETCH NEXT FROM ForEachDeletedRowTriggerCursor INTO /* trigger has no references to :NEW*/ @column_new_value$1 /* trigger has references to :NEW*/ @column_new_value$X, @column_new_value$Y ... WHILE @@fetch_status = 0 BEGIN ---------- /* Oracle-trigger INSTEAD OF INSERT trigger_1 implementation: begin */ BEGIN < INSTEAD OF INSERT trigger_1 BODY> END /* Oracle-trigger INSTEAD OF INSERT trigger_1 implementation: end */ /* Oracle-trigger INSTEAD OF INSERT trigger_2 implementation: begin */ BEGIN < INSTEAD OF INSERT trigger_1 BODY> END /* Oracle-trigger INSTEAD OF INSERT trigger_2 implementation: end */ ---------- OPEN ForEachInsertedRowTriggerCursor FETCH NEXT FROM ForEachDeletedRowTriggerCursor INTO /* trigger has no references to :NEW*/ @column_new_value$1 /* trigger has references to :NEW*/ @column_new_value$X, @column_new_value$Y ... END CLOSE ForEachInsertedRowTriggerCursor

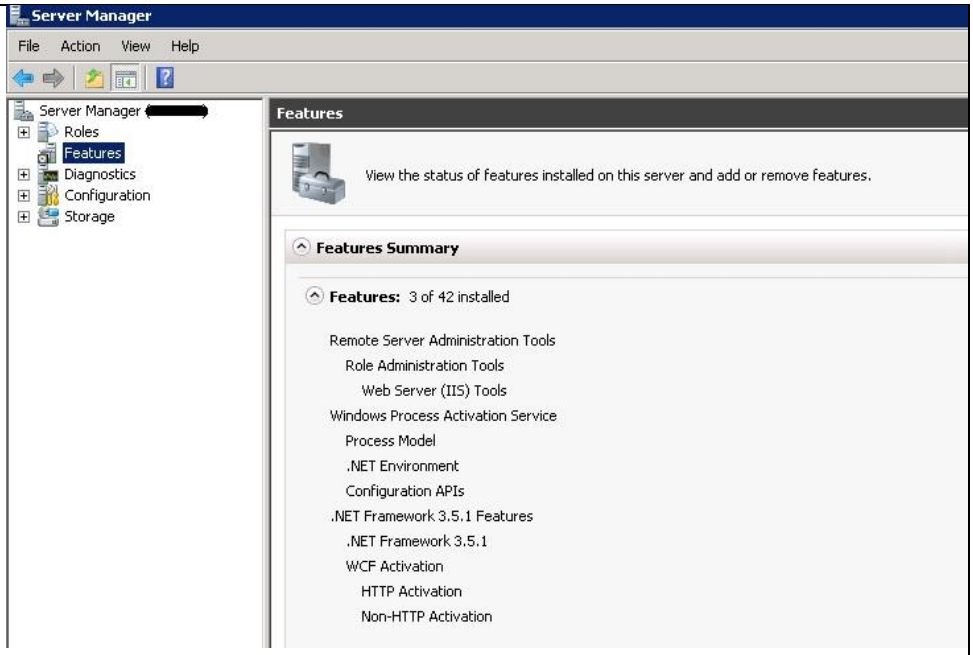
	DEALLOCATE ForEachInsertedRowTriggerCursor /* end of trigger implementation */ Below query gives a list of triggers in the database. SELECT [so].[name] AS [trigger_name], USER_NAME([so].[uid]) AS [trigger_owner], USER_NAME([so2].[uid]) AS [table_schema], OBJECT_NAME([so].[parent_obj]) AS [table_name], OBJECTPROPERTY([so].[id], 'ExecIsUpdateTrigger') AS [isupdate], OBJECTPROPERTY([so].[id], 'ExecIsDeleteTrigger') AS [isdelete], OBJECTPROPERTY([so].[id], 'ExecIsInsertTrigger') AS [isinsert], OBJECTPROPERTY([so].[id], 'ExecIsAfterTrigger') AS [isafter], OBJECTPROPERTY([so].[id], 'ExecIsInsteadOfTrigger') AS [isinsteadof], OBJECTPROPERTY([so].[id], 'ExecIsTriggerDisabled') AS [disabled] FROM sysobjects AS [so] INNER JOIN sysobjects AS so2 ON so.parent_obj = so2.Id WHERE [so].[type] = 'TR'
References	https://docs.oracle.com/cd/B10500_01/server.920/a96524/c18trigs.htm https://technet.microsoft.com/en-us/library/ms179288(v=sql.105).aspx

Sample Scan

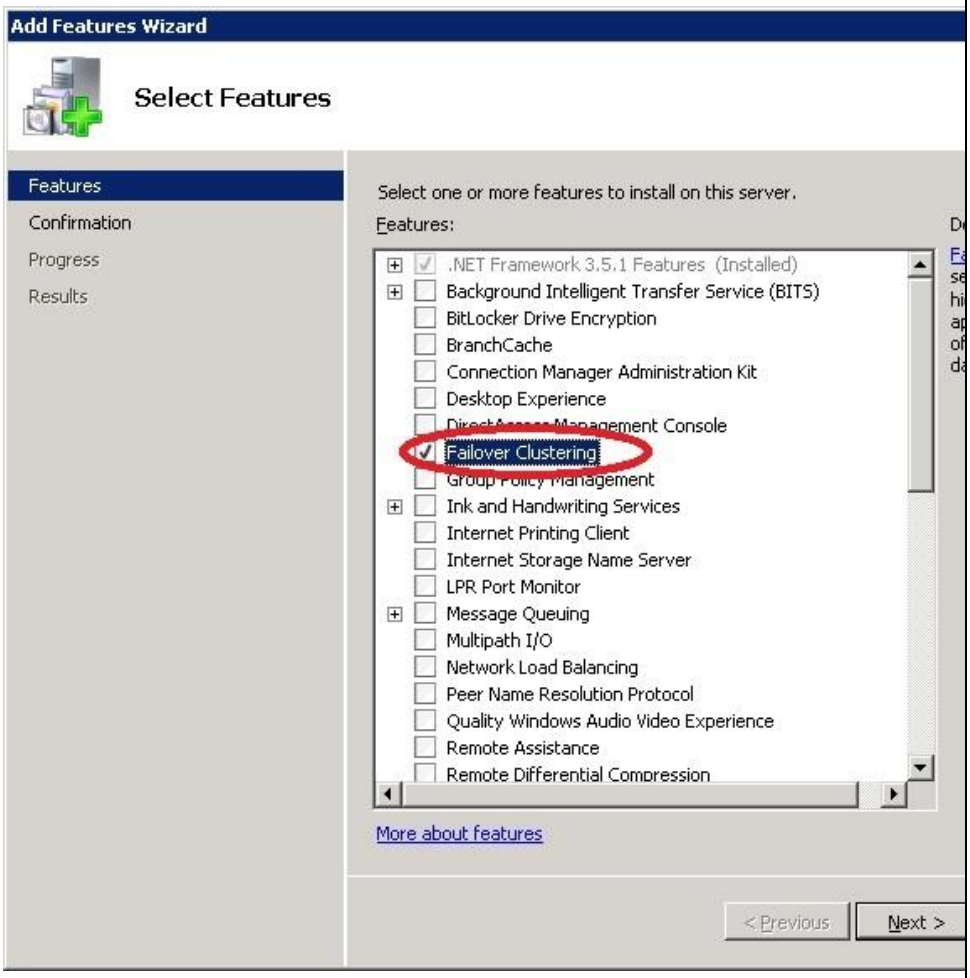
Feature ID	76
Feature	Sample Scan
Description	A sample table scan retrieves a random sample of data from a simple table or a complex SELECT statement, such as a statement involving joins and views. •This access path is used when a statement's FROM clause includes the SAMPLE clause or the SAMPLE BLOCK clause. •To perform a sample table scan when sampling by rows with the SAMPLE clause, Oracle reads a specified percentage of rows in the table. •To perform a sample table scan when sampling by blocks with the SAMPLE BLOCK clause, Oracle reads a specified percentage of table blocks.
Category	Platform
To Find Feature Enablement	
Feature Usage	○ SELECT * FROM (TABLE_NAME) SAMPLE BLOCK (1);
Recommendation	
Migration Approach	
References	

Transparent Application Failover

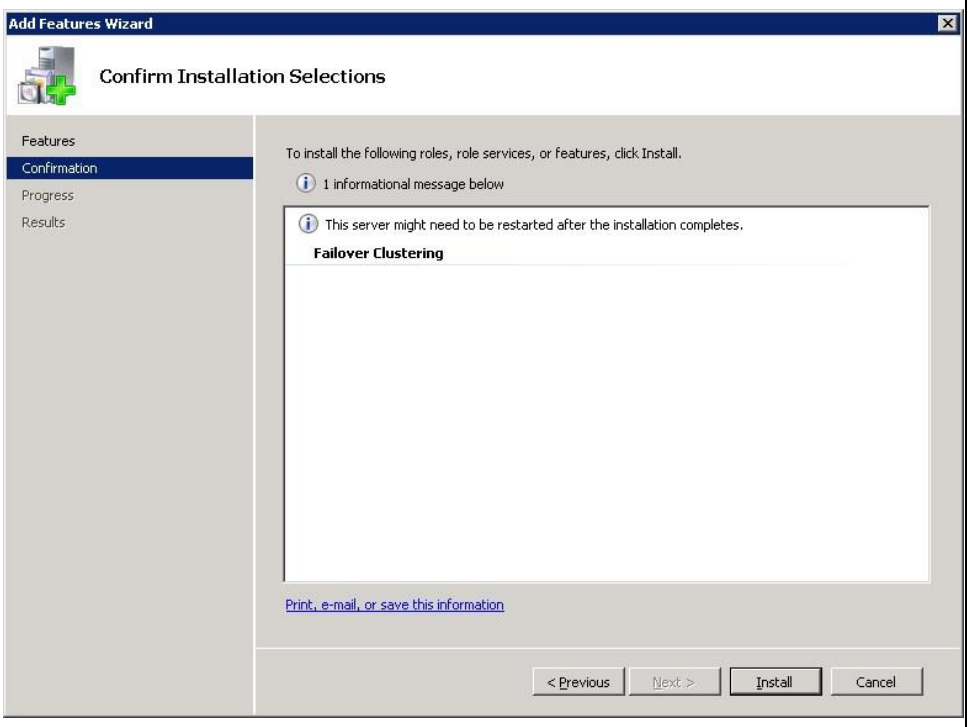
Feature ID	77
Feature	Transparent Application Failover
Description	Transparent Application Failover (TAF) is a feature of the Java Database Connectivity (JDBC) Oracle Call Interface (OCI) driver. It enables the application to automatically reconnect to a database, if the database instance to which the connection is made fails. In this case, the active transactions roll back. When an instance to which a connection is established fails or is shutdown, the connection on the client side becomes stale and would throw exceptions to the caller trying to use it. TAF enables the application to transparently reconnect to a preconfigured secondary instance creating a fresh connection, but identical to the connection that was established on the first original instance. That is, the connection properties are the same as that of the earlier connection. This is true regardless of how the connection was lost.
Category	Platform
To Find Feature Enablement	
Feature Usage	`Select username, sid, serial#,failover_type, failover_method, failed_over from v$session where username not in ('SYS','SYSTEM','PERFSTAT') and failed_over = 'YES';`
Recommendation	**Feature Description** A *Windows Server Failover Clustering* (WSFC) cluster is a group of independent servers that work together to increase the availability of applications and services. SQL Server 2016 takes advantage of WSFC services and capabilities to support Always On Availability Groups and SQL Server Failover Cluster Instances. **Feature Comparison** SQL Server relies on Windows Failover Cluster to perform failovers during any timeout conditions
Migration Approach	We need to configure Transparent Application Failover separately on SQL Server and hence this feature will not be a part of migration from SSMA perspective. **To set up Failover Clustering on Windows Server** Step1 : On the server manager window click on “Add Features”.



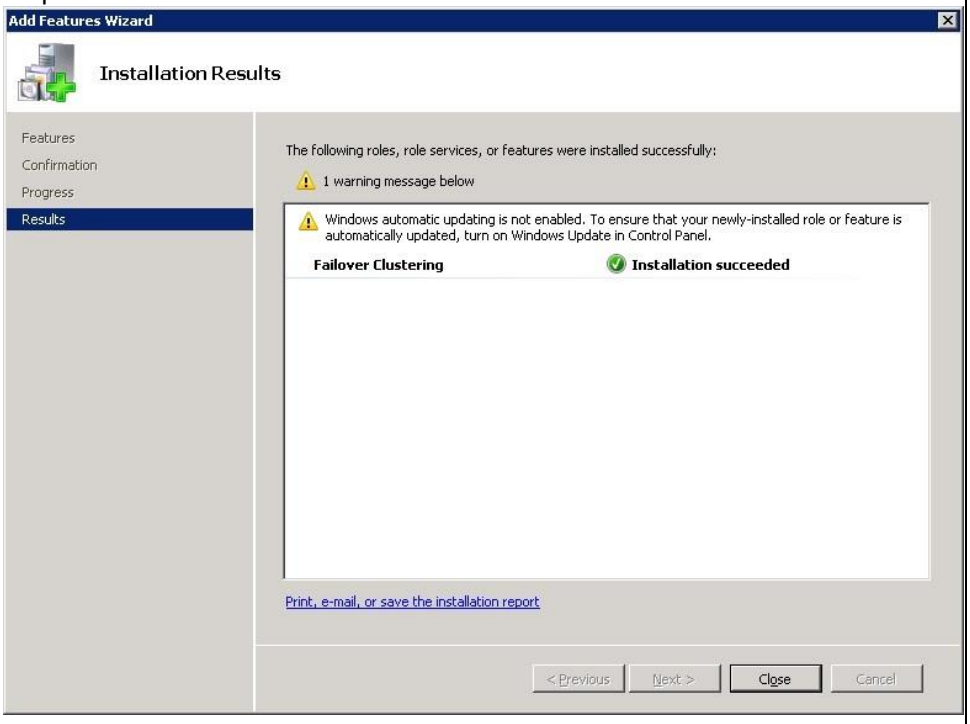
Step2 : On the Add Feature Wizard select “Failover Clustering” and click on Next.



Step3 : Confirm and click on Install.



Step4 : Check installed feature and click “Close” button to exit.



References	https://docs.oracle.com/cd/B19306_01/java.102/b14355/ocitaf.htm#BABCFEBH https://msdn.microsoft.com/en-us/library/hh270278.aspx (Windows Server Failover Clustering (WSFC) with SQL Server)

Export Transportable Tablespaces

Feature ID	81
Feature	Export transportable tablespaces
Description	• Transportable tablespaces were introduced in Oracle 8i to allow whole tablespaces to be copied between databases in the time it takes to copy the datafiles. • In Oracle 8i one of the restrictions was that the block size of both databases must be the same. In Oracle 9i the introduction of multiple block sizes has removed this restriction.
Category	Platform
To Find Feature Enablement	
Feature Usage	--For cross-platform transport, check the endian format of both platforms by querying the V$TRANSPORTABLE_PLATFORM view. Select * from V$TRANSPORTABLE_PLATFORM --Determine if Platforms are Supported and Determine Endianness SELECT d.PLATFORM_NAME, ENDIAN_FORMAT FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
Recommendation	**Feature Description-** SQL Server does not provide similar feature compared to Tramsportable Tablespaces in Oracle. **Feature Comparison:** • Like Oracle's transportable tablespaces feature, SQL Server provides a simple way to move databases from one instance to another. • The move can be a logical and/or physical move and the difference in execution is minimal and the operations can be performed via SQL Server Management Studio or command line using T-SQL. • Do note that different versions/editions of Windows have different levels of functionality so please be aware of your database's requirements prior to moving.
Migration Approach	• To move a SQL Server database from one instance to another, you only need to detach the database from one instance and attach it to another. • If physical movement is necessary, simply move the datafiles and log files associated with that database to the desired new location before attaching to the new instance.
References	https://oracle-base.com/articles/misc/transportable-tablespaces

Materialized Views

Feature ID	82
Feature	Materialized Views
Description	**Materialized view** is a table segment whose contents are periodically refreshed based on a query, either against a local or remote table. Using materialized views against remote tables is the simplest way to achieve replication of data between sites.
Category	Platform
To Find Feature Enablement	select * from all_objects where OBJECT_TYPE='MATERIALIZED VIEW';
Feature Usage	select OBJECT_SCHEMA_NAME(object_id) as [SchemaName], OBJECT_NAME(object_id) as [ViewName], Name as IndexName from sys.indexes where object_id in (select object_id from sys.views)
Recommendation	**Feature Description:** • *Materialized views in SQL Server are called Indexed Views*. • An indexed view has a unique clustered index. The clustered index is stored in SQL Server and updated like any other clustered index, providing SQL Server with another place to look to potentially optimize a query utilizing the indexed view. • Queries that don't specifically use the indexed view can even benefit from the existence of the clustered index from the view. ***Feature Comparison:*** • SQL Server's indexed views are always kept current, Oracle's materialized views can be static. • SQL Server's indexed views are *always* kept up to date. In SQL Server, if a view's base tables are modified, then the view's indexes are also kept up to date in the same atomic transaction. If Oracle's materialized views are created without the REFRESH FAST ON COMMIT option, then the materialized view is *not* modified when its base tables are.
Migration Approach	• While converting a materialized view, SSMA creates necessary unique clustered index on the view in SQL Server and adds WITH SCHEMABINDING option to the CREATE VIEW statement. • The Indexes and Triggers nodes are added as sub nodes to Views in SQL Server Metadata Explorer. • Indexed view is created using the following statements:

	CREATE VIEW <materialized_view_name> WITH SCHEMABINDING AS SELECT ... ; GO CREATE UNIQUE CLUSTERED <index_name> ON <materialized_view_name> (<field1>, <field2> ...); GO The view has to have unique clustered index. Index fields are set of primary keys (or other unique fields/field sets) of participating tables at least. The view must reference only base tables that are in the same database as the view. The view cannot reference other views. SSMA parses SELECT statement of the materialized view DDL definition and determines a degree of compatibility with SQL Server requirements for indexed views. Below is the example of SSMA conversion of materialized views to SQL Server: **Oracle** CREATE MATERIALIZED VIEW PRODUCTS_MV (PROD_ID, PRODUCT_NAME) AS SELECT p.prod_id, p.prod_name FROM products p; **SQL Server** CREATE VIEW dbo.PRODUCTS_MV WITH SCHEMABINDING AS SELECT p.PROD_ID, p.PROD_NAME FROM dbo.PRODUCTS AS p GO IF EXISTS (SELECT * FROM sys.objects so JOIN sys.indexes si ON so.object_id = si.object_id JOIN sys.schemas sc ON so.schema_id = sc.schema_id WHERE so.name = N'PRODUCTS_MV' AND sc.name = N'dbo' AND si.name = N'UIX_PROD_dbo_PRODUCTS_MV_p_PROD_ID' AND so.type in (N'U')) DROP INDEX [dbo].[PRODUCTS_MV].[UIX_PROD_dbo_PRODUCTS_MV_p_PROD_ID] GO CREATE UNIQUE CLUSTERED INDEX [UIX_ATEST_dbo_PRODUCTS_MV_p_PROD_ID] ON [dbo].[PRODUCTS_MV] ([PROD_ID] ASC) WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, IGNORE_DUP_KEY = OFF, ONLINE = OFF) ON [PRIMARY] GO Below query lists the views in the database. select

	OBJECT_SCHEMA_NAME(object_id) as [SchemaName], OBJECT_NAME(object_id) as [ViewName], Name as IndexName from sys.indexes where object_id in (select object_id from sys.views)
References	https://docs.oracle.com/cd/B10501_01/server.920/a96567/repmview.htm https://msdn.microsoft.com/en-us/library/ms190174.aspx

Function based-In Index

Feature ID	91
Feature	Function Based-In Index
Description	Traditionally, performing a function on an indexed column in the where clause of a query guaranteed an index would not be used. Oracle 8i introduced Function-Based Indexes to counter this problem. Rather than indexing a column, you index the function on that column, storing the product of the function, not the original column data. When a query is passed to the server that could benefit from that index, the query is rewritten to allow the index to be used. The following code samples give an example of the use of Function-Based Indexes.
Category	SQL
To Find Feature Enablement	`select count(*) from dba_indexes where index_type like 'FUNCTION-BASED%'`
Feature Usage	`select owner, index_name, index_type from dba_indexes where index_type like 'FUNCTION-BASED%'`

Recommendation	**Feature Description:**
	
	In Oracle, you can create a function-based index that stores precomputed results of a function or expression applied to the table columns. Function-based indexes are used to increase the performance of queries that use functions in the WHERE clause.
	
	-- Create a function-based index that stores names in upper case
	`CREATE INDEX cities_fn_idx ON cities (UPPER(name));`
	
	-- Index range scan will be used instead of expensive full table scan
	`SELECT name FROM cities WHERE UPPER(name) = 'HOUSTON';`
	
	SQL Server does not support function-based indexes, but you can use computed columns and indexes on computed columns to increase the performance of queries that use functions in the WHERE clause.

In SQL Server, you can use a computed column and index defined on the computed column to increase the performance of a query that uses a function in WHERE condition.

Note that for case-insensitive search in SQL Server, you do not need function-based indexes, you have to use case-insensitive collation instead (it is default).

```
 -- Search is case-insensitive in SQL Server by default, no need to use UPPER function
  SELECT name FROM cities WHERE name = 'HOUSTON';

 -- Define a computed column
  CREATE TABLE cities
  (
     ...
     upper_name AS UPPER(name)
  );

  CREATE INDEX cities_fn_idx ON cities (upper_name));
```

Feature Comparison:

In Oracle, you can create a function-based index that stores **precomputed** results of a function or expression applied to the table columns.

Function-based indexes are used to increase the **performance** of queries that use functions in the WHERE clause.

Summary information:

	Oracle	**SQL Server**
Feature	Function-based index	Index on computed column
		Collation for case-insensitive search
Built-in Functions	✓	✓
Expressions	✓	✓
WHERE Clause	No changes required after index added	No changes required after computed column and index are added
INSERT Statement	No changes required	No changes required

Migration Approach	Below is an example to create an function-based index in Oracle and its equivalent in SQL Server.

	Oracle:
	Create a function-based index that stores names in upper case.
	`CREATE INDEX cities_fn_idx ON cities (UPPER(name));`
	
	Index range scan will be used instead of expensive full table scan
	`SELECT name FROM cities WHERE UPPER(name) = 'HOUSTON'`
	
	Note: Here function is used in where clause.
	
	Also note that for **case-insensitive** search in SQL Server, you **do not need** function-based indexes, you must use case-insensitive **collation** instead (which is the default).
	
	SQL Server:
	Search is case-insensitive in SQL Server by default, no need to use UPPER function
	`SELECT name FROM cities WHERE name = 'HOUSTON';`
	
	In other cases, you can use a computed column and index on it:
	Define a computed column
	`CREATE TABLE cities`
	`( ...`
	`upper_name AS UPPER(name)`
	`);`
	
	`CREATE INDEX cities_fn_idx ON cities (upper_name));`

References	https://oracle-base.com/articles/8i/function-based-indexes http://www.sqlines.com/oracle/function_based_indexes

Tablespace Point in Time Recovery

Feature ID	92
Feature	TableSpace Point In Time Recovery
Description	Recovery Manager (RMAN) TSPITR enables quick recovery of one or more tablespaces in a database to an earlier time without affecting the rest of the tablespaces and objects in the database. RMAN TSPITR is most useful for the following situations: • To recover a logical database to a point different from the rest of the physical database, when multiple logical databases exist in separate tablespaces of one physical database. For example, you maintain logical databases in the orders and personnel tablespaces. An incorrect batch job or data manipulation language (DML) statement corrupts the data in only one tablespace. • To recover data lost after data definition language (DDL) operations that change the structure of tables. You cannot use Flashback Table to rewind a table to before the point of a structural change such as a truncate table operation. • To recover a table after it has been dropped with the PURGE option. • To recover from the logical corruption of a table. • To recover dropped tablespaces. In fact, RMAN can perform TSPITR on dropped tablespaces even when a recovery catalog is not used.
Category	Platform
To Find Feature Enablement	
Feature Usage	SELECT * FROM SYS.TS_PITR_CHECK WHERE ('SYSTEM' IN (TS1_NAME, TS2_NAME) AND TS1_NAME <> TS2_NAME AND TS2_NAME <> '-1') OR (TS1_NAME <> 'SYSTEM' AND TS2_NAME = '-1');
Recommendation	**Feature Description:** • A point in time recovery is restoring a database to a specified date and time. When you have completed a point in time recovery, your database will be in the state it was at the specific date and time you identified when restoring your database. A point in time recovery is a method to recover your database to any point in time since the last database backup. • To perform a point in time recovery you will need to have an entire series of backups (complete, differential, and transaction log backups) up to and/or beyond the point in time in which you want to recover.

	• If you are missing any backups, or have truncated the transaction log without first performing a transaction log backup, then you will not be able to perform a point in time recovery. • At a minimum, you will need a complete backup and all the transaction log backups taken following the complete backup. • Optionally if you are taking differential backups, then you will need the complete backup, the last differential backup prior to the corruption, then all the transaction log backups taken following the differential backup. **Feature Comparison:** Oracle and SQL Server provides similar functionality for Point in time backup.
Migration Approach	We need to implement this feature in SQL Server explicitly and hence there will be no migration approach from SSMA perspective.
References	https://docs.oracle.com/cd/E11882_01/backup.112/e10642/rcmtspit.htm#BRADV009 https://technet.microsoft.com/en-us/library/ms190982(v=sql.105).aspx

Table Partitioning

<table>
<tr><td>Feature ID</td><td>12</td></tr>
<tr><td>Feature</td><td>Table Partitioning</td></tr>
<tr><td>Description</td><td>

Partitioning enables tables and indexes to be subdivided into individual smaller pieces.
Each piece of the database object is called a partition. A partition has its own name, and may optionally have its own storage characteristics. From the perspective of a database administrator, a partitioned object has multiple pieces that can be managed either collectively or individually.
This gives the administrator considerable flexibility in managing a partitioned object.
However, from the perspective of the application, a partitioned table is identical to a non-partitioned table; no modifications are necessary when accessing a partitioned table using SQL DML commands.
Logically, it is still only one table and any application can access this one table as they do for a non-partitioned table.

</td></tr>
<tr><td>Category</td><td>Platform</td></tr>
<tr><td>To Find Feature Enablement</td><td><code>select * from v$option where parameter='Partitioning';</code></td></tr>
<tr><td>Feature Usage</td><td><code>SELECT count(*) FROM dba_tab_partitions;</code></td></tr>
<tr><td>Recommendation</td><td>
<u>Feature Description:</u>

SQL Server supports table and index partitioning.
The data of partitioned tables and indexes is divided into units that can be spread across more than one filegroup in a database.
The data is partitioned horizontally, so that groups of rows are mapped into individual partitions.
All partitions of a single index or table must reside in the same database.
The table or index is treated as a single logical entity when queries or updates are performed on the data.

<u>Feature Comparison:</u>
<table>
<tr><th>Oracle</th><th>SQL Server</th></tr>
<tr><td>Oracle allows table partitioning by range, list, hash, or composite (range-hash or range-list).</td><td>SQL Server provide Range and List partitioning</td></tr>
<tr><td>Adding a partition is done with the ALTER TABLE ADD PARTITION which only allows you to add a partition to the "high" end of the range partitioned table.</td><td>To add a partition, you need to change the partition function and scheme</td></tr>
<tr><td>Dropping a partition is done with the ALTER TABLE DROP PARTITION command which results in the data</td><td>Dropping a partition requires the creation of a dummy table and then swapping an</td></tr>
</table>
</td></tr>
</table>

	for that partition being destroyed. As well, the drop will mark any global indexes as invalid causing you to have to rebuild them or you can specify the UPDATE INDEXES clause on the drop which updates the global index as part of the drop	existing partition into that table and then dropping that table.
	When you create the partitioned table you only have what we refer to as the LONG form in terms of range definitions. That is, you must explicitly define the boundaries of each range (at least the high end of the boundary).	First create a partitioning function to define the ranges, then you create a partitioning scheme that defines what 'tablespaces' should be used for which ranges.

Migration Approach

- Migration of Oracle Partitioned Tables is not supported by SSMA.
- Partitioned tables are migrated as a Non-partitioned simple table.
- Partitioning of these Tables in SQL server is required to be done manually as per the physical database architecture planning and logical drives of the server system.
- Any partition maintenance (adding or dropping or truncating the partitions) related code need to be re-rewritten in SQL Server.

Partitioning a table using T-SQL

To create a partitioned table for storing monthly reports we will first create additional filegroups. A filegroup is a logical storage unit. Every database has a primary filegroup that contains the primary data file (.mdf). An additional, user-defined, filegrups can be created to contain secondary files (.ndf). We will create 12 filegroups for every month:

```
ALTER DATABASE PartitioningDB
ADD FILEGROUP January
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP February
GO…
```

To check created and available file groups in the current database run the following query:

```
SELECT name AS AvailableFilegroups
FROM sys.filegroups
WHERE type = 'FG'
```

	AvailableFilegroups
1	PRIMARY
2	January
3	February
4	March
5	April

When filegroups are created we will add .ndf file to every filegroup:

```
ALTER DATABASE [PartitioningDB]
    ADD FILE
    (
    NAME = [PartJan],
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB.ndf',
        SIZE = 3072 KB,
        MAXSIZE = UNLIMITED,
        FILEGROWTH = 1024 KB
    ) TO FILEGROUP [January]
```

To check files created added to the filegroups run the following query:

```
SELECT
name as [FileName],
physical_name as [FilePath]
FROM sys.database_files
where type_desc = 'ROWS'
GO
```

	FileName	FilePath
1	PartitioningDB	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB.mdf
2	PartJan	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB.ndf
3	PartFeb	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB2.ndf
4	PartMar	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB3.ndf

After creating additional filegroups for storing data we'll create a partition function. A partition function is a function that maps the rows of a partitioned table into partitions based on the values of a partitioning column. In this example we will create a partitioning function that partitions a table into 12 partitions, one for each month of a year's worth of values in a `datetime` column:

```
CREATE PARTITION FUNCTION [PartitioningByMonth] (datetime)
AS RANGE RIGHT FOR VALUES ('20140201', '20140301', '20140401',
                '20140501', '20140601', '20140701', '20140801',
                '20140901', '20141001', '20141101', '20141201');
```

To map the partitions of a partitioned table to filegroups and determine the number and domain of the partitions of a partitioned table we will create a partition scheme:

```
CREATE PARTITION SCHEME PartitionBymonth
AS PARTITION PartitioningBymonth
TO (January, February, March,
    April, May, June, July,
    Avgust, September, October,
    November, December);
```

Now we're going to create the table using the `PartitionBymonth` partition scheme, and fill it with the test data:

```
CREATE TABLE Reports
(ReportDate datetime PRIMARY KEY,
MonthlyReport varchar(max))
ON PartitionBymonth (ReportDate);
GO
```

```
INSERT INTO Reports (ReportDate,MonthlyReport)
SELECT '20140105', 'ReportJanuary' UNION ALL
SELECT '20140205', 'ReportFebryary' UNION ALL
SELECT '20140308', 'ReportMarch' UNION ALL
```

We will now verify the rows in the different partitions:

```
SELECT
p.partition_number AS PartitionNumber,
f.name AS PartitionFilegroup,
p.rows AS NumberOfRows
FROM sys.partitions p
JOIN sys.destination_data_spaces dds ON p.partition_number =
dds.destination_id
JOIN sys.filegroups f ON dds.data_space_id = f.data_space_id
WHERE OBJECT_NAME(OBJECT_ID) = 'Reports'
```

PartitionNumber	PartitionFilegroup	NumberOfRows
1	January	1
2	February	1
3	March	1
4	April	1
5	May	1

Now just copy data from your table and rename a partitioned table.

Performance Recommendation	Table partitioning is useful on very large data tables for primarily two reasons. The major reason for partitioning is to gain better management and control of large tables by partitioning them. To gain better management of large tables, you can: • Rebuild and reorganize indexes by partition. • Use partition-aligned indexed views in switching operations. • Use a sliding window strategy for quickly bringing in new data and archiving old data. Additionally, SQL Server's query optimizer can use partition elimination and parallelism to increase appropriately filtered query performance against partitioned tables. To make use of partition elimination: • Ensure that indexes are aligned with the partitioned table, and that indexed views are partition-aligned. • Ensure that queries against the partitioned tables have filters based on the partition column. • On data warehouse joins, keep the join column simple (such as an integer or date) and explicit, to take advantage of bitmap filtering for star joins. In general, to take full advantage of table partitioning, you should: • Make sure that the configuration of max degree of parallelism is set sufficiently high to take advantage of parallel operations, or else add a MAXDOP query hint to fine-tune the degree of parallelism.

	• Maintain an empty partition on both ends of the partitioned table and ensure that only empty partitions are split and merged in a sliding window scenario. • Remember that RANGE RIGHT may be more convenient than RANGE LEFT in partition functions, especially when you are specifying date ranges. • Use data types without fractional components as partition columns, such as a date or an integer. • Always use a standard language-independent date format when specifying partition function boundary values. • Use an integer-based date and date dimension in data warehouses. • Use a single column of the table as the partitioned column whenever possible. If you must partition across more than one column, you can use a persisted computed column as the partitioning column. But then to achieve partition elimination, you must control queries to ensure they reference the partition column in their filters. • Use SWITCH with MERGE to drop partition data: Switch out the partition and remove the partition's boundary value using MERGE. • Use TRUNCATE TABLE to delete partition data by switching a partition out to a staging table and truncating the staging table. • Check for partition elimination in query plans. • Place read only data on read-only filegroups to reduce locking and simplify recovery for piecemeal restores. • Spread filegroups across all disks for maximum I/O performance. • Automate sliding window scenarios using available tools. Below link provides a detailed explanation on Table Partitioning https://technet.microsoft.com/en-us/library/dd578580(v=sql.100).aspx
References	http://www.oracle.com/technetwork/database/options/partitioning/partitioning-wp-12c-1896137.pdf https://msdn.microsoft.com/en-us/library/ms190787.aspx https://social.msdn.microsoft.com/Forums/sqlserver/en-US/a7115324-cb6e-422b-83ff-780dd37bb88a/convert-partitions-from-oracle-to-sql-server-2008-using-ssma-v40?forum=sqltools http://www.sqlshack.com/database-table-partitioning-sql-server/

Flashback Data Archive

Feature ID	93

<table>
<tr><td>Feature</td><td>Flashback Data Archive</td></tr>
<tr><td>Description</td><td>Flashback Data Archive also known as Flashback Archive (FBA), was introduced in Oracle 11g to provide long term storage of undo data, allowing undo-based flashback operations to be performed over an extended period.
Oracle database 12c includes several changes that will allow FDA to reach a wider audience.</td></tr>
<tr><td>Category</td><td>Platform</td></tr>
<tr><td>To Find Feature Enablement</td><td>select count(*) from DBA_FLASHBACK_ARCHIVE_TABLES</td></tr>
<tr><td>Feature Usage</td><td>SELECT * FROM DBA_FLASHBACK_ARCHIVE_TABLES</td></tr>
<tr><td>Recommendation</td><td><u>Feature Description:</u>
SQL Server 2016 system-versioned Temporal Tables provide information about data stored in the table at any point in time.
SQL Server uses a separate history table that tracks changes to the table data. A system-versioned Temporal Table is a new type of user table in SQL Server 2016 designed to keep a full history of data changes and allow easy point-in-time analysis.
This type of Temporal Table is referred to as a system-versioned Temporal Table because the period of validity for each row is managed by the system (i.e. database engine).
<u>Feature Comparison:</u><table><tr><th>Oracle</th><th>SQL Server</th></tr><tr><td>Use the Automatic Undo Management System to obtain metadata and historical data for transactions</td><td>Uses a separate history table that tracks changes to the table data. It is designed to keep a full history of data changes and allow easy point-in-time analysis.</td></tr></table></td></tr>
<tr><td>Migration Approach</td><td>As we know, SQL Server 2016 uses support for system-versioned temporal tables as a database feature that brings built-in support for providing information about data stored in the table at any point in time rather than only the data that is correct at the current moment in time. Temporal is a database feature that was introduced in ANSI SQL 2011 and is now supported in SQL Server 2016. But, SSMA does not copy history data from Oracle Flashback Data Archive tables. You need to manually copy the data during the migration process. Also, SSMA does not display the history table in the SQL Server metadata explorer because it’s treated as a system table — you can see it in SQL Server Management Studio.

SQL Server 2016 does not support all of the Oracle Flashback features, including:
Oracle Flashback Transaction Query
DBMS_FLASHBACK Package</td></tr>
</table>

	• Flashback Transaction • Flashback Data Archive • Flashback Table • Flashback Drop • Flashback Database SSMA also has limitations with some features of Oracle Flashback that need to be handled manually. For example, the CM0536 error indicates that converting a select statement from an Oracle flash table failed, which happens because SQL Server does not not support the System Change Number (SCN) period in Temporal Tables. It relates to Oracle Flashback Query. We can't convert the SCN option in the VERSIONS BETWEEN clause.
References	https://oracle-base.com/articles/12c/flashback-data-archive-fda-enhancements-12cr1 https://msdn.microsoft.com/en-us/library/ms190787.aspx https://social.msdn.microsoft.com/Forums/sqlserver/en-US/a7115324-cb6e-422b-83ff-780dd37bb88a/convert-partitions-from-oracle-to-sql-server-2008-using-ssma-v40?forum=sqltools

Advanced Queue

<table>
<tr><td>Feature ID</td><td>99</td></tr>
<tr><td>Feature</td><td>Advanced Queue</td></tr>
<tr><td>Description</td><td>

When Web-based business applications communicate with each other, producer applications enqueue messages and consumer applications dequeue messages.
Advanced Queuing provides database-integrated message queuing functionality.
Advanced Queuing leverages the functions of the Oracle database so that messages can be stored persistently, propagated between queues on different machines and databases, and transmitted using Oracle Net Services, HTTP(S), and SMTP.
Since Oracle Advanced Queuing is implemented in database tables, all the operational benefits of high availability, scalability, and reliability are applicable to queue data.
Standard database features such as recovery, restart, and security are supported in Advanced Queuing, and queue tables can be imported and exported

</td></tr>
<tr><td>Category</td><td>Platform</td></tr>
<tr><td>To Find Feature Enablement</td><td>SELECT count(*)
FROM dba_queue_tables;</td></tr>
<tr><td>Feature Usage</td><td>SELECT owner, queue_table, type
FROM dba_queue_tables;</td></tr>
<tr><td>Recommendation</td><td>
<u>Feature Description:</u>

SQL server does not have Advanced Queue as a feature, but Service broker is a similar feature to that of Oracle’s Advanced Queue.
Service Broker uses queues to provide loose coupling between the message sender and the message receiver.
The sender can use the SEND command to put a message in a queue and then continue with the application, relying on Service Broker to ensure that the message reaches its destination.
Queues permit a lot of scheduling flexibility. For example, the sender can send out multiple messages for multiple receivers to process in parallel.
The receivers might not process the messages until long after they were sent, but because incoming messages are queued, the receivers can process them at their own rate and the sender doesn't have to wait for the receivers to finish before continuing.

<u>Feature Comparison:</u>
<table>
<tr><th>Oracle</th><th>SQL Server</th></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>OAQ does message transformation, message data based routing, and interfaces to other message systems</td><td>Service Broker doesn't have</td></tr>
<tr><td>Most of the OAQ logic is implemented as stored procedures</td><td>Service Broker logic was compiled into the SQL Server executable as DML and DDL commands</td></tr>
<tr><td>OAQ uses SqlNet for communications</td><td>Service Broker created a new communications channel optimized for reliable message delivery. This enables Service Broker to do things like end to end session based encryption and authentication of persistent sessions so a message that is forwarded through multiple intermediate hops is only encrypted and decrypted once</td></tr>
<tr><td>OAQ supports complex rules for activating internal or external logic to handle a message based on the message contents and headers</td><td>Service Broker's activation is simpler and more efficient primarily because it doesn't support the complexity that OAQ does.</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Migration Approach</td><td>We need to implement this feature in SQL Server explicitly and hence it is not a part of migration from SSMA perspective. However, there are documentations available for service broker in SQL Server 2016 which is a similar feature like advanced queue in oracle. There are different sections for this:
• Data Definition statements: for CREATE, ALTER, and DROP statements
• Service Broker Statements: SQL Server provides various service broker statements such as BEGIN CONVERSATION TIMER, BEGIN DIALOG CONVERSATION, END CONVERSATION, GET CONVERSATION GROUP etc.
• Service Broker Catalog Views: Such as sys.conversation_endpoints, sys.conversation_groups, sys.conversation_priorities (Transact-SQL) etc.
• Service Broker Related Dynamic Management Views: Such as sys.dm_broker_activated_tasks, sys.dm_broker_forwarded_messages etc.
• ssbdisgnose Utiliy : The ssbdiagnose utility reports issues in Service Broker conversations or the configuration of Service Broker services. Configuration checks can be made for either two services or a single service. Issues are reported either in the command prompt window as human-readable text, or as formatted XML that can be redirected to a file or another program.

For more information on different Service broker features, you can use the following url:

https://msdn.microsoft.com/en-GB/Library/bb522893(v=sql.105).aspx</td></tr>
</table>

References	http://docs.oracle.com/cd/B10501_01/appdev.920/a96587/qintro.htm https://www.mssqltips.com/sqlservertip/1836/sql-server-service-broker-example-on-how-to-configure-send-and-receive-messages/ https://msdn.microsoft.com/en-us/library/bb522893.aspx https://blogs.msdn.microsoft.com/rogerwolterblog/2006/05/01/oracle-advanced-queuing/

Event Triggers

Feature ID	100
Feature	Event Triggers

Description	You can use triggers to publish information about database events to subscribers. Applications can subscribe to database events just as they subscribe to messages from other applications. These database events can include:
	
	• System events
	  ○ Database startup and shutdown
	  ○ Data Guard role transitions
	  ○ Server error message events
	• User events
	  ○ User logon and logoff
	  ○ DDL statements (CREATE, ALTER, and DROP)
	  ○ DML statements (INSERT, DELETE, and UPDATE)
	
	Triggers on system events can be defined at the database level or schema level. The DBMS_AQ package is one example of using database triggers to perform certain actions.

Category	SQL
To Find Feature Enablement	SELECT a.obj#, a.sys_evts, b.name FROM sys.trigger$ a, sys.obj$ b WHERE a.sys_evts > 0 AND a.obj#=b.obj# AND baseobject IN (0, 88);
Feature Usage	SELECT a.obj#, a.sys_evts, b.name FROM sys.trigger$ a, sys.obj$ b WHERE a.sys_evts > 0 AND a.obj#=b.obj# AND baseobject IN (0, 88);
Recommendation	**Feature Description:** In SQL Server, also, there are DDL, DML and logon triggers. • Server Level Triggers can be used in SQL Server • Event Notification Creates an object that sends information about a database or server event to a service broker service. • Event notifications are created only by using Transact-SQL statements. • DML triggers is a special type of stored procedure that automatically takes effect when a data manipulation language (DML) event takes place that affects the table or view defined in the trigger. DML events include INSERT, UPDATE, or DELETE statements. DML triggers can be used to enforce business rules and data integrity, query other tables, and include complex Transact-SQL statements. • DDL events can be used to fire a DDL trigger or event notification. These events primarily correspond to Transact-SQL CREATE, ALTER, and DROP statements, and certain system stored procedures that perform

	DDL-like operations. Note that each event corresponds to a Transact-SQL statement or stored procedure, with the statement syntax modified to include an underscore character (_) between keywords. • Logon triggers fire in response to the LOGON event that is raised when a user sessions is being established. Triggers can be created directly from Transact-SQL statements or from methods of assemblies that are created in the Microsoft .NET Framework common language runtime (CLR) and uploaded to an instance of SQL Server. SQL Server allows for creating multiple triggers for any specific statement. **Feature Comparison:** • Server Level Triggers may not support all the options available I • Triggers are processed synchronously, within the scope of the transactions that cause them to fire. • User event are handled in SSMA as a part of migration. • Event notifications may be processed asynchronously and do not run in the scope of the transactions that cause them to fire. • The consumer of a trigger is tightly coupled with the event that causes it to fire. • The consumer of an event notification is decoupled from the event that causes it to fire. • Triggers must be processed on the local server. • Event notifications can be processed on a remote server. • Because BEFORE triggers do not exist in SQL Server as Oracle, it is equivalent to INSTEAD OF triggers in SQL Server. That change requires that the triggering statement be moved into the body of the trigger. Also, all triggers for a specific event should go into one target INSTEAD OF trigger. • The first major difference between Oracle and SQL Server triggers is that the most common Oracle trigger is a row-level trigger (FOR EACH ROW), which fires for each row of the source statement. SQL Server, however, supports only statement-level triggers, which fire only once per statement, irrespective of the number of rows affected. Oracle Row-level triggers are emulated with a cursor loop in SQL Server. • No column sensitive trigger in SQL Server. Sometimes an Oracle trigger is defined for a specific column with the UPDATE OF column [, column]...] clause. •
Migration Approach	• Triggers are migrated as a part of migration • We need to create and event notification explicitly in SQL Server, and hence it not a part of migration from SSMA perscpective. SSMA handles conversion of triggers from Oracle to SQL Server through various pattern changes. • As we know, for Oracle triggers are row level where as for SQL Server it fires only once per statement. irrespective of the number of rows affected. In a row-level trigger, Oracle uses an :OLD alias to refer to column values that existed before the statement executes, and to the

changed values by using a :NEW alias. SQL Server uses two pseudotables, inserted and deleted, which can each have multiple rows. If the triggering statement is UPDATE, a row's older version is present in deleted, and the newer in inserted. But it is not easy to tell which pair belongs to the same row if the updated table does not have a primary key or the primary key was modified. You can resolve this problem only if SSMA generates a special ROWID column for the table.

- The second major difference between Oracle and SQL Server triggers comes from Oracle BEFORE triggers. Because Oracle fires these triggers before the triggering statement, it is possible to modify the actual field values that will be stored in the table, or even cancel the execution of the triggering statement if it is found to be unnecessary. To emulate this in SQL Server, you must create INSTEAD OF triggers. That means you must incorporate the triggering statement into the target trigger's body. Because multiple rows can be affected, SSMA puts the statement in a separate cursor loop.
- In some cases, you cannot convert Oracle triggers to SQL Server triggers with one-to one correspondence. If an Oracle trigger is defined for several events at once (for example, INSERT or UPDATE), you must create two separate target triggers, one for INSERT and one for UPDATE. In addition, because SQL Server supports only one INSTEAD OF trigger per table, SSMA combines the logic of all BEFORE triggers on that table into a single target trigger. This means that triggers are not converted independently of each other; SSMA takes the entire set of triggers belonging to a table and converts them into another set of SQL Server triggers so that the general relation is many-to-many.

In brief, the conversion rules are:

- All BEFORE triggers for a table are converted into one INSTEAD OF trigger.
- AFTER triggers remain AFTER triggers in SQL Server.
- INSTEAD OF triggers on Oracle views remain INSTEAD OF triggers.
- Row-level triggers are emulated with a cursor loop.
- Triggers that are defined for multiple events are split into separate target triggers.

Sometimes an Oracle trigger is defined for a specific column with the UPDATE OF column [, column]...] clause. To emulate this, SSMA wraps the trigger body with the following SQL Server construction:

```
IF (UPDATE(column) [OR UPDATE(column) . . .]
BEGIN
  <trigger body>
END
```

SSMA emulates the trigger-specific functions performing INSERT, UPDATE, and DELETE operations by saving the current trigger type in a variable, and then checking that value. For example:

```
DECLARE @triggerType char(1)
SELECT @triggerType = 'I'    /* if the current type is inserting */
. . .
IF (@triggerType = 'I' ) . . .  /* emulation of INSERTING */
```

	IF (@triggerType = 'U') . . . /* emulation of UPDATING */
	IF (@triggerType = 'D') . . . /* emulation of DELETING */
	
	The UPDATING function can have a column name as an argument. SSMA can convert such usage if the argument is a character literal. In this case, the Oracle expression:
	UPDATING ('column_name')
	
	Is transformed into:
	UPDATE (columns_name)

References	https://docs.oracle.com/cd/B19306_01/server.102/b14220/triggers.htm#i6061 https://msdn.microsoft.com/en-us/library/ms189453.aspx https://technet.microsoft.com/en-us/library/ms189855.aspx

Supplemental Logging

Feature ID	101
Feature	Supplemental Logging
Description	• Redo log files are generally used for instance recovery and media recovery. The data needed for such operations is automatically recorded in the redo log files. • However, a redo-based application may require that additional columns be logged in the redo log files. • The process of logging these additional columns is called **supplemental logging.** • By default, Oracle Database does not provide any supplemental logging, which means that by default LogMiner is not usable. • Therefore, you must enable at least minimal supplemental logging before generating log files which will be analyzed by LogMiner.
Category	Admin
To Find Feature Enablement	SELECT supplemental_log_data_min, supplemental_log_data_pk, supplemental_log_data_ui, supplemental_log_data_fk, supplemental_log_data_all, supplemental_log_data_pl FROM v$database;
Feature Usage	SELECT supplemental_log_data_min, supplemental_log_data_pk, supplemental_log_data_ui, supplemental_log_data_fk, supplemental_log_data_all, supplemental_log_data_pl FROM v$database;
Recommendation	**Feature Description:** SQL Server does not provide an as is feature as that of Supplemental Logging in SQL Server. **Feature Comparison:** • In Oracle difference is that everything gets logged, even the undo information. Redo log files are used just like SQL Server transaction log files. • Like SQL Server's transaction log, Oracle can have multiple redo log files. These log files are written to in a circular fashion – the log files are written to in order and, when all log files are full, Oracle will circle around to the beginning again. • Unlike SQL Server's transaction log, you need to have multiple redo log files in Oracle. You can get by with two log files, but three or more is the preferred way to configure Oracle. • Unlike SQL Server, having multiple redo log files is the preferred way to manage Oracle logging. There are even multiple groups of redo log files, by default: two, but this can and should be configured, based on RPO/RTO needs.

Migration Approach	We don't have a migration approach as it is an independent feature and needs to be implemented individually. However, to enable supplemental logging you can follow the below steps. • **Setting the database to full recovery model**: Oracle requires a SQL Server source database to be set to the full recovery model. To verify or set the recovery model: 1. Connect to the SQL Server instance with SQL Server Management Studio for SQL Server. 2. Expand the Databases folder. 3. Right-click the source database, and then select **Properties**. 4. Select the **Options** tab. 5. Under **Recovery**, set Model to **Full if not already**. 6. If the database was in Simple recovery or never had a Full database backup, take a Full database backup before starting Extract. 7. Click **OK**. • **Backing up the transaction log:** Oracle GoldenGate requires the log backup files on a source system to meet the following conditions: 1. The log backup file must remain in the original location where the backup was made. 2. The backup must be made to a DISK device. 3. The backup must be a native SQL Server backup made by issuing the BACKUP LOG command (or the corresponding GUI command) and can be compressed using the native SQL Server compression features using the native SQL server compression feature (SQL Server 2012 only). Third-party log backup tools are not supported. 4. Do not overwrite backup files to the same name as old ones. 5. Striped log backups are not supported. For optimal performance of the Extract process, do the following: • Make only one log backup per backup file. • **Retaining the log backups:** Retain enough log backups so that if you stop Extract or there is an unplanned outage, Extract can start again from its checkpoints. Extract must have access to the data in the transaction log or a log backup that contains the start of the oldest uncommitted unit of work, and all log backups thereafter. If data that Extract needs during processing is not retained, either in online logs or in the backups, one of the following corrective actions might be required:

1. Alter Extract to capture from a later point in time for which log data is available (and accept possible data loss on the target).
2. Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

- **Enabling supplemental logging:**

This procedure requires a database user who is a member of the SQL Server System Administrators (sysadmin) role.

1. On the source system, run GGSCI.
2. Issue the following command to log into the database.
3. DBLOGIN SOURCEDB *DSN*[, {USERID *user*, PASSWORD *password* | USERIDALIAS *alias*}]

   Where:

   - SOURCEDB *DSN* is the name of the SQL Server data source.
   - USERID *user* is the Extract login and PASSWORD *password* is the password that is required if Extract uses SQL Server authentication. Alternatively, USERIDALIAS *alias* is the alias for the credentials if they are stored in a credentials store. If using DBLOGIN with a DSN that is using Integrated Windows authentication, the connection to the database for the GGSCI session will be that of the user running GGSCI. In order to issue ADD TRANDATA or DELETE TRANDATA, this user must be a member of the SQL Server sysadmin server role.
4. In GGSCI, issue the following command for each table that is, or will be, in the Extract configuration. You can use a wildcard to specify multiple table names.

   ADD TRANDATA *owner.table*

- **Managing the secondary truncation point:** When you enable supplemental logging with the ADD TRANDATA command for at least one table in a SQL Server database, a secondary truncation point is created in the transaction log that has to be moved for log space to be released as needed, following subsequent log backups. Use the TRANLOGOPTIONS parameter to control how the secondary truncation point is managed.

For more information regarding this, you can use the following url:

https://docs.oracle.com/goldengate/1212/gg-winux/GIMSS/log_config.htm#GIMSS233

References	https://docs.oracle.com/goldengate/1212/gg-winux/GIMSS/log_config.htm#GIMSS231 https://docs.oracle.com/database/121/SUTIL/GUID-D2DDD67C-E1CC-45A6-A2A7-198E4C142FA3.htm#SUTIL1583

Block Change Tracking

Feature ID	Block Change Tracking
Feature	106
Description	• Block changing tracking improves the performance of incremental backups by recording changed blocks in the block change tracking file. • During an incremental backup, instead of scanning all data blocks to identify which blocks have changed, RMAN uses this file to identify the changed blocks that need to be backed up. • You can enable block change tracking when the database is either open or mounted. • This section assumes that you intend to create the block change tracking file as an Oracle Managed File in the database area, which is where the database maintains active database files such as data files, control files, and online redo log files.
Category	Admin
To Find Feature Enablement	SELECT count(*) FROM V$BLOCK_CHANGE_TRACKING;
Feature Usage	SELECT status, filename FROM V$BLOCK_CHANGE_TRACKING;
Recommendation	**Feature Recommendation:** SQL Server does not have an as is functionality as that of oracle block change tracking, but it provides similar feature named as file group restore and piecemeal restore. **File Restore:** • In a file restore, the goal is to restore one or more damaged files without restoring the whole database. • A file restore scenario consists of a single restore sequence that copies, rolls forward, and recovers the appropriate data. • If the filegroup that is being restored is read/write, an unbroken chain of log backups must be applied after the last data or differential backup is restored. • This brings the filegroup forward to the log records in the current active log records in the log file. The recovery point is typically near the end of log, but not necessarily. • If the filegroup that is being restored is read-only, usually applying log backups is unnecessary and is skipped. • If the backup was taken after the file became read-only, that is the last backup to restore. Roll forward stops at the target point. **Piecemeal restore** • It allows databases that contain multiple filegroups to be restored and recovered in stages.

	• Piecemeal restore involves a series of restore sequences, starting with the primary filegroup and, in some cases, one or more secondary filegroups. • Piecemeal restore maintains checks to ensure that the database will be consistent in the end. • After the restore sequence is completed, recovered files, if they are valid and consistent with the database, can be brought online directly. Piecemeal restore works with all recovery models, but is more flexible for the full and bulk-logged models than for the simple model. **Feature Comparison:** Oracle and SQL Server has defined block change tracking differently, and hence we can use file restore or piecemeal restore options as a workaround to it.

Migration Approach	follow below steps to perform file restore and piecemeal restore:
	
	Piecemeal Restore of Database (Full Recovery Model)
	A piecemeal restore sequence restores and recovers a database in stages at the filegroup level, beginning with the primary and all read-write, secondary filegroups.
	
	In this example, database adb is restored to a new computer after a disaster. The database is using the full recovery model; therefore, before the restore starts, a tail-log backup must be taken of the database. Before the disaster, all the filegroups are online. Filegroup B is read-only. All of the secondary filegroups must be restored, but they are restored in order of importance: A (highest), C, and lastly B. In this example, there are four log backups, including the tail-log backup.
	
	Tail-Log Backup:
	Before restoring the database, the database administrator must back up the tail of the log. Because the database is damaged, creating the tail-log backup requires using the NO_TRUNCATE option:
	`BACKUP LOG adb TO tailLogBackup WITH NORECOVERY, NO_TRUNCATE`
	
	The tail-log backup is the last backup that is applied in the following restore sequences.
	
	Restore Sequences:
	The syntax for an online restore sequence is the same as for an offline restore sequence.
	
	1. Partial restore of the primary and secondary filegroup A.
	`RESTORE DATABASE adb FILEGROUP='Primary' FROM backup1`
	`WITH PARTIAL, NORECOVERY`
	`RESTORE DATABASE adb FILEGROUP='A' FROM backup2`
	`WITH NORECOVERY`
	`RESTORE LOG adb FROM backup3 WITH NORECOVERY`
	`RESTORE LOG adb FROM backup4 WITH NORECOVERY`

	RESTORE LOG adb FROM backup5 WITH NORECOVERY RESTORE LOG adb FROM tailLogBackup WITH RECOVERY 2. Online restore of filegroup C. At this point, the primary filegroup and secondary filegroup A are online. All the files in filegroups B and C are recovery pending, and the filegroups are offline. Messages from the last RESTORE LOG statement in step 1 indicate that rollback of transactions that involve filegroup C was deferred, because this filegroup is not available. Regular operations can continue, but locks are held by these transactions and log truncation will not occur until the rollback can complete. In the second restore sequence, the database administrator restores filegroup C: RESTORE DATABASE adb FILEGROUP='C' FROM backup2a WITH NORECOVERY RESTORE LOG adb FROM backup3 WITH NORECOVERY RESTORE LOG adb FROM backup4 WITH NORECOVERY RESTORE LOG adb FROM backup5 WITH NORECOVERY RESTORE LOG adb FROM tailLogBackup WITH RECOVERY At this point the primary and filegroups A and C are online. Files in filegroup B remain recovery pending, with the filegroup offline. Deferred transactions have been resolved, and log truncation occurs. 3. Online restore of filegroup B. In the third restore sequence, the database administrator restores filegroup B. The backup of filegroup B was taken after the filegroup became read-only; therefore, it does not have to be rolled forward during recovery. RESTORE DATABASE adb FILEGROUP='B' FROM backup2b WITH RECOVERY All filegroups are now online.
References	https://docs.oracle.com/database/121/ADMQS/GUID-3BAA0D48-CA35-4CD7-810E-50C703DC6FEB.htm https://msdn.microsoft.com/en-us/library/ms177425.aspx (Piecemeal Restores-SQL Server)

Streams, CDC and apply

Feature ID	107
Feature	Streams, CDC and apply
Description	
Category	Admin
To Find Feature Enablement	
Feature Usage	
Recommendation	
Migration Approach	
References	

Automated Maintenance Tasks

Feature ID	3
Feature	Automated Maintenance Tasks
Description	Oracle includes three automated database maintenance tasks: • Automatic Optimizer Statistics Collection - Gathers stale or missing statistics for all schema objects (more info). The task name is 'auto optimizer stats collection'. • Automatic Segment Advisor - Identifies segments that could be reorganized to save space (more info). The task name is 'auto space advisor'. • Automatic SQL Tuning Advisor - Identifies and attempts to tune high load SQL (more info). The task name is 'sql tuning advisor'. These tasks run during maintenance windows scheduled to open over night. Configuration of the maintenance tasks, their schedules and resource usage is possible using Enterprise Manager or PL/SQL APIs.
Category	Admin
Find Feature Enabled	`select * from dba_stmt_audit_opts union select * from dba_priv_audit_opts;`

Feature Usage	`show parameter audit_sys_operations;`
	
	`show parameter audit_trail;`
	`select * from dba_stmt_audit_opts union select * from dba_priv_audit_opts;`
	
	`-- if a non-container database`
	`-- conn / as sysdba`
	
	`-- connect to each PDB in turn and run the following queries`
	
	`show parameter audit`
	
	`SELECT table_name, tablespace_name, num_rows`
	`FROM dba_tables`
	`WHERE table_name IN ('AUD$', 'FGA_LOG$')`
	`ORDER BY 1;`

Recommendation	

Oracle	SQL Server
Automatic Optimizer Statistics Collection	Update Statistics
Automatic Segment Advisor	There is no exact feature available. Database engine tuning advisor could be used for this
Automatic SQL Tuning advisor	Database engine tuning advisor

In SQL Server, automated database maintenance tasks are more elaborate and it can be done for various event. SQL server also allows maintenance plan creation in which workflow defines the architecture of the automated maintenance. SQL Server Integration Services includes a set of tasks that perform database maintenance functions. These tasks are commonly used in database maintenance plans, but the tasks can also be included in SSIS packages. For more information, see Maintenance Plan Wizard and Maintenance Plans. The maintenance tasks can be used with SQL Server 2000 and SQL Server databases and database objects. The following table lists the maintenance tasks.

Task	Description
Back Up Database Task	Performs different types of SQL Server database backups.
Check Database Integrity Task	Checks the allocation and structural integrity of database objects and indexes.
Execute SQL Server Agent Job Task	Runs SQL Server Agent jobs.
Execute T-SQL Statement Task	Runs Transact-SQL statements
History Cleanup Task	Deletes entries in the history tables in the SQL Server msdb database.
Maintenance Cleanup Task	Removes files related to maintenance plans, including reports created by maintenance plans and database backup files.
Notify Operator Task	Sends notification messages to SQL Server Agent operators.
Rebuild Index Task	Rebuilds indexes in SQL Server database tables and views.
Reorganize Index Task	Reorganizes indexes in SQL Server database tables and views.
Shrink Database Task	Reduces the size of SQL Server database data and log files.

	<table><tr><td>Update Statistics Task</td><td>Updates information about the distribution of key values for one or more sets of statistics on the specified table or view.</td></tr></table>
Migration Approach	Automated maintenance tasks cannot be migrated through SSMA tool. Each automated maintenance tasks in Oracle needs to be rewritten into SQL server. As a workaround, this can be achieved through creating a job in SQL server. For creating a job a user must be a member of one of the SQL Server Agent fixed database roles or the **sysadmin** fixed server role. For security reasons, only the job owner or a member of the **sysadmin** role can change the definition of the job. • In the **Object Explorer**, expand the server where you want to create a SQL Server Agent job. • On the general page, general properties of the job such as name, owner, category, master server which would be working as a source are set. • On the Steps page, the job steps can be organized and viewed. A *job* is a specified series of actions that SQL Server Agent performs. Jobs can be used to define an administrative task that can be run one or more times and monitored for success or failure. A job can run on one local server or on multiple remote servers. There are various options such as to list the job steps, to move any particular step up or down. There are also options for editing and deleting a selected job step. • On the Schedules page, schedules for the job can be organized and viewed. A *schedule* specifies when a job runs. More than one job can run on the same schedule, and more than one schedule can apply to the same job. Like steps, there are also options in schedules page to list, create, edit and remove a selected schedule. • On the **Alerts** page, organize the alerts for the job. An *alert* is an automatic response to a specific event. For example, an event can be a job that starts or system resources that reach a specific threshold. The user can define the conditions under which an alert can occurs. An alert can notify one or more operators or run a job. • On the **Notifications** page, set actions for Microsoft SQL Server Agent to perform when the job completes. There are various options such as to send e-mail when the job completes or to send e-mail to an operator's pager when the job completes or to send e-mail to an operator's pager when the job completes or to write an entry in the application event log when the job completes or to delete the job when the job completes. • On the **Targets** page, manage the target servers for the job. There are options to run the job on local server or one or more target servers.
References	https://oracle-base.com/articles/11g/automated-database-maintenance-task-management-11gr1 https://technet.microsoft.com/en-us/library/ms140255(v=sql.105).aspx https://msdn.microsoft.com/en-us/library/ms174202.aspx#Start https://msdn.microsoft.com/en-us/library/ms187658.aspx https://technet.microsoft.com/en-us/library/ms175887(v=sql.105).aspx

	https://docs.microsoft.com/en-us/sql/ssms/agent/create-a-job

Database Email

Feature ID	6
Feature	Database Email
Description	The protocol consists of a set of commands for an email client to dispatch emails to a SMTP server. The UTL_SMTP package provides interfaces to the SMTP commands. For many of the commands, the package provides both a procedural and a functional interface. The functional form returns the reply from the server for processing by the client. The procedural form checks the reply and will raise an exception if the reply indicates a transient (400-range reply code) or permanent error (500-range reply code). Otherwise, it discards the reply. Note that the original SMTP protocol communicates using 7-bit ASCII. Using UTL_SMTP, all text data (in other words, those in VARCHAR2) will be converted to US7ASCII before it is sent over the wire to the server. Some implementations of SMTP servers that support SMTP extension 8BITMIME [RFC1652] support full 8-bit communication between client and server. The body of the DATA command may be transferred in full 8 bits, but the rest of the SMTP command and response should be in 7 bits. When the target SMTP server supports 8BITMIME extension, users of multibyte databases may convert their non-US7ASCII, multibyte VARCHAR2 data to RAW and use the WRITE_RAW_DATA subprogram to send multibyte data using 8-bit MIME encoding. UTL_SMTP provides for SMTP communication as specified in RFC821, but does not provide an API to format the content of the message per RFC 822 (for example, setting the subject of an electronic mail). You must format the message appropriately. In addition, UTL_SMTP does not have the functionality to implement an SMTP server for an email clients to send emails using SMTP.
Category	Admin
To Find Feature Enablement	`SELECT * FROM ALL_SYNONYMS WHERE OWNER = 'PUBLIC' and table_name LIKE 'UTL_MAIL';`
Feature Usage	`SELECT count(*) FROM ALL_SYNONYMS WHERE OWNER = 'PUBLIC' and table_name LIKE 'UTL_MAIL';`
Recommendation	**Feature Description:** Database Mail is an enterprise solution for sending e-mail messages from the SQL Server Database Engine. Using Database Mail, your database applications can send e-mail messages to users. The messages can contain query results, and can also include files from any resource on your network. Database Mail is designed for reliability, scalability, security, and supportability. Database Mail is not active by default. To use Database Mail, you must explicitly enable Database Mail by using either the Database Mail Configuration Wizard, the sp_configure stored procedure, or by using the Surface Area Configuration facet of Policy-Based Management.

	• No Microsoft Outlook or Extended Messaging Application Programming Interface (Extended MAPI) requirement. Database Mail uses the standard Simple Mail Transfer Protocol (SMTP) to send mail. • To minimize the impact on SQL Server, the component that delivers e-mail runs outside of SQL Server, in a separate process. • Database Mail profile allows you to specify more than one SMTP server. Should an SMTP server be unavailable, mail can still be delivered to another SMTP server. • Database Mail is cluster-aware and is fully supported on a cluster. • Database Mail provides background, or asynchronous, delivery. • Database Mail allows you to create multiple profiles within a SQL Server instance. • Each profile can contain multiple failover accounts. **Feature Comparison:** SQL server supports database email same as Oracle. While migrating Database mail wizard should be used to create profiles that can be used from the scripts
Migration Approach	SSMA does not support migrating email configurations. All email profiles need to be created in SQL Server manually and that should be used in Database email feature. As a workaround, we have to enable and configure database mail in SQL server. Prerequisites: • Enable Database Mail. • Create a Database Mail account for the SQL Server Agent service account to use. • Create a Database Mail profile for the SQL Server Agent service account to use and add the user to the **DatabaseMailUserRole** in the **msdb** database. • Set the profile as the default profile for the **msdb** database. Now configure SQL Server Agent to use Database Mail, • In Object Explorer, expand a SQL Server instance. • Right-click SQL Server Agent, and then click Properties. • Click Alert System. • Select Enable Mail Profile. • In the Mail system list, select Database Mail. • In the Mail profile list, select a mail profile for Database Mail. • Restart SQL Server Agent.
References	https://technet.microsoft.com/en-us/library/ms175887(v=sql.105).aspx

Collation

Feature ID	7
Feature	Collation
Description	Oracle bases its language support on the values of parameters that begin with NLS. These parameters specify, for example, how to display currency or how the name of a day is spelled. Oracle Database provides the following types of collation: • Binary • Monolingual • Multilingual • Unicode Collation Algorithm (UCA)
Category	SQL

To Find Feature Enablement	`SELECT *`
	`FROM V$NLS_VALID_VALUES`
	`WHERE parameter = 'SORT'`

Feature Usage	`SELECT * from NLS_SESSION_PARAMETERS WHERE parameter IN ( 'NLS_LANGUAGE', 'NLS_TERRITORY', 'NLS_CHARACTERSET', 'NLS_SORT');`
Recommendation	**Feature Description:** 1.SQL collations are provided for backward compatibility with earlier versions of SQL Server. Windows collations provide consistent string comparisons for both Unicode and for non-Unicode text in SQL Server that are also consistent with string comparisons in the Windows operating system. For all these reasons, Windows collations are preferred unless there are backward compatibility issues or specific performance issues that require a SQL collation. 2.If you are considering a SQL collation based only on the performance characteristics of a SQL collation, realize that the performance of most applications does not benefit significantly from a change in collation. Make sure that you have isolated queries that show a benefit from a SQL collation. As soon as you identify the affected queries, consider the following alternatives to a change in collation. Both alternatives provide a performance benefit that is greater than what you will see if you change the instance collation to a SQL collation: a. If the overhead for the Windows collations is traced to Transact-SQL routines that perform explicit string manipulation or parsing, and if you are using non-Unicode data types, you may want to specify a SQL collation or a binary Windows collation for the operation that is frequently executed and that is most expensive b. If the overhead for the Windows collations is traced to more mundane queries that do not use complex string manipulation functions, improved index or query designs might provide improvements that dwarf those you would see by changing to a SQL collation.

	Feature Comparison: SQL Server supports all types of collations in Oracle. Below link provides information on the types of collations supported in SQL Server. https://technet.microsoft.com/en-us/library/ms144250(v=sql.105).aspx

Migration Approach	SSMA does not support migrating collation. SQL server needs to be configured with proper collation depends on the existing system designs.
	
	Limitations and Restrictions
	• Windows Unicode-only collations can only be used with the COLLATE clause to apply collations to the **nchar**, **nvarchar**, and **ntext** data types on column level and expression-level data. They cannot be used with the COLLATE clause to change the collation of a database or server instance.
	• If the specified collation or the collation used by the referenced object uses a code page that is not supported by Windows, the Database Engine displays an error
	
	Recommendations
	• When you change the database collation, you change the following:
	  ○ Any **char**, **varchar**, **text**, **nchar**, **nvarchar**, or **ntext** columns in system tables are changed to the new collation.
	  ○ All existing **char**, **varchar**, **text**, **nchar**, **nvarchar**, or **ntext** parameters and scalar return values for stored procedures and user-defined functions are changed to the new collation.
	  ○ The **char**, **varchar**, **text**, **nchar**, **nvarchar**, or **ntext** system data types, and all user-defined data types based on these system data types, are changed to the new default collation.
	• You can change the collation of any new objects that are created in a user database by using the COLLATE clause of the ALTER DATABASE statement. This statement does not change the collation of the columns in any existing user-defined tables. These can be changed by using the COLLATE clause of ALTER TABLE.
	
	SQL Server supports setting collations at the following levels:
	
	• Server
	• Database
	• Column
	• Expression
	
	To set or change the database collation
	1. In **Object Explorer**, connect to an instance of the SQL Server Database Engine, expand that instance, and then expand **Databases**.
	2. If you are creating a new database, right-click **Databases** and then click **New Database**. If you do not want the default collation, click the **Options** page, and select a collation from the **Collation** drop-down list.
	Alternatively, if the database already exists, right-click the database that you want and click **Properties**. Click the **Options** page, and select a collation from the **Collation** drop-down list.
	3. After you are finished, click **OK**

References	https://technet.microsoft.com/en-us/library/ms175887(v=sql.105).aspx

Global Temporary Tables

Feature ID	11
Feature	Global temporary tables
Description	Applications often use some form of temporary data store for processes that are too complicated to complete in a single pass. Often, these temporary stores are defined as database tables or PL/SQL tables. From Oracle 8i onward, the maintenance and management of temporary tables can be delegated to the server by using Global Temporary Tables.
Category	Admin
To Find Feature Enablement	select TABLESPACE_NAME, BYTES_USED, BYTES_FREE from V$TEMP_SPACE_HEADER;
Feature Usage	select count(*) from V$TEMP_SPACE_HEADER;
Recommendation	**Feature Description**: SQL Server also supports temporary tables. There are two types of temporary tables: local and global. Local temporary tables are visible only to their creators during the same connection to an instance of SQL Server as when the tables were first created or referenced. Local temporary tables are deleted after the user disconnects from the instance of SQL Server. Global temporary tables are visible to any user and any connection after they are created, and are deleted when all users that are referencing the table disconnect from the instance of SQL Server. • Local temporary tables (CREATE TABLE #t) are visible only to the connection that creates it, and are deleted when the connection is closed. • Global temporary tables (CREATE TABLE ##t) are visible to everyone, and are deleted when all connections that have referenced them have closed. • Tempdb permanent tables (USE tempdb CREATE TABLE t) are visible to everyone, and are deleted when the server is restarted. **Feature Comparison:** Temporary tables in Oracle are permanent objects that hold temporary data that is session local. Temporary tables in SQL Server are temporary objects. In SQL Server, a global temp table holds data that is visible to all sessions. "Global temporary tables are visible to any user and any connection after they are created." Global temp tables are still temporary objects that do not persist indefinitely and may need to be created before use. "Global temporary tables are ... are deleted when all users that are referencing the table disconnect from the instance of SQL Server. Local temporary table, or table variable, is the closest to being the same to Oracle's global temp table, the big difference is you must create it every time.

Migration Approach	SSMA used to migrate all tables into SQL server. Temporary table in Oracle are created as base tables in SQL Server as a part of migration.
	
	We can create a global temporary table in SQL Server as below,
	
	`CREATE TABLE ##tempGlobalB`
	`(`
	`  Column1  INT  NOT NULL ,`
	`  Column2  NVARCHAR(4000)`
	`);`
	
	The following URL explain migration steps: https://msdn.microsoft.com/en-us/library/hh313159(v=sql.110).aspx

References	https://technet.microsoft.com/en-us/library/ms186986.aspx

Stored Procedures

Feature ID	15
Feature	Stored procedures
Description	PL/SQL is a third-generation language that has the expected procedural and namespace constructs, and its tight integration with SQL makes it possible to build complex and powerful applications. Because PL/SQL is executed in the database, you can include SQL statements in your code without having to establish a separate connection. The main types of program units you can create with PL/SQL and store in the database are standalone procedures and functions, and packages. Once stored in the database, these PL/SQL components, collectively known as stored procedures, can be used as building blocks for several different applications. While standalone procedures and functions are invaluable for testing pieces of program logic, Oracle recommends that you place all your code inside a package. Packages are easier to port to another system, and have the additional benefit of qualifying the names of your program units with the package name. For example, if you developed a schema-level procedure called continue in a previous version of Oracle Database, your code would not compile when you port it to a newer Oracle Database installation. This is because Oracle recently introduced the statement CONTINUE that exits the current iteration of a loop and transfers control to the next iteration. If you developed your procedure inside a package, the procedure package_name.continue would have been protected from such name capture.
Category	Admin
To Find Feature Enablement	`SELECT * FROM ALL_OBJECTS WHERE OBJECT_TYPE IN ('PROCEDURE')`
Feature Usage	`SELECT * FROM USER_OBJECTS WHERE OBJECT_TYPE IN ('PROCEDURE')`
Recommendation	**Feature Description:** Like procedures in other programming languages, stored procedures in Microsoft SQL Server can be used to do the following: • Accept input parameters and return multiple values in the form of output parameters to the calling procedure or batch. • Contain programming statements that perform operations in the database, including calling other procedures. • Return a status value to a calling procedure or batch to indicate success or failure, and the reason for failure. **Feature Comparison:** SQL server support stored procedures like in Oracle.
Migration Approach	SSMA can used to migrate Stored procedures to SQL server. However complex PL/SQL scripts cannot be migrated using SSMA. The entire PL/SQL scripts must be to be rewritten for to SQL Server.

	Below are the steps to migrate your Oracle Schema to SQL Server • Start SSMA • **Change Default Project Setting** - By default, SSMA loads only basic Oracle system schemas and packages. You need to customize project settings to allow loading of the schema you want to migrate. Click on Tools from the menu and select Default Project Settings. On the Default Project Setting menu, click on Loading System Objects and check **'<Schema Name You Want to Migrate>'** system object. • **Create a new project**- Once the program is running, click on the **New Project** icon in the upper left corner to get started. Specify the name of the project and the location of the file to save the project information: • **Connect to Oracle -** Click on the **Connect to Oracle** icon from the menu toolbar and provide connection information to your Oracle database. • **Create a schema migration report** - Select the schema, then right-click the schema then select ***Create Report***: • **Connect to SQL Server -** Click on *the Connect to SQL Server* icon from the File Menu. Specify the server name (e.g. *localhost* if SSMA is running on the SQL server machine) and port number (if using other than default 1433 SQL Server port number). Type the name of the database you are migrating to (e.g. **HR**). If the database does not exist, SSMA will create a new database using the default setting. Specify authentication information and click *Connect* to continue. • **Map Schema and Type -** In the **Oracle Metadata Explorer**, check the schema and expand. You can select (or deselect) objects to be migrated as well as map schema. Schema mapping can be done at the Oracle schema level or at the individual object (such as specific table in Oracle) to SQL Server schema. • **Convert the schema -** In the **Oracle Metadata Explorer**, right-click the **schema** and select ***Convert Schema***: • **Review conversion report and resolve error as necessary**. • **Synchronize the SQL Server database**. To deploy the changes to the SQL server, right-click the database in the SQL Server metadata explorer and select **Synchronize with Database**. • **Migrate the data**. From Oracle Metadata Explorer window, right-click on the schema and select ***Migrate Data***. Provide connection information to both the Oracle source database and the target SQL server. **Review Migration Report.** After the data is migrated, a report will be displayed with migration statistics Most metadata settings are read-only. However, you can alter the following metadata: • In Oracle Metadata Explorer, you can alter procedures and type mappings. To convert the altered procedures and type mappings, make changes before you convert schemas. • In SQL Server Metadata Explorer, you can alter the Transact-SQL for stored procedures. To see these changes in SQL Server, make these changes before you load the schemas into SQL Server.
References	https://technet.microsoft.com/en-us/library/ms186986.aspx

Memory Management

Feature ID	16
Feature	Performance
Description	Automatic Memory Management Oracle Database can manage the SGA memory and instance PGA memory completely automatically. You designate only the total memory size to be used by the instance, and Oracle Database dynamically exchanges memory between the SGA and the instance PGA as needed to meet processing demands. This capability is referred to as automatic memory management. With this memory management method, the database also dynamically tunes the sizes of the individual SGA components and the sizes of the individual PGAs. Manual Memory Management If you prefer to exercise more direct control over the sizes of individual memory components, you can disable automatic memory management and configure the database for manual memory management. There are a few different methods available for manual memory management. Some of these methods retain some degree of automation. The methods therefore vary in the amount of effort and knowledge required by the DBA. These methods are: • Automatic shared memory management - for the SGA • Manual shared memory management - for the SGA • Automatic PGA memory management - for the instance PGA • Manual PGA memory management - for the instance PGA
To Find Feature Enablement	SHOW PARAMETER SGA_TARGET SHOW PARAMETER PGA_AGGREGATE_TARGET The above queries to find the current size of SGA and PGA targets SELECT VALUE/1048576 FROM V$PGASTAT WHERE NAME='maximum pga allocated'; To determine the maximum instance PGA allocated in megabytes since the database was started For example, if SGA_TARGET is 272M and PGA_AGGREGATE_TARGET is 90M as shown above, and if the maximum PGA allocated is determined to be 120M, then MEMORY_TARGET should be at least 392M (272M + 120M).
Feature Usage	SELECT VALUE/1048576 FROM V$PGASTAT WHERE NAME='total freeable PGA memory'; SELECT VALUE/1048576 FROM V$PGASTAT WHERE NAME='total PGA used';
Recommendation	**Feature Description:** SQL Server In-Memory OLTP consumes memory in different patterns than disk-based tables. You can monitor the amount of memory allocated and used by memory-optimized tables and indexes in your database using the DMVs or performance counters provided for memory and the garbage collection subsystem. This gives you visibility

	at both the system and database level and lets you prevent problems due to memory exhaustion. **Feature Comparison:** SQL server support memory management by **Using SQL Server Management Studio** SQL Server 2014 ships with built-in standard reports to monitor the memory consumed by in-memory tables. You can access these reports using Object Explorer. You can also use the object explorer to monitor memory consumed by individual memory-optimized tables. **Using DMVs** There are several DMVs available to monitor memory consumed by memory-optimized tables, indexes, system objects, and by run-time structures.
Migration Approach	There is no migration approach for memory management. These needs to be done manually. • As a workaround, we have the option to set the memory options manually in SQL server. This method is useful for system or database administrators to configure an instance of SQL Server in conjunction with the memory requirements of other applications that run on the same computer. • **Using SQL Server Management Studio:** Use the two server memory options, **min server memory** and **max server memory,** to reconfigure the amount of memory (in megabytes) managed by the SQL Server Memory Manager for an instance of SQL Server. By default, SQL Server can change its memory requirements dynamically based on available system resources. To set a fixed amount of memory 1. In Object Explorer, right-click a server and select **Properties.** 2. Click the **Memory** node. 3. Under **Server Memory Options,** enter the amount that you want for **Minimum server memory** and **Maximum server memory.** Use the default settings to allow SQL Server to change its memory requirements dynamically based on available system resources
References	https://msdn.microsoft.com/en-us/library/dn465869.aspx#bkmk_UsingDMVs

CPU Resources usage

Feature ID	17
Feature	CPU Resources usage
Description	Using resource analytics CPU utilization could be analyzed in Oracle. The first chart (Aggregate CPU Across all Databases) shows you the actual CPU utilization (Used CPU) and the available capacity (Available CPU). The CPU Breakdown by Individual Databases chart enables you to compare how your different databases are using their resources. To view CPU utilization breakdown by individual databases, complete the following steps: 1. Click the Group By drop-down list to select what metrics to group the databases by. Available options include the following: • Life Cycle Status • Target Version • Cost Center • Host Name • Contact 2. Optionally, to view the 10 databases consuming the most CPU resources, select the option Top 10 databases. 3. Select the databases for which you want to view specific information by clicking on them.
Category	Performance

To Find Feature Enablement	`SELECT TO_CHAR(SAMPLE_TIME, 'HH24:MI ') AS SAMPLE_TIME,`
	`ROUND(OTHER / 60, 3) AS OTHER,`
	`ROUND(CLUST / 60, 3) AS CLUST,`
	`ROUND(QUEUEING / 60, 3) AS QUEUEING,`
	`ROUND(NETWORK / 60, 3) AS NETWORK,`
	`ROUND(ADMINISTRATIVE / 60, 3) AS ADMINISTRATIVE,`
	`ROUND(CONFIGURATION / 60, 3) AS CONFIGURATION,`
	`ROUND(COMMIT / 60, 3) AS COMMIT,`
	`ROUND(APPLICATION / 60, 3) AS APPLICATION,`
	`ROUND(CONCURRENCY / 60, 3) AS CONCURRENCY,`
	`ROUND(SIO / 60, 3) AS SYSTEM_IO,`
	`ROUND(UIO / 60, 3) AS USER_IO,`
	`ROUND(SCHEDULER / 60, 3) AS SCHEDULER,`
	`ROUND(CPU / 60, 3) AS CPU,`
	`ROUND(BCPU / 60, 3) AS BACKGROUND_CPU`

	FROM (SELECT TRUNC(SAMPLE_TIME, 'MI') AS SAMPLE_TIME, DECODE(SESSION_STATE, 'ON CPU', DECODE(SESSION_TYPE, 'BACKGROUND', 'BCPU', 'ON CPU'), WAIT_CLASS) AS WAIT_CLASS FROM V$ACTIVE_SESSION_HISTORY WHERE SAMPLE_TIME > SYSDATE - INTERVAL '1' HOUR AND SAMPLE_TIME <= TRUNC(SYSDATE, 'MI')) ASH PIVOT(COUNT(*) FOR WAIT_CLASS IN('ON CPU' AS CPU,'BCPU' AS BCPU, 'Scheduler' AS SCHEDULER, 'User I/O' AS UIO, 'System I/O' AS SIO, 'Concurrency' AS CONCURRENCY, 'Application' AS APPLICATION, 'Commit' AS COMMIT, 'Configuration' AS CONFIGURATION, 'Administrative' AS ADMINISTRATIVE, 'Network' AS NETWORK, 'Queueing' AS QUEUEING, 'Cluster' AS CLUST, 'Other' AS OTHER)) ORDER BY 1
Feature Usage	SELECT TO_CHAR(SAMPLE_TIME, 'HH24:MI ') AS SAMPLE_TIME, ROUND(OTHER / 60, 3) AS OTHER, ROUND(CLUST / 60, 3) AS CLUST, ROUND(QUEUEING / 60, 3) AS QUEUEING, ROUND(NETWORK / 60, 3) AS NETWORK, ROUND(ADMINISTRATIVE / 60, 3) AS ADMINISTRATIVE, ROUND(CONFIGURATION / 60, 3) AS CONFIGURATION, ROUND(COMMIT / 60, 3) AS COMMIT, ROUND(APPLICATION / 60, 3) AS APPLICATION, ROUND(CONCURRENCY / 60, 3) AS CONCURRENCY, ROUND(SIO / 60, 3) AS SYSTEM_IO, ROUND(UIO / 60, 3) AS USER_IO, ROUND(SCHEDULER / 60, 3) AS SCHEDULER, ROUND(CPU / 60, 3) AS CPU, ROUND(BCPU / 60, 3) AS BACKGROUND_CPU FROM (SELECT TRUNC(SAMPLE_TIME, 'MI') AS SAMPLE_TIME, DECODE(SESSION_STATE, 'ON CPU', DECODE(SESSION_TYPE, 'BACKGROUND', 'BCPU', 'ON CPU'), WAIT_CLASS) AS WAIT_CLASS FROM V$ACTIVE_SESSION_HISTORY WHERE SAMPLE_TIME > SYSDATE - INTERVAL '1' HOUR AND SAMPLE_TIME <= TRUNC(SYSDATE, 'MI')) ASH PIVOT(COUNT(*) FOR WAIT_CLASS IN('ON CPU' AS CPU,'BCPU' AS BCPU, 'Scheduler' AS SCHEDULER, 'User I/O' AS UIO, 'System I/O' AS SIO, 'Concurrency' AS CONCURRENCY, 'Application' AS APPLICATION,

	'Commit' AS COMMIT, 'Configuration' AS CONFIGURATION, 'Administrative' AS ADMINISTRATIVE, 'Network' AS NETWORK, 'Queueing' AS QUEUEING, 'Cluster' AS CLUST, 'Other' AS OTHER)) ORDER BY 1;
Recommendation	**Feature Description:** • Monitor an instance of Microsoft SQL Server periodically to determine whether CPU usage rates are within normal ranges. • A continually high rate of CPU usage may indicate the need to upgrade the CPU or add multiple processors. Alternatively, a high CPU usage rate may indicate a poorly tuned or designed application. Optimizing the application can lower CPU utilization. • An efficient way to determine CPU usage is to use the **Processor:% Processor Time** counter in System Monitor. • This counter monitors the amount of time the CPU spends executing a thread that is not idle. A consistent state of 80 percent to 90 percent may indicate the need to upgrade your CPU or add more processors. For multiprocessor systems, monitor a separate instance of this counter for each processor. • This value represents the sum of processor time on a specific processor. To determine the average for all processors, use the **System: %Total Processor Time** counter instead. **Feature Comparison:** When you monitor SQL Server and the Microsoft Windows operating system to investigate performance-related issues, concentrate your initial efforts in three main areas: • Disk activity • Processor utilization • Memory usage Monitoring a computer on which System Monitor is running can affect computer performance slightly. Therefore, either log the System Monitor data to another disk (or computer) so that it reduces the effect on the computer being monitored, or run System Monitor from a remote computer. Monitor only the counters in which you are interested. If you monitor too many counters, resource usage overhead is added to the monitoring process and affects the performance of the computer that is being monitored.

Migration Approach	This feature is not a migration feature as we are looking for resources utilized, we can use the below query to get the memory usage as a percent of the "Maximum server memory" option.
	
	`select convert(float, memory_utilization_percentage) / 100 as memory_usage`
	`from sys.dm_os_process_memory`
	
	If you want physical memory use vs. total system memory, look in both `sys.dm_os_process_memory` and sys.dm_os_sys_info:
	
	`select * from sys.dm_os_sys_info`
	
	`declare @physicalMemoryInUseKB bigint`
	`declare @totalSystemMemoryBytes bigint`

	``` select @physicalMemoryInUseKB = physical_memory_in_use_kb from sys.dm_os_process_memory select @totalSystemMemoryBytes = physical_memory_in_bytes from sys.dm_os_sys_info select convert(float, @physicalMemoryInUseKB) * 1024 / convert(float, @totalSystemMemoryBytes) as memory_usage ```
References	http://dba.stackexchange.com/questions/140188/oracle-em-11g-query-to-find-out-cpu-utilization https://docs.oracle.com/cloud/latest/em_home/ITACS/GUID-3BDBFBF2-0A38-4D75-89FF-D9A6CDD8014D.htm#ITACS-GUID-3BDBFBF2-0A38-4D75-89FF-D9A6CDD8014D http://www.sqlshack.com/performance-dashboard-reports-sql-server-2014/

Data concurrency and consistency

Feature ID	19
Feature	Data concurrency and consistency
Description	Data Concurrency arrange that both official data source and duplicated data values are consistent, that means whenever data values official data source is updated then the corresponding replicated data values must also be updated via synchronization to maintain consistency. In a single user database, each transaction is processed serially; therefore, there is no need for contention with interference from other transactions. But in a large data repository conditions, there could be hundreds or thousands of users and data consumers from across many different locations trying to access the warehouse simultaneously. Therefore, a single user database wills not there are three common ways that databases manage data currency and they are as follows: • Pessimistic concurrency control - In this method, a path is available to the users when the record is being retrieved and stays with the user until it is updated within the database. • Optimistic concurrency control - With this method, a path cannot be available to other users while the data is currently being renovated. During renovating, the database analysis the row in the database to determine whether any alteration has been made. An attempt to renew a record that has already been changed can be flagged as concurrency violation. • Multiversion Concurrency Control • Statement-Level Read Consistency • Transaction-Level Read Consistency • Read Consistency with Real Application Crew • Oracle Isolation Levels • Comparison of Serializable Isolation and Read Committed Choice of Isolation Level
Category	Performance

To Find Feature Enablement	`SELECT session_id,lock_type,`
	`mode_held,`
	`mode_requested,`
	`blocking_others,`
	`lock_id1`
	`FROM dba_lock l`
	`WHERE lock_type`
	`NOT IN ('Media Recovery', 'Redo Thread');`

Feature Usage	`SELECT * FROM DBA_BLOCKERS;`
	`SELECT * FROM DBA_DDL_LOCKS;`
	`SELECT * FROM DBA_DML_LOCKS;`

Recommendation	**Feature Description:** Your application can have impeachable indexes and queries, but they won't help you if you can't get to your data because another application has it locked. That's why every DBA and developer must understand SQL Server concurrency and how to troubleshoot excessive blocking or deadlocking. If you've designed your SQL code intelligently and implemented a sensible indexing strategy, there's a good chance your queries will "fly", when tested in isolation. In the real world, however, where multiple processes can access the same data at the same time, SQL Server often must make one process wait, sacrificing **concurrency** and **performance** in order that all processes can succeed without destroying data integrity. **Transactions** are at the heart of concurrency. Server's **optimistic** and **pessimistic** models for mediating concurrent access. **Pessimistic concurrency**, SQL Server's default, uses **locks** to avoid concurrency problems. **Optimistic concurrency** uses row versions to support concurrency.
Migration Approach	There is no direct migration approach available. Before migrating to SQL Server, the Database and Applications must be designed specifically to achieve Data concurrency and consistency.
References	http://mrbool.com/oracle-data-concurrency-and-consistency/28518 https://technet.microsoft.com/en-us/library/cc917674.aspx

Indexes

Feature ID	21
Feature	Indexes
Description	An index is a schema object that contains an entry for each value that appears in the indexed column(s) of the table or cluster and provides direct, fast access to rows. Oracle Database supports several types of index: Normal indexes. (By default, Oracle Database creates B-tree indexes.) Bitmap indexes, which store rowids associated with a key value as a bitmap Partitioned indexes, which consist of partitions containing an entry for each value that appears in the indexed column(s) of the table Function-based indexes, which are based on expressions. They enable you to construct queries that evaluate the value returned by an expression, which in turn may include built-in or user-defined functions. Domain indexes, which are instances of an application-specific index of type indextype
Category	SQL
To Find Feature Enablement	`select dbms_metadata.get_ddl('INDEX', index_name, owner)` from all_indexes;
Feature Usage	`select owner, count(*) from dba_indexes group by OWNER;`
Recommendation	**Feature Description**: The primary reason indexes are built is to provide faster data access to the specific data your query is trying to retrieve. This could be either a clustered or non-clustered index. Without having an index SQL Server will read through all data in order to find the rows that satisfy the query. If you have ever looked at a query plan the difference would be an Index Seek vs a Table Scan as well as some other operations depending on the data selected. **Feature Comparison**:

Following table highlights comparative support:

Feature	Oracle	SQL Server
B-tree unique	Yes	Yes
B-tree non-unique	Yes	Yes
B-tree composite	Yes (32 cols)	Yes (16 cols)
B-tree ascending	Yes	Yes
B-tree cluster	Yes	Yes
B-tree reverse key	Yes	No
B-tree key compressed	Yes	No
B-tree function-based	Yes	No
B-tree index organized table	Yes	Yes (clustered)
B-tree partitioned	Yes	No
Bitmap	Yes	No
Bitmap join	Yes	No
Columnstore index	No	Yes
In-Memory OLTP table index	No	Yes
Invisible index	Yes	No*

Migration Approach	There are various types of indexes are available in the system. Each index can be migrated through SSMA ○ During migration, make sure you understand the index capabilities of SQL Server and use those capabilities. ○ Before migrating indexes, it is important to ensure that they are used. Do not migrate unused indexes as this can result in a lot of unnecessary processing.

	○ Also, note that since SQL is more “set-based” than Oracle, it may need fewer indexes due to advanced query optimizations. ○ We recommend that you use the index tuning wizard after migration to recommend new indexes. Do not run this wizard during peak load times, however. ○ Oracle’s invisible indexes are maintained like any other index, but they are ignored by the optimizer unless the OPTIMIZER_USE_INVISIBLE_INDEXES parameter is set to TRUE at the instance or session level. ○ Disabled indexes in SQL Server prevent user access to the index, and for clustered indexes, to the underlying table data. The index definition remains in metadata and index statistics are kept on nonclustered indexes. Disabling a clustered index on a table prevents access to the data; the data still remains in the table, but is unavailable for DML operations until the index is dropped or rebuilt. ○ Clustered Columnstore indexes in the SQL Server Database Engine, with improved data compression, can be used to significantly speed-up the processing time of common data warehousing workloads that primarily perform bulk loads and read-only queries. If most of your queries are small lookup queries, seeking into a B-tree index may be faster and you may not find a columnstore index to be beneficial. If you test a columnstore index and it does not benefit your workload, you can drop or disable the index.
References	https://www.techonthenet.com/oracle/indexes.php http://viralpatel.net/blogs/invisible-indexes-in-oracle-11g/ https://www.simple-talk.com/sql/performance/14-sql-server-indexing-questions-you-were-too-shy-to-ask/ https://www.brentozar.com/sql/index-all-about-sql-server-indexes/ http://logicalread.solarwinds.com/sql-server-2016-columnstore-pd01/#.WCcA_vorKM8

Queries

Feature ID	22
Feature	Queries
Description	A query is an operation that retrieves data from one or more tables or views. In this reference, a top-level SELECT statement is called a query, and a query nested within another SQL statement is called a subquery. This section describes some types of queries and subqueries and how to use them. The top level of the syntax is shown in this chapter. Refer to SELECT for the full syntax of all the clauses and the semantics of this statement. Structured Query Language (SQL) is the set of statements with which all programs and users access data in an Oracle database. Application programs and Oracle tools often allow users access to the database without using SQL directly, but these applications in turn must use SQL when executing the user's request. This chapter provides background information on SQL as used by most database systems.
Category	SQL
To Find Feature Enablement	

Feature Usage	`SELECT sess.sid,`
	`sess.username,`
	`sqla.optimizer_mode,`
	`sqla.hash_value,`
	`sqla.address,`
	`sqla.cpu_time,`
	`sqla.elapsed_time,`
	`sqla.sql_text`
	`FROM v$sqlarea sqla, v$session sess`
	`WHERE sess.sql_hash_value = sqla.hash_value`
	`AND sess.sql_address = sqla.address ORDER BY sess.username;`

Recommendation	**Feature Description**: A query is a request for data that is stored in SQL Server. A query can be issued by using several forms: • An MS Query or Microsoft Access user can use a graphical user interface (GUI) to pick the data the user wants to see from one or more SQL Server tables. • A user of SQL Server Management Studio or the osql utility can issue a SELECT statement. • A client or middle tier-based application, such as a Microsoft Visual Basic application, can map the data from a SQL Server table into a bound control, such as a grid. Although queries have various ways of interacting with a user, they all accomplish the same task: They present the result set of a SELECT statement to the user. Even if the user never specifies a SELECT statement, as is usually the case with graphical tools such as Visual Studio Query Designer, the client software transforms each user query into a SELECT statement that is sent to SQL Server.

The SELECT statement retrieves data from SQL Server and returns it to the user in one or more result sets. A result set is a tabular arrangement of the data from the SELECT. Like an SQL table, the result set is made up of columns and rows.

Feature Comparison:

Description	Oracle	MS SQL Server
Smallest integer >= n	CEIL	CEILING
Modulus	MOD	%
Truncate number	TRUNC	<none>
Max or min number or string in list	GREATEST, LEAST	<none>
Translate NULL to n	NVL	ISNULL
Return NULL if two values are equal	DECODE	NULLIF
String concatenation	CONCAT(str1,str2)	str1 + str2
Convert ASCII to char	CHR	CHAR
Capitalize first letters of words	INITCAP	<none>
Find string in string	INSTR	CHARINDEX
Find pattern in string	INSTR	PATINDEX
String length	LENGTH	DATALENGTH
Pad string with blanks	LPAD, RPAD	<none>
Trim leading or trailing chars other than blanks	LTRIM(str,chars), RTRIM(str,chars)	<none>
Replace chars in string	REPLACE	STUFF
Convert number to string	TO_CHAR	STR, CAST
Convert string to number	TO_NUMBER	CAST
Get substring from string	SUBSTR	SUBSTRING
Char for char translation in string	TRANSLATE	<none>
Date addition	ADD_MONTH or +	DATEADD
Date subtraction	MONTHS_BETWEEN or -	DATEDIFF
Last day of month	LAST_DAY	<none>
Time zone conversion	NEW_TIME	<none>
Next specified weekday after date	NEXT_DAY	<none>
Convert date to string	TO_CHAR	DATENAME, CONVERT
Convert string to date	TO_DATE	CAST
Convert date to number	TO_NUMBER(TO_CHAR(d))	DATEPART
Date round	ROUND	CONVERT
Date truncate	TRUNC	CONVERT

	Current date	SYSDATE	GETDATE
	Convert hex to binary	HEXTORAW	CAST
	Convert binary to hex	RAWTOHEX	CONVERT
	If statement in an expression	DECODE	CASE ... WHEN or COALESCE
	User's login id number or name	UID, USER	SUSER_ID, SUSER_NAME
	User's database id number or name	UID, USER	USER_ID, USR_NAME
	Current user	USER	USER
Migration Approach	Migrating Oracle hierarchical queries. Oracle provides the following syntax elements to build hierarchical queries: 1. The START WITH condition. Specifies the hierarchy's root rows. 2. The CONNECT BY condition. Specifies the relationship between the hierarchy's parent rows and child rows. 3. The PRIOR operator. Refers to the parent row. 4. The CONNECT_BY_ROOT operator. Retrieves the column value from the root row. 5. The NO_CYCLE parameter. Instructs the Oracle Database to return rows from a query, even if a cycle exists in the data. 6. The LEVEL, CONNECT_BY_ISCYCLE, and CONNECT_BY_ISLEAF pseudocolumns. 7. The SYS_CONNECT_BY_PATH function. Retrieves the path from the root to node. 8. The ORDER SIBLINGS BY clause. Applies ordering to the siblings of the hierarchy. Oracle processes hierarchical queries in this order: 1. Evaluates a join first, if one is present, whether the join is specified in the FROM clause or with WHERE clause predicates. 2. Evaluates the CONNECT BY condition. 3. Evaluates any remaining WHERE clause predicates. Oracle then uses the information from these evaluations to form the hierarchy as follows: 4. Oracle selects the hierarchy's root row(s) (those rows that satisfy the START WITH condition). 5. Oracle selects each root row's child rows. Each child row must satisfy the CONNECT BY condition with respect to one of the root rows. 6. Oracle selects successive generations of child rows. Oracle first selects the children of the rows returned in Step 2, and then the children of those children, and so on. Oracle always selects children by evaluating the CONNECT BY condition with respect to a current parent row. 7. If the query contains a WHERE clause without a join, Oracle eliminates all rows from the hierarchy that do not satisfy the WHERE clause's conditions. Oracle evaluates that condition for each row individually, rather than removing all the children of a row that does not satisfy the condition. 8. Oracle returns the rows in the order shown in Figure 3. In the figure, children appear below their parents.		
References	https://technet.microsoft.com/en-us/library/ms190659(v=sql.105).aspx		

Regular Expressions

<table>
<tr><td>Feature ID</td><td>27</td></tr>
<tr><td>Feature</td><td>Regular Expression</td></tr>
<tr><td>Description</td><td>Regular expressions enable you to search for patterns in string data by using standardized syntax conventions. You specify a regular expression through the following types of characters:

• Metacharacters, which are operators that specify search algorithms
• Literals, which are the characters for which you are searching

A regular expression can specify complex patterns of character sequences. For example, the following regular expression searches for the literals f or ht, the t literal, the p literal optionally followed by the s literal, and finally the colon (:) literal:</td></tr>
<tr><td>Category</td><td>SQL</td></tr>
<tr><td>To Find Feature Enablement</td><td>NA</td></tr>
<tr><td>Feature Usage</td><td>NA</td></tr>
<tr><td>Recommendation</td><td>Feature Description: A regular expression describes one or more strings to match when you search a body of text. The expression serves as a template for matching a character pattern to the string that is being searched.

A regular expression consists of ordinary characters (for example, letters a through z) and special characters, known as metacharacters.

Feature Comparison: Oracle has various inbuilt functions. While migrating to SQL Server all these inbuilt functions needs to be translated into POSIX regular expression which is supported by SQL server. Oracle support POSIX Regular expressions, which also supported by SQL Server</td></tr>
<tr><td>Migration Approach</td><td>There is no migration approach available directly in SSMA.

We can write regular expressions by using the below table syntax<table><tr><td>Syntax</td></tr><tr><td>. = Indicates any character</td></tr></table></td></tr>
</table>

	**** = *Indicates that the character that follows is interpreted as is, instead of as a special character*
	() = *Operators that are inside the parentheses are grouped*
	{n} = *Generates **n** instances of previous item*
	{n,m} = *Generates at least **n** instances but no more than **m** instances of the previous item*
	{n,} = *Generates **n** or more instances of the previous item*
	***** = *Generates zero or more instances of the previous item*
	+ = *Generates one or more instances of the previous item*
	? = *Generates zero or one instance of the previous item*
	| = *Generates an item on either side of the **|** character*
	[] = *Generates any character inside the brackets*
	[a-z] = *Generates any character in the specified range of characters*
	[^abc] = *Generates any character except those inside the brackets*
References	http://www.regular-expressions.info/oracle.html https://docs.oracle.com/cd/B28359_01/appdev.111/b28424/adfns_regexp.htm#CHDIDJJC

DML

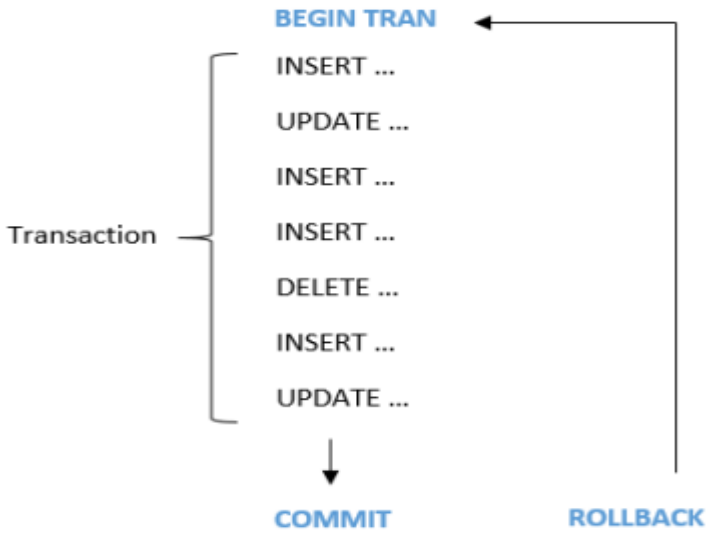
Feature ID	29
Feature	DML
Description	Data manipulation language (DML) statements access and manipulate data in existing tables. In the SQL*Plus environment, you can enter a DML statement after the SQL> prompt. In the SQL Developer environment, you can enter a DML statement in the Worksheet. Alternatively, you can use the SQL Developer Connections frame and tools to access and manipulate data. To see the effect of a DML statement in SQL Developer, you might have to select the schema object type of the changed object in the Connections frame and then click the Refresh icon. The effect of a DML statement is not permanent until you commit the transaction that includes it. A transaction is a sequence of SQL statements that Oracle Database treats as a unit (it can be a single DML statement). Until a transaction is committed, it can be rolled back (undone).
Category	SQL
To Find feature enablement	
Feature Usage	

Recommendation	**Feature Comparison**: SQL Server supports DML which is basic of SQL specification. In Oracle DDL statements are not transactional. Which means in Oracle the database implicitly commits the current transaction before and after every DDL statement.
	
	Data Manipulation Language (DML) is a vocabulary used to retrieve and work with data in SQL Server 2016. Use these statements to add, modify, query, or remove data from a SQL Server database. DML commands are not auto-committed. It means changes made by DML command are not permanent to database, it can be rolled back.
	
	SQL Server DML – MERGE Statement
	The SQL Server MERGE statement is used to synchronise the data of two tables, based on differences found between them, if the same row exists in both tables (row with the same customer id for example), but still each row has different values (each table holds a different phone number of that customer), UPDATE operation will be executed. If the row only exists in one table, INSERT operation will be executed.
	`MERGE INTO destination_table alias`
	`USING source_table alias`
	`ON condition`
	`WHEN MATCHED THEN`
	`UPDATE SET`

```
destination_table_alias.column = source_table_alias.column,
destination_table_alias.column = source_table_alias.column
…
WHEN NOT MATCHED THEN
INSERT VALUES  (source_table_alias.column, source_table_alias.column ..
               )
```

Database Transactions

Transactions are a single unit of various modification commands (such as UPDATE, INSERT, DELETE), which in most cases are associated with a single logical group. The term “a single logical group” refers to a set of operations with logical connection; for example: a batch of DML operations that are meant for updating specific data items in the customers table.



The COMMIT command used to save all changes made by the transaction in the database. The COMMIT command saves all modifications since the last COMMIT or ROLLBACK command.

The ROLLBACK command used to undo changes made by a transaction. The ROLLBACK command can only undo modifications since the last COMMIT or ROLLBACK command that was issued.

SQL Server Autocommit Mode – In SQL Server, by default, every modification (such as UPDATE, DELETE, INSERT) is committed automatically once it completes. In SQL Server, you can start an Explicit Transaction (one that you decide when and how to close) using the BEGIN TRAN command.

Migration Approach	DML statements can be migrated through SSMA. There is various level of DML statements available in function, T-SQL etc., Each DML statements are handled differently by DML. Oracle and SQL Server use different dialects of the SQL language, but SSMA solves most of the incompatibilities introduced by this difference. For example, Oracle uses CONNECT BY statements for hierarchical queries, while SQL Server implements hierarchical queries by using common table expressions SSMA handles another nonstandard Oracle feature: the special outer join syntax with the (+) qualifier. SSMA converts these queries by transforming them into ANSI format. Oracle pseudocolumns, such as ROWID or ROWNUM, present a special problem. When converting ROWNUM, SSMA emulates it with the TOP keyword of the SELECT statement if this pseudocolumn is

	used only to limit the size of the result set. If the row numbers appear in a SELECT list, SSMA uses the ROW_NUMBER( ) function. The ROWID problem can be solved by an optional column named ROWID, which stores a unique identifier in SQL Server. SSMA does not convert dynamic SQL statements because the actual statement is not known until execution time and, in most cases, it cannot be reconstructed at conversion time. There is a workaround: The Oracle metabase tree displayed in SSMA contains a special node named Statements in which you can create and convert ad hoc SQL statements. If you can manually reproduce the final form of a dynamic SQL command, you can convert it as an object in the Statements node.
References	https://technet.microsoft.com/en-us/library/ms186986.aspx http://ramkedem.com/en/sql-server-dml/

Datatypes

<table>
<tr><td>Feature ID</td><td>30</td></tr>
<tr><td>Feature</td><td>Datatypes</td></tr>
<tr><td>Description</td><td>Each column value and constant in a SQL statement has a datatype, which is associated with a specific storage format, constraints, and a valid range of values. When you create a table, you must specify a datatype for each of its columns.

Oracle provides the following categories of built-in datatypes:

• Overview of Character Datatypes
• Overview of Numeric Datatypes
• Overview of DATE Datatype
• Overview of LOB Datatypes
• Overview of RAW and LONG RAW Datatypes
• Overview of ROWID and UROWID Datatypes</td></tr>
<tr><td>Category</td><td>SQL</td></tr>
<tr><td>To Find Feature Enablement</td><td><pre>select distinct data_type,data_length,data_precision,data_scale
from all_tab_columns</pre></td></tr>
<tr><td>Feature Usage</td><td><pre>select distinct data_type,data_length,data_precision,data_type_owner
from all_tab_columns</pre></td></tr>
<tr><td>Recommendation</td><td>
<table>
<tr><th>Microsoft SQL Server</th><th>Description</th><th>Oracle</th></tr>
<tr><td>INTEGER</td><td>Four-byte integer, 31 bits, and a sign. May be abbreviated as "INT" (this abbreviation was required prior to version 5).</td><td>NUMBER(10)</td></tr>
<tr><td>SMALLINT</td><td>Two-byte integer, 15 bits, and a sign.</td><td>NUMBER(6)</td></tr>
<tr><td>TINYINT</td><td>One byte integer, 8 bits and no sign. Holds whole numbers between 0 and 255.</td><td>NUMBER(3)</td></tr>
<tr><td>REAL</td><td>Floating point number. Storage is four bytes and has a binary precision of 24 bits, a 7-digit precision.

Data can range from –3.40E+38 to 3.40E+38.</td><td>FLOAT</td></tr>
<tr><td>FLOAT</td><td>A floating point number. This column has 15-digit precision.</td><td>FLOAT</td></tr>
</table>
</td></tr>
</table>

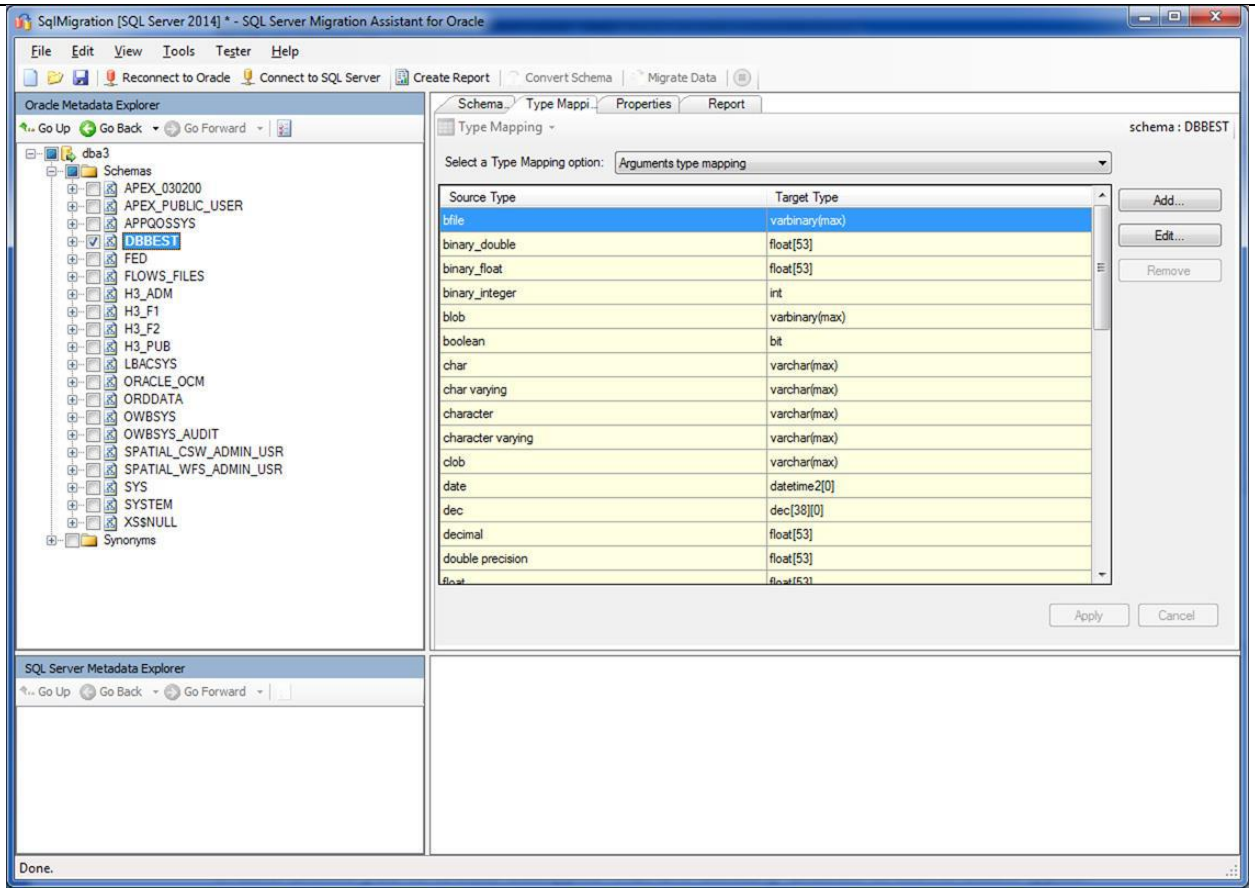
BIT	A Boolean 0 or 1 stored as one bit of a byte. Up to 8-bit columns from a table may be stored in a single byte, even if not contiguous. Bit data cannot be NULL, except for Microsoft SQL Server 7.0, where null is allowed by the BIT data type.	NUMBER(1)
CHAR(n)	Fixed-length string of exactly n 8-bit characters, blank padded. Synonym for CHARACTER. 0 < n < 256 for Microsoft SQL Server. 0 < n < 8000 for Microsoft SQL Server 7.0.	CHAR(n)
VARCHAR(n)	Varying-length character string. 0 < n < 256 for Microsoft SQL Server. 0 < n < 8000 for Microsoft SQL Server 7.0.	VARCHAR2(n)
TEXT	Character string of 8-bit bytes allocated in increments of 2k pages. "TEXT" is stored as a linked-list of 2024-byte pages, blank padded. TEXT columns can hold up to (231-1) characters.	CLOB
IMAGE	Binary string of 8-bit bytes. Holds up to (231-1) bytes of binary data.	BLOB
BINARY(n)	Fixed length binary string of exactly n 8-bit bytes. 0 < n < 256 for Microsoft SQL Server. 0 < n < 8000 for Microsoft SQL Server 7.0.	RAW(n)/BLOB
VARBINARY(n)	Varying length binary string of up to n 8-bit bytes. 0 < n < 256 for Microsoft SQL Server. 0 < n < 8000 for Microsoft SQL Server 7.0.	RAW(n)/BLOB
DATETIME	Date and time are stored as two 4-byte integers. The date portion is represented as a count of the number of days offset from a baseline date (1/1/1900) and is stored in the first integer. Permitted values are legal dates between 1st January, 1753 AD and 31st December, 9999 AD. Permitted values in the time portion are legal times in the range 0 to 25920000. Accuracy is to the nearest 3.33 milliseconds with rounding	DATE

	downward. Columns of type DATETIME have a default value of 1/1/1900.	
SMALL-DATETIME	Date and time stored as two 2-byte integers. Date ranges from 1/1/1900 to 6/6/2079. Time is the count of the number of minutes since midnight.	DATE
MONEY	A monetary value represented as an integer portion and a decimal fraction, and stored as two 4-byte integers. Accuracy is to the nearest 1/10,000. When inputting Data of this type it should be preceded by a dollar sign ($). In the absence of the "$" sign, Microsoft SQL Server creates the value as a float. Monetary data values can range from -922,337,203,685,477.5808 to 922,337,203,685,477.5807, with accuracy to a ten-thousandth of a monetary unit. Storage size is 8 bytes.	NUMBER(19,4)
NCHAR(n)	Fixed-length character data type which uses the UNICODE UCS-2 character set. n must be a value in the range 1 to 4000. SQL Server storage size is two times n. Note: Microsoft SQL Server storage size is two times n. The Oracle Migration Workbench maps columns sizes using byte semantics, and the size of Microsoft SQL Server NCHAR data types appear in the Oracle Migration Workbench Source Model with "Size" specifying the number of bytes, as opposed to the number of Unicode characters. Thus, a SQL Server column NCHAR(1000) will appear in the Source Model as NCHAR(2000).	CHAR(n*2)
NVARCHAR(n)	Fixed-length character data type which uses the UNICODE UCS-2 character set. n must be a value in the range 1 to 4000. SQL Server storage size is two times n. Note: Microsoft SQL Server storage size is two times n. The Oracle Migration Workbench maps columns sizes using byte semantics, and the size of Microsoft SQL	VARCHAR(n*2)

		Server NVARCHAR data types appear in the Oracle Migration Workbench Source Model with "Size" specifying the number of bytes, as opposed to the number of Unicode characters. Thus, a SQL Server column NVARCHAR(1000) will appear in the Source Model as NVARCHAR(2000).	
	SMALLMONEY	Same as MONEY except monetary data values from -214,748.3648 to +214,748.3647, with accuracy to one ten-thousandth of a monetary unit. Storage size is 4 bytes.	NUMBER(10,4)
	TIMESTAMP	TIMESTAMP is defined as VARBINARY(8) with NULL allowed. Every time a row containing a TIMESTAMP column is updated or inserted, the TIMESTAMP column is automatically increment by the system. A TIMESTAMP column may not be updated by users.	NUMBER
	SYSNAME	VARCHAR(30) in Microsoft SQL Server. NVARCHAR(128) in Microsoft SQL Server 7.0.	VARCHAR2(30) and VARCHAR2(128) respectively.

Feature Comparison: SQL Server supports all datatypes of Oracle, while migrating there should be a clear mapping with appropriate datatypes. SSMA for Oracle take care of datatypes matching automatically any incompatible data can be fixed manually.

Migration Approach	SSMA helps to migrate schemas from Oracle to SQL server. Mapping Oracle schema into SQL Server schema is very important migration step. The following URL explains steps for schema mapping. https://msdn.microsoft.com/en-us/library/hh313153(v=sql.110).aspx SSMA supports all built-in Oracle types. SSMA type mapping is applied to table columns, subprogram arguments, a function's returned value, and to local variables. Usually the mapping rules are the same for all these categories, but in some cases, there are differences. In SSMA, you can adjust mapping rules for some predefined limits. You can establish custom mappings for the whole schema, for specific group of objects, or to a single object on the Oracle view pane's **Type Mapping** tab.



Oracle allows you to create subtypes that are aliases of some basic types.

SSMA does not process subtypes, but you can emulate that functionality manually if you can convert the basic type.

Generally, it is enough to replace the Oracle declaration:
SUBTYPE <type-name> IS <basic-type> [NOT NULL]

With the SQL Server 2014 declaration:
CREATE TYPE <type-name> FROM <basic-type-converted> [NOT NULL]

You may need to change the target <type-name> if the subtype is defined in the Oracle package. To establish the scope of this name, add a package prefix such as PackageName$<type-name>.

References	https://docs.oracle.com/cd/B28359_01/server.111/b28318/datatype.htm#CNCPT113

DDL

Feature ID	31
Feature	Data definition language
Description	Data definition language (DDL) statements let you to perform these tasks: • Create, alter, and drop schema objects • Grant and revoke privileges and roles • Analyze information on a table, index, or cluster • Establish auditing options • Add comments to the data dictionary The CREATE, ALTER, and DROP commands require exclusive access to the specified object. For example, an ALTER TABLE statement fails if another user has an open transaction on the specified table. The GRANT, REVOKE, ANALYZE, AUDIT, and COMMENT commands do not require exclusive access to the specified object. For example, you can analyze a table while other users are updating the table. Oracle Database implicitly commits the current transaction before and after every DDL statement. Many DDL statements may cause Oracle Database to recompile or reauthorize schema objects. For information on how Oracle Database recompiles and reauthorizes schema objects and the circumstances under which a DDL statement would cause this, see Oracle Database Concepts. DDL statements are supported by PL/SQL with the use of the DBMS_SQL package.
Category	SQL
Feature Usage	
Recommendation	SQL server DDLs are similar to Oracle DDLs. In addition, there is a tool is provided my Microsoft to convert the Schema and views from oracle to SQL Server. SQL Server Migration Asistant: https://msdn.microsoft.com/en-us/library/hh313179.aspx
Migration Approach	Various level of DDL statements can be migrated through SSMA. SQL Server Migration Asistant: https://msdn.microsoft.com/en-us/library/hh313179.aspx

Joins and Operations

<table>
<tr><td>Feature ID</td><td>32</td></tr>
<tr><td>Feature</td><td>Joins and operations</td></tr>
<tr><td>Description</td><td>A join combines the output from exactly two row sources, such as tables or views, and returns one row source. The returned row source is the data set.

A join is characterized by multiple tables in the WHERE (non-ANSI) or FROM ... JOIN (ANSI) clause of a SQL statement. Whenever multiple tables exist in the FROM clause, Oracle Database performs a join.

A join condition compares two row sources using an expression. The join condition defines the relationship between the tables. If the statement does not specify a join condition, then the database performs a Cartesian join, matching every row in one table with every row in the other table.</td></tr>
<tr><td>Category</td><td>SQL</td></tr>
<tr><td>To find feature enablement</td><td></td></tr>
<tr><td>Feature Usage</td><td></td></tr>
<tr><td>Recommendation</td><td>Join Operation widely differs from Oracle to SQL Server if was return in Old style. But with the new types of Joins which is recognizable in SQL servers as well.

<table>
<tr><th>Oracle</th><th>Microsoft SQL Server</th></tr>
<tr><td>Join Conditions</td><td>Available</td></tr>
<tr><td>Equijoins</td><td>Available</td></tr>
<tr><td>Self Joins</td><td>Available</td></tr>
<tr><td>Inner Joins</td><td>Available</td></tr>
<tr><td>Outer Joins</td><td>Available</td></tr>
<tr><td>Antijoins</td><td>Available</td></tr>
<tr><td>Semijoins</td><td>Available</td></tr>
</table>
</td></tr>
<tr><td>Migration Approach</td><td>SSMA could handle migration of Queries which handles the join operations while migrating those queries. Migrating Hierarchical queries migration are handled by SSMA tool as steps

1. Evaluates a join first, if one is present, whether the join is specified in the FROM clause or with WHERE clause predicates.
2. 2. Evaluates the CONNECT BY condition.
3. 3. Evaluates any remaining WHERE clause predicates.</td></tr>
<tr><td>References</td><td>https://docs.oracle.com/cd/B28359_01/server.111/b28286/queries006.htm#SQLRF52338</td></tr>
</table>

NoSQL Feature

Feature ID	33
Feature	NoSQL Feature
Description	Oracle does not support native NoSQL feature. Oracle provides a separate server instance to support NoSQL feature
Category	SQL
To find feature enablement	
Feature Usage	
Recommendation	Feature Description: SQL Server's memory-optimized tables offers significant performance advantages over traditional tables including lock-free writes, fully compiled stored procedures, and the option to avoid disk I/O altogether. But it also came with a lot of limitations, including the inability to work with the large documents favored by NoSQL style designs. With SQL Server 2016, many of those limitations have been removed. First up is support for LOB types in both memory-optimized tables and compiled stored procedures. This means that you can use varChar(max), nVarChar(max) (both of which cover XML and JSON data) and varBinary(max). The 8060 byte row size limit has also been lifted, even for wide tables that don't contain LOB types. That said, Microsoft is recommending against using this feature if possible. If you can fit everything into a varChar(8000) or smaller column instead of varChar(max), you won't have to pay for hitting the hidden table that stores large objects during writes. Constraints for Memory-Optimized Tables Another limitation of memory-optimized tables was the inability to create constraints (aside from unique primary keys). While not strictly necessary from an application design standpoint, constraints do eliminate the possibility for several types of data corruption. • FOREIGN KEY constraints between memory-optimized tables • CHECK constraints • UNIQUE constraints Note that foreign key constraints between a normal table and a memory-optimized table are still not permitted.
Migration Approach	There is no migration approach as this feature is not directly available in Oracle. **To transform relational schema into NoSQL in SQL server:**

SQL Server 2016/Azure SQL Database introduce hybrid approach where you can choose between relational and NoSQL concepts. As an example, if you have products and their reviews you don't need to create separate tables if you don't want them. You can create additional columns in the primary table that will contain collection of related entities formatted as JSON arrays:

```
ALTER TABLE Production.Product
 ADD Reviews NVARCHAR(MAX)
 CONSTRAINT [Reviews are formatted as JSON] CHECK(ISJSON(Reviews)>0)
```

In this example, we are adding a simple text column with a constraint that specifies that reviews should be formatted as JSON (similar to NoSQL databases). There is no new syntax for JSON constraint - you can use standard check constraint with function ISJSON that checks is Reviews text formatted as a JSON object.

If we want to move related product reviews from a separate table into this column we can use a simple UPDATE query:

```
UPDATE Production.Product
 SET Reviews = (
 SELECT ReviewerName AS [Reviewer.Name],
 EmailAddress AS [Reviewer.Email],
 ReviewDate, Rating, ModifiedDate
 FROM Production.ProductReview
 WHERE Production.ProductReview.ProductID = Production.Product.ProductID
 FOR JSON PATH)
```

Also, Azure platform provides four fully-managed NoSQL services:

- Azure DocumentDB
- Azure Table Storage
- Azure HBase as a part of HDInsight
- Azure Redis Cache

The following comparison chart maps out the key differentiators for each service:

	Azure DocumentDB	Azure Table Storage	Azure HBase as part of HDInsight	Azure Redis Cache
Technology	Document store	Key/value store	Column family store	Redis
Use case	For massive scale applications that need rich queries over a flexible data model, predictable performance, limitless throughput, and/or global distribution to provide low-latency access to any number of regions over a single dataset.	For Internet scale applications requiring large data sets, cost effective storage, and fast time to solution.	For sparsely populated tables that are too big for a relational database. Columns are grouped together into column families.	For a dedicated in memory store and message broker, using the popular open source Redis project.
Data storage	JSON documents in SSD backed collections	Entities in the form of key-value pairs	Tables, rows, columns, cells, and column families	An in-memory store, with different data structures such as strings, sorted sets, and more. Optional periodic backup to Azure storage
Query language	Enhanced subset of SQL syntax	OData subset	HiveQL and SQL-like syntax	Commands for searching keys and items in a set
Transactional boundary	All documents in the same partition	All entities in the same partition	All cells in same row	All Redis operations are atomic in the context of a shard
Stored procedure support	Yes in JavaScript	No	Yes in Java	Full power of the Lua scripting language
Indexing	All properties indexed by default, custom indexes can include or exclude certain paths. Full-text search available through Azure Search.	Indexed by Partition Key and Row Key by default. Secondary indexes can be achieved programmatically. Full-text search available through Azure Search.	RowKey, lexicographically sorted on the primary row key. No secondary indexes.	Primary index based on keys. Secondary indexes possible if implemented by users.
Consistency options	Data replicates to any number of regions of your choice with well-defined tunable consistency semantics (at the click of a button): Strong, bounded staleness, session, and eventual	Strong consistency	Strict consistency	Strong consistency
Performance	<10ms single document reads for 99% of requests <15ms single document writes for 99% of requests 10,000 operations per second per partition	2000 entities/second per partition	Milliseconds for writes and reads, highly tunable	Low latency and very high throughput Up to 250k operations per second per shard

References

https://blogs.msdn.microsoft.com/sqlserverstorageengine/2015/09/01/combining-relational-and-nosql-concepts-in-sql-server/

Synonyms

Feature ID	52
Feature	Synonyms
Description	Use the CREATE SYNONYM statement to create a synonym, which is an alternative name for a table, view, sequence, procedure, stored function, package, materialized view, Java class schema object, user-defined object type, or another synonym. Synonyms provide both data independence and location transparency. Synonyms permit applications to function without modification regardless of which user owns the table or view and regardless of which database holds the table or view. However, synonyms are not a substitute for privileges on database objects. Appropriate privileges must be granted to a user before the user can use the synonym. You can refer to synonyms in the following DML statements: SELECT, INSERT, UPDATE, DELETE, FLASHBACK TABLE, EXPLAIN PLAN, and LOCK TABLE. You can refer to synonyms in the following DDL statements: AUDIT, NOAUDIT, GRANT, REVOKE, and COMMENT.
Category	SQL

To find feature enablement	`Select * from lewis.testtab`
	`Create synonym testtab for lewis.testtab;`
	
	`Select * from testtab`

Feature Usage	`select count(*) from DBA_synonyms;`
Recommendation	Feature Description: A synonym is a database object that serves the following purposes: • Provides an alternative name for another database object, referred to as the base object, that can exist on a local or remote server. • Provides a layer of abstraction that protects a client application from changes made to the name or location of the base object. A synonym belongs to a schema, and like other objects in a schema, the name of a synonym must be unique. You can create synonyms for the following database objects: • Assembly (CLR) stored procedure • Assembly (CLR) table-valued function • Assembly (CLR) scalar function • Assembly (CLR) aggregate functions • Replication-filter-procedure • Extended stored procedure • SQL scalar function • SQL table-valued function • SQL inline-tabled-valued function

	• SQL stored procedure • View • Table (user defined: local and global temporary tables) Other than this, there are some points which needed to be remembered for synonyms. • Only synonym owners, members of **db_owner**, or members of **db_ddladmin** can grant permission on a synonym. • You can use synonyms in place of their referenced base object in several SQL statements and expression contexts. When you are working with synonyms in the contexts previously stated, the base object is affected. For example, if a synonym references a base object that is a table and you insert a row into the synonym, you are actually inserting a row into the referenced table. • Synonyms are not schema-bound.

Migration Approach	SSMA could handle Synonyms while migration.
	• Oracle private synonyms are converted to SQL Server synonyms stored in the target database.
	• SSMA converts public synonyms to synonyms defined in the ssma_oracle schema.
	
	Now to create synonyms in SQL Server you can use either use SQL Server management studio or can use transact sql.
	
	Using SQL server management studio:
	1. In **Object Explorer**, expand the database where you want to create your new view.
	2. Right-click the **Synonyms** folder, then click **New Synonym…**.
	3. In the **Add Synonym** dialog box, enter the following information.
	Synonym name
	Type the new name you will use for this object.
	Synonym schema
	Type the schema of the new name you will use for this object.
	Server name
	Type the server instance to connect to.
	Database name
	Type or select the database containing the object.
	Schema
	Type or select the schema that owns the object.
	Object type
	Select the type of object.
	Object name
	Type the name of the object to which the synonym refers.
	
	Using Transact-SQL:
	1. Connect to the Database Engine.
	2. From the Standard bar, click **New Query**.
	3. Copy and paste the following examples into the query window and click **Execute**.
	`USE tempdb;`
	`GO`
	`CREATE SYNONYM MyAddressType`
	`FOR AdventureWorks2012.Person.AddressType;`

	GO The example creates a synonym for an existing table in the **AdventureWorks2012** database. The synonym is then used in subsequent examples. Now to get information about synonyms, you can refer sys.synonyms catalog view. it contains an entry for each synonym in a given database. This catalog view exposes synonym metadata such as the name of the synonym and the name of the base object
References	https://blogs.msdn.microsoft.com/sqlserverstorageengine/2015/09/01/combining-relational-and-nosql-concepts-in-sql-server/ https://msdn.microsoft.com/en-IN/library/ms187552.aspx https://msdn.microsoft.com/en-in/library/ms177544.aspx

Operators

Feature ID	55
Feature	Operators
Description	An operator manipulates individual data items and returns a result. The data items are called operands or arguments. Operators are represented by special characters or by keywords. For example, the multiplication operator is represented by an asterisk (*) and the operator that tests for nulls is represented by the keywords IS NULL. There are two general classes of operators: unary and binary. Oracle Database Lite SQL also supports set operators.
Category	SQL
Find Feature Enabled	
Feature Usage	
Recommendation	

Operator	Same in Both Databases	Microsoft SQL Server Only	Oracle Only
Equal to	=		
Not equal to	!=		^=
	<>		
Less than	<		
Greater than	>		
Less than or equal to	<=	!>	
Greater than or equal to	>=	!<	
Greater than or equal to xand less than or equal to y	BETWEEN x AND y		
Less than x or greater than y	NOT BETWEEN x AND y		

Pattern Matches	LIKE 'a%'	LIKE'a[x-z]'	LIKE 'a\%'
afollowed by 0 or more characters	LIKE 'a_'	LIKE'a[^x-z]'	ESCAPE '\'
afollowed by exactly 1 character			
afollowed by any character between x and z			
afollowed by any character except those between x and z			
afollowed by %			
Does not match pattern	NOT LIKE		
No value exists	IS NULL		
A value exists	IS NOT NULL		
At least one row returned by query	EXISTS (query)		
No rows returned by query	NOT EXISTS (query)		
Equal to a member of set	IN =ANY		= SOME
Not equal to a member of set	NOT IN != ANY <> ANY		!= SOME <> SOME
Less than a member of set	< ANY		< SOME
Greater than a member of set	> ANY		> SOME
Less than or equal to a member of set	<= ANY	!> ANY	<= SOME

Greater than or equal to a member of set	>= ANY	!< ANY	>= SOME
Equal to every member of set	=ALL		
Not equal to every member of set	!= ALL <> ALL		
Less than every member of set	< ALL		
Greater than every member of set	> ALL		
Less than or equal to every member of set	<= ALL	!> ALL	
Greater than or equal to every member of set	>= ALL	!< ALL	

NULL Construct	Microsoft SQL Server	Oracle
where col1 = NULL	depends on the data	FALSE
where col1 != NULL	depends on the data	TRUE
where col1 IS NULL	depends on	depends on

	the data the data
where col1 IS NOT NULL	depends on the data depends on the data
	TRUE FALSE

Operator	Same in Both Databases	Microsoft SQL Server Only	Oracle Only
Add	+		
Subtract	-		
Multiply	*		
Divide	/		
Modulo	v	%	mod(x, y)

where NULL = NULL

Arthimatic Operators

String Operators

Operator	Same in Both Databases	Microsoft SQL Server Only	Oracle Only
Concatenate	s	+	||
Identify Literal	'this is a string'	"this is also a string"	

Set Operators

Operator	Same in Both Databases	Microsoft SQL Server Only	Oracle Only
Distinct row from either query	UNION		
All rows from both queries	UNION ALL		
All distinct rows in both queries	d		INTERSECT
All distinct rows in the first query but not in the second query	d		MINUS

Bit Operators

Operator	Same in Both Databases	Microsoft SQL Server Only	Oracle Only
bit and		&	
bit or		|	
bit exclusive or		^	
bit not		~	

Feature Comparison: SSMA for Oracle will take care of operator assignment while migration.

Migration approach	SSMA take care of most of the operators while migrating. The following operators are not supported in the indexed view definition and SSMA marks their conversion with error messages UNION, MINUS, INTERSECT operators

References	https://msdn.microsoft.com/en-us/library/ms181091.aspx

Text Search

Feature ID	62
Feature	Text Search
Description	The SQL Repository has built-in support for Oracle's ConText full text search engine, which processes queries and returns information based on the content or themes of text stored in a text column of an Oracle database. To enable full text searching on columns, you must create ConText indexes for the columns. See your Oracle documentation for information about how to do this. **Note:** By default, an Oracle database rebuilds a full-text index after each commit. This behavior can cause a full deployment to hang indefinitely. To prevent this, you should configure ConText indexing to occur at regular intervals, using the following format: .
Category	SQL
To find feature enablement	
Feature usage	
Recommendation	**Feature Description**: Full-Text Search in SQL Server lets users and applications run full-text queries against character-based data in SQL Server tables. Before you can run full-text queries on a table, the database administrator must create a full-text index on the table. The full-text index includes one or more character-based columns in the table. These columns can have any of the following data types: **char**, **varchar**, **nchar**, **nvarchar**, **text**, **ntext**, **image**, **xml**, or **varbinary(max)** and FILESTREAM. Each full-text index indexes one or more columns from the table, and each column can use a specific language. Full-text queries perform linguistic searches against text data in full-text indexes by operating on words and phrases based on rules of a particular language such as English or Japanese. Full-text queries can include simple words and phrases or multiple forms of a word or phrase. A full-text query returns any documents that contain at least one match (also known as a *hit*). A match occurs when a target document contains all the terms specified in the full-text query, and meets any other search conditions, such as the distance between the matching terms. **Feature Comparison**: Migrating Oracle text search to SQL server involves index deletion and query rebuilding. SQL server support text search effectively without any changes into the index. Oracle provides text search using special indexes. In SQL server normal text search and full text searches available. These both differs in nature, but there is no need of any schema changes or special indexes are required.
Migration Approach	**SSMA** tool don't directly support complex text searches. The queries need to rewritten to support SQL server text searches. **To set up full-text search by using SSMS wizard**

	1. In Object Explorer, right-click the table on which you want to create a full-text index, point to Full-Text index, and then click Define Full-Text Index. This action launches the Wizard in a separate window. Click Next 2. Unique Index. Select an index from the drop down list. The index must be a single-key-column, unique, non-nullable index. Select the smallest unique key index for the full-text unique key. For best performance, a clustered index is recommended. 3. Available Columns. Check the box next to all column names for columns you want to include. check box next to the column name. Ineligible columns are greyed out and their check boxes disabled. 4. Language for Word Breaker. Select a language from the drop-down list. This choice will be used to identify the correct word breakers for the index. SQL Server uses word breakers to identify word boundaries in the full-text indexed data. 5. Type Column. Select the name of the column that holds the document type of column being full-text indexed. NOTE: The Type Column is enabled only when the column named in the Available Columns column is of type varbinary(max) or image. 6. Statistical Semantics. Select whether to enable semantic indexing for the selected column. 7. Select the change tracking options. o Automatically: Select this radio button to have the full-text index updated automatically as changes occur to the underlying data. o Manually: Select this radio button if you do not want the full-text index to be updated automatically as changes occur to the underlying data. Changes to the underlying data are maintained. However, to apply the changes to the full-text index you must start or schedule this process manually. o Do not track changes: Select this radio button if you do not want the full-text index to be updated with changes to the underlying data. 8. Start full population when index is created (Available only when you Do not track changes). Select this radio button to kick off a full population at the successful completion of this wizard. This will consist of creating the full-text index structure in the catalog and populating it with full-text indexed data. Click Next
References	https://msdn.microsoft.com/en-us/library/ms142571.aspx

Linked server instances

Feature ID	64
Feature	Linked server instances
Description	Use the CREATE DATABASE LINK statement to create a database link. A database link is a schema object in one database that enables you to access objects on another database. The other database need not be an Oracle Database system. However, to access non-Oracle systems you must use Oracle Heterogeneous Services. After you have created a database link, you can use it to refer to tables and views on the other database. In SQL statements, you can refer to a table or view on the other database by appending @dblink to the table or view name. You can query a table or view on the other database with the SELECT statement. You can also access remote tables and views using any INSERT, UPDATE, DELETE, or LOCK TABLE statement.
Category	SQL
To Find Feature Enablement	`select * from DBA_DB_LINKS;`
Feature Usage	`select * from USER_DB_LINKS;`
Recommendation	**Feature Description:** Configure a linked server to enable the SQL Server Database Engine to execute commands against OLE DB data sources outside of the instance of SQL Server. Typically, linked servers are configured to enable the Database Engine to execute a Transact-SQL statement that includes tables in another instance of SQL Server, or another database product such as Oracle. Many types OLE DB data sources can be configured as linked servers, including Microsoft Access and Excel. Linked servers offer the following advantages: • The ability to access data from outside of SQL Server. • The ability to issue distributed queries, updates, commands, and transactions on heterogeneous data sources across the enterprise. • The ability to address diverse data sources similarly. You can configure a linked server by using SQL Server Management Studio or by using the sp_addlinkedserver (Transact-SQL) statement. OLE DB providers vary greatly in the type and number of parameters required. For example some providers require you to provide a security context for the connection using sp_addlinkedsrvlogin (Transact-SQL). Some OLE DB providers allow SQL Server to update data on the OLE DB source. Others provide only read-only data access. For information about each OLE DB provider, consult documentation for that OLE DB provider.
Migration Approach	SSMA does not support migrating linked databases. We must manually link databases and databases should be configured first before linking them. To configure linked servers in sql server, one can use using SQL Server Management Studio or by using the sp_addlinkedserver (Transact-SQL) statement. OLE DB providers vary greatly in the type and number of parameters required. For example some providers require you to provide a security context for the connection using sp_addlinkedsrvlogin (Transact-SQL).

	Some OLE DB providers allow SQL Server to update data on the OLE DB source. Others provide only read-only data access. For information about each OLE DB provider, consult documentation for that OLE DB provider.
	Below are the steps to create a linked server to another instance of SQL server using SQL Server management studio.
	• In SQL Server Management Studio, open Object Explorer, expand **Server Objects**, right-click **Linked Servers**, and then click **New Linked Server**.
	• On the **General** page, in the **Linked server** box, type the name of the instance of **SQL Server** that you are linking to.
	• In the **Server type** area, select **SQL Server** to indicate that that the linked server is another instance of **SQL Server**.
	• On the **Security** page, specify the security context that will be used when the original SQL Server connects to the linked server.
	• Optionally, to view or specify server options, click the **Server Options** page and click ok.
	• To view the options that the provider makes available, click the **Providers Options** page.
	You can also create a linked server by using Transact-SQL , use the sp_addlinkedserver (Transact-SQL)CREATE LOGIN (Transact-SQL) and sp_addlinkedsrvlogin (Transact-SQL) statements.
	1. In Query Editor, enter the following Transact-SQL command to link to an instance of SQL Server named `SRVR002\ACCTG`:
	Transact-SQL
	`USE [master]`
	`GO`
	`EXEC master.dbo.sp_addlinkedserver`
	`@server = N'SRVR002\ACCTG',`
	`@srvproduct=N'SQL Server' ;`
	`GO`
	2. Execute the following code to configure the linked server to use the domain credentials of the login that is using the linked server.
	Transact-SQL
	`EXEC master.dbo.sp_addlinkedsrvlogin`
	`@rmtsrvname = N'SRVR002\ACCTG',`
	`@locallogin = NULL ,`
	`@useself = N'True' ;`
	`GO`
	For more information regarding this you can follow the below link,
	https://msdn.microsoft.com/en-us/library/ff772782.aspx

References	https://msdn.microsoft.com/en-us/library/ms188279.aspx https://docs.oracle.com/cd/B28359_01/server.111/b28310/ds_concepts002.htm#ADMIN12084

Service Broker

Feature ID	65
Feature	Service broker
Description	A messaging environment stores information in queues. **Enqueuing** is the process of placing messages into queues. **Dequeuing** is the process of retrieving messages from queues. The information in queues can be used to complete tasks, or it can be processed by applications. A messaging environment allows applications to communicate with each other asynchronously. That is, one application does not need to wait for another application to complete a task. Asynchronous communication means that a messaging system has minimal impact on the functionality of the applications that use the system. For example, when one application wants to communicate with another application, it can put messages in a queue. The messages can be stored in the queue until the other application retrieves them. In fact, one application might not be running while another application is enqueuing messages for it to process later. The messages might instruct the retrieving application to perform an action, or the messages might contain information that must be processed by the retrieving application. When an organization has several different systems that must communicate with each other, a messaging environment can be a good solution. The various systems might be in different locations, some might be older than others, and some might run on different platforms. Messaging provides a standard, reliable way to transport critical information between these systems.
Category	SQL
Recommendation	Feature Description: • Service Broker helps database developers build reliable and scalable applications. Because Service Broker is part of the Database Engine, administration of these applications is part of the routine administration of the database. • Service Broker provides queuing and reliable messaging for SQL Server. Service Broker is used both for applications that use a single SQL Server instance and applications that distribute work across multiple instances. • Within a single SQL Server instance, Service Broker provides a robust asynchronous programming model. Database applications typically use asynchronous programming to shorten interactive response time and increase overall application throughput. • Service Broker also provides reliable messaging between SQL Server instances. Service Broker helps developers compose applications from independent, self-contained components called services. • Applications that require the functionality exposed in these services use messages to interact with the services. Service Broker uses TCP/IP to exchange messages between instances. Service Broker includes features to help prevent unauthorized access from the network and to encrypt messages sent over the network.

Migration Approach	There is no direct migration approach available. We must manually create appropriate messages as per the architecture. There are documentations available for service broker in SQL Server 2016. There are different sections for this: • Data Definition statements: for CREATE, ALTER, and DROP statements • Service Broker Statements: SQL Server provides various service broker statements such as BEGIN CONVERSATION TIMER, BEGIN DIALOG CONVERSATION, END CONVERSATION, GET CONVERSATION GROUP etc. • Service Broker Catalog Views: Such as sys.conversation_endpoints, sys.conversation_groups, sys.conversation_priorities (Transact-SQL) etc. • Service Broker Related Dynamic Management Views: Such as sys.dm_broker_activated_tasks, sys.dm_broker_forwarded_messages etc. • ssbdiagnose Utiliy : The **ssbdiagnose** utility reports issues in Service Broker conversations or the configuration of Service Broker services. Configuration checks can be made for either two services or a single service. Issues are reported either in the command prompt window as human-readable text, or as formatted XML that can be redirected to a file or another program. For more information on different Service broker features, you can use the following url: https://msdn.microsoft.com/en-GB/Library/bb522893(v=sql.105).aspx
References	https://docs.oracle.com/cd/E15438_01/doc.50/e15180/cpt_platform_overview.htm

Processes and Threads

Feature ID	66
Feature	Processes and Threads
Description	A process is a mechanism in an operating system that can run a series of steps. The process execution architecture depends on the operating system. For example, on Windows an Oracle background process is a thread of execution within a process. On Linux and UNIX, an Oracle process is either an operating system process or a thread within an operating system process. Processes run code modules. All connected Oracle Database users must run the following modules to access a database instance: • Application or Oracle Database utility A database user runs a database application, such as a precompiler program or a database tool such as SQL*Plus, that issues SQL statements to a database. • Oracle database code Each user has Oracle database code executing on his or her behalf that interprets and processes the application's SQL statements. A process normally runs in its own private memory area. Most processes can periodically write to an associated trace file (see "Trace Files").
Category	SQL

To Find Feature Enablement	`SELECT sess.process, sess.status, sess.username, sess.schemaname, sql.sql_text`
	`  FROM v$session sess,`
	`       v$sql     sql`
	` WHERE sql.sql_id(+) = sess.sql_id`
	`   AND sess.type     = 'USER'`

Feature Usage	`select * from V$THREAD;`
	
	`SELECT sess.process, sess.status, sess.username, sess.schemaname, sql.sql_text`
	`  FROM v$session sess,`
	`       v$sql     sql`
	` WHERE sql.sql_id(+) = sess.sql_id`
	`   AND sess.type     = 'USER'`

Recommendation	**Feature Description**: In SQL Server by using SQL Server Management Studio or Transact-SQL. The max worker threads option configures the number of worker threads that are available to SQL Server processes. SQL Server uses the native thread services of the operating systems so that one or more threads support each network that SQL Server supports simultaneously, another thread handles

<table>
<tr><td></td><td>database checkpoints, and a pool of threads handles all users. The default value for max worker threads is 0. This enables SQL Server to automatically configure the number of worker threads at startup. The default setting is best for most systems. However, depending on your system configuration, setting max worker threads to a specific value sometimes improves performance.</td></tr>
<tr><td>Migration Approach</td><td>

Thread pooling helps optimize performance when large numbers of clients are connected to the server. Usually, a separate operating system thread is created for each query request. However, with hundreds of connections to the server, using one thread per query request can consume large amounts of system resources. The max worker threads option enables SQL Server to create a pool of worker threads to service a larger number of query requests, which improves performance.
The following table shows the automatically configured number of max worker threads for various combinations of CPUs and versions of SQL Server.
<table>
<tr><th>Number of CPUs</th><th>32-bit computer</th><th>64-bit computer</th></tr>
<tr><td><= 4 processors</td><td>256</td><td>512</td></tr>
<tr><td>8 processors</td><td>288</td><td>576</td></tr>
<tr><td>16 processors</td><td>352</td><td>704</td></tr>
<tr><td>32 processors</td><td>480</td><td>960</td></tr>
<tr><td>64 processors</td><td>736</td><td>1472</td></tr>
<tr><td>128 processors</td><td>4224</td><td>4480</td></tr>
</table>

</td></tr>
</table>

		256 processors	8320	8576	
References	https://msdn.microsoft.com/en-us/library/ms190219.aspx				

In-memory Optimization

Feature ID	69
Feature	In-memory optimization
Description	The Oracle Database 12c In-Memory Option is based on a dual-format data store: • Data are persistently stored on disk, and they are stored in row format only • Whenever data are requested for read/write operations (data manipulations), they are loaded into the traditional Row Store (Buffer Cache) • Whenever data are requested for read-only operations, they are populated into a new In-Memory Column Store. This population, of course, includes a transformation from row to columnar format • Whenever a transaction that includes inserts, updates, or deletes is committed, the new data will immediately and simultaneously appear in both the row store and the in-memory column store. Therefore both stores are transactionally consistent Note that this approach does not necessarily require more memory. There is no need to populate the same data in both stores. If they are required for OLTP only, they will not be populated into the column store, and if they are used for DSS only, they will not be kept in the row store. In addition (as we will see shortly) it is possible to restrict the data populated into the in-memory column store to subsets of the table data.
Category	SQL
To Find Feature Enablement	`select * from v$sga;` `select * from V$SGASTAT;`

Feature Usage	`SELECT to_char(ssn.sid, '9999') || ' - ' || nvl(ssn.username,`
	`nvl(bgp.name, 'background')) ||`
	`nvl(lower(ssn.machine), ins.host_name) "SESSION",`
	`to_char(prc.spid, '999999999') "PID/THREAD",`
	`to_char((se1.value/1024)/1024, '999G999G990D00') || ' MB' " CURRENT SIZE",`
	`to_char((se2.value/1024)/1024, '999G999G990D00') || ' MB' " MAXIMUM SIZE"`
	`FROM v$sesstat se1, v$sesstat se2, v$session ssn, v$bgprocess bgp, v$process prc,`
	`v$instance ins, v$statname stat1, v$statname stat2`
	`WHERE se1.statistic# = stat1.statistic# and stat1.name = 'session pga memory'`
	`AND se2.statistic# = stat2.statistic# and stat2.name = 'session pga memory max'`
	`AND se1.sid = ssn.sid`
	`AND se2.sid = ssn.sid`
	`AND ssn.paddr = bgp.paddr (+)`
	`AND ssn.paddr = prc.addr (+);`

Recommendation	**Feature Description:** In-Memory OLTP, also known as 'Hekaton' and 'In-Memory Optimization', is Microsoft's latest in-memory processing technology. In-Memory OLTP is optimized for Online Transaction Processing (OLTP). It is integrated into SQL Server's Database Engine and can be used in the exact same manner as any other Database Engine component. In-Memory OLTP originally shipped with SQL Server 2014 and it mainly features two new data structures which are Memory-Optimized Tables, and Natively-Compiled Stored Procedures.

	Memory-optimized tables Memory-optimized tables store their data into memory using multiple versions of each row's data. This technique is characterized as 'non-blocking multi-version optimistic concurrency control' and eliminates both locks and latches, thereby achieving significant performance advantages. The main features of memory-optimized tables are: • Rows in the table are read from, and written to, memory • The entire table resides in memory • Non-blocking multi-version optimistic concurrency control • The option of durable & non-durable data • A second copy is maintained on disk for durability (if enabled) • Data in memory-optimized tables is only read from disk during database recovery • It is interoperable with disk-based tables
Migration Approach	SQL Server 2014 introduced In-Memory OLTP database concept which improves OLTP database performance. The In-Memory OLTP feature includes memory-optimized tables, table types and native compilation of stored procedures for efficient access to these tables. Memory-optimized tables is an advanced technology of table storage that provides high speed of data access due to holding data in memory. Memory-optimized tables are based on special OLTP engine (together with In-Memory Precompiled procedures). The increased speed of memory-optimized tables processing allows to reproduce processing of Oracle tables build on hash cluster index. AS memory-optimized tables reside in memory, rows in the table are read from and written to memory. A second copy of the table data is maintained on disk, but only for durability purposes. Each row in the table potentially has multiple versions. This row versioning is used to allow concurrent reads and writes on the same row. SSMA allows migrating Oracle tables to memory-optimized tables in SQL Server. For Tables node in Oracle Metadata Explorer there is In Memory tab on the right pane of SSMA window. It allows checking the tables you want to migrate to memory-optimized ones. Another way to check a table for conversion to memory-optimized tables is clicking on the table name under Tables node in Oracle Metadata Explorer and check Convert to memory optimized table check box on In Memory tab on the right pane of SSMA window. DDL syntax for creating memory-optimized table is as follows: `CREATE TABLE database_name.schema_name.table_name (column_name data_type [COLLATE collation_name] [NOT] NULL [DEFAULT constant_expression] [IDENTITY] [PRIMARY KEY NONCLUSTERED [HASH WITH (BUCKET_COUNT = bucket_count)]] [INDEX index_name [NONCLUSTERED [HASH WITH (BUCKET_COUNT = bucket_count)]]] [,…] [PRIMARY KEY { NONCLUSTERED HASH (column [,…]) WITH (BUCKET_COUNT = bucket_count)	NONCLUSTERED`

	`(column [ASC|DESC] [,…] ) }  }]  [INDEX index_name  {    NONCLUSTERED HASH (column [,…]) WITH (BUCKET_COUNT = bucket_count) |   NONCLUSTERED      (column [ASC|DESC] [,…] ) }  }] [,…]`
References	https://www.simple-talk.com/sql/learn-sql-server/introducing-sql-server-in-memory-oltp/

Connection Pooling

Feature ID	71
Feature	Connection Pooling
Description	Connection pooling is generally the practice of a middle tier (application server) getting N connections to a database (say 20 connections). These connections are stored in a pool in the middle tier, an "array" if you will. Each connection is set to "not in use" When a user submits a web page to the application server, it runs a piece of your code, your code says "i need to get to the database", instead of connecting right there and then (that takes time), it just goes to this pool and says "give me a connection please". the connect pool software marks the connection as "in use" and gives it to you. You generate the page, format the html whatever -- and then return the connection to the pool where someone else can use it. In this fashion, using 20 connections to the database, you can avoid the connect/disconnect overhead and (more importantly perhaps) you can service a community of hundreds of users simultaneously using just 20 connections! (as opposed to client server where by hundreds of simultaneous users would take hundreds of connections!)
Category	SQL
To Find Feature Enablement	select substr(a.spid,1,9) pid, substr(b.sid,1,5) sid, substr(b.serial#,1,5) ser#, substr(b.machine,1,6) box, substr(b.username,1,10) username, -- b.server, substr(b.osuser,1,8) os_user, substr(b.program,1,30) program from v$session b, v$process a where b.paddr = a.addr and type='USER' order by spid;
Feature Usage	select substr(a.spid,1,9) pid, substr(b.sid,1,5) sid, substr(b.serial#,1,5) ser#, substr(b.machine,1,6) box, substr(b.username,1,10) username,

	`--  b.server,`
	`  substr(b.osuser,1,8) os_user,`
	`  substr(b.program,1,30) program`
	`from v$session b, v$process a`
	`where`
	`b.paddr = a.addr`
	`and type='USER'`
	`order by spid;`

Recommendation	**Feature Description:** Connection pooling reduces the number of times that new connections must be opened. The pooler maintains ownership of the physical connection. It manages connections by keeping alive a set of active connections for each given connection configuration. Whenever a user calls Open on a connection, the pooler looks for an available connection in the pool. If a pooled connection is available, it returns it to the caller instead of opening a new connection. When the application calls Close on the connection, the pooler returns it to the pooled set of active connections instead of closing it. Once the connection is returned to the pool, it is ready to be reused on the next Open call. Pooling connections can significantly enhance the performance and scalability of your application. By default, connection pooling is enabled in ADO.NET. Connection pooling in SQL server can be done through clients. This depends on the SQL server connection availability and settings. **Feature Comparison:** SQL server support connection pooling by default, the total number of connections supported by SQL server is 32767

Migration Approach	There is no direct Migration approach, you need to configure connection polling on SQL Server.
	
	SQL Server allows a maximum of 32767 connections by default.
	
	Using the user connections option helps avoid overloading the server with too many concurrent connections. You can estimate the number of connections based on system and user requirements. For example, on a system with many users, each user would not usually require a unique connection. Connections can be shared among users.
	
	It can be altered using sp_configure.
	
	T-SQL to configure the user connections option:
	`USE AdventureWorks2012 ;`
	`GO`
	`EXEC sp_configure 'show advanced options', 1;`
	`GO`
	`RECONFIGURE ;`
	`GO`
	`EXEC sp_configure 'user connections', 325 ;`
	`GO`
	`RECONFIGURE;`
	`GO`

	In SSMS, under Object explorer under Connections node, in the Max number of concurrent connections box, type or select a value from 0 through 32767 to set the maximum number of users that are allowed to connect simultaneously to the instance of SQL Server.
	
	To view your current configuration for this setting, use the following query:
	`select * from sys.configurations where name ='user connections'`
	
	By default, you should see a maximum of 32767, `value_in_use` equal to 0 (use the default settings). If this has been altered, you can reconfigure SQL Server to use other values.
	
	You should also review how many connections are *actually* being made, as there might be more activity outside your application (or your application is making more connections than you think).
	
	You will want to look at General Statistics -> Logical Connections in either perfmon or query the values in `sys.dm_os_performance_counters` (cntr_value will show the current point in time value):
	
	`select * from sys.dm_os_performance_counters where counter_name ='User Connections'`

References	https://msdn.microsoft.com/en-us/library/8xx3tyca(v=vs.110).aspx (SQL Server Connection Pooling- ADO.NET) https://technet.microsoft.com/en-us/library/ms187030.aspx (Configure the user connections Server Configuration Option)

Connection Multiplexing

Feature ID	70
Feature	Connection Multiplexing
Description	Oracle Connection Manager Configuration Overview Oracle Connection Manager is a proxy server, an intermediate server that forwards connection requests to database servers or to other proxy servers. It has two primary functions: Session multiplexing Access control With session multiplexing, you can quickly enable Oracle Connection Manager to funnel multiple client sessions through a network connection to a shared server destination. With access control, you can use rule-based configuration to filter out certain client requests and accept others.
Category	SQL
To find feature enablement	
Feature Usage	
Recommendation	**Feature Comparison**: SQL Server sessions are equivalent to session multiplexing in Oracle. Basically, in SQL server sessions are 1:1 relationship with connections. In some cases, a connection may have two or more sessions such as service broker session. Sessions – when the client application connects to SQL Server the two sides establish a "session" on which to exchange information. Strictly speaking a session is not the same as the underlying physical connection, it is a SQL Server logical representation of a connection. But for practical purposes, you can think of this as being a connection (session =~ connection). See sys.dm_exec_sessions. This is the old SPID that existed in SQL Server 2000 and earlier. You may sometimes notice a single session repeating multiple times in a DMV output. This happens because of parallel queries. A parallel query uses the same session to communicate with the client, but on the SQL Server side multiple worker (threads) are assigned to service this request. So, if you see multiple rows with the same session ID, know that the query request is being serviced by multiple threads. As per the migration perspective there is no much impact in connection multiplexing.
Migration Approach	There is not migration approach available for connection multiplexing. This needs to be done manually. Here is concept behind **SQL Server Thread Management and Scheduling:** **Scheduler (SOS Scheduler)** the object that manages thread scheduling in SQL Server and allows threads to be exposed to the CPU (described in sys.dm_os_schedulers). This is the all-powerful but benign and graceful master whom everyone abides. He does not control things but lets the workers work with each other and relies on their cooperation (co-operative scheduling mode). Each scheduler /master (one per logical CPU) accepts new tasks and hands them off to workers. SOS Scheduler allows one worker at a time to be exposed to the CPU. **Task**

	a task represents the work that needs to be performed (sys.dm_os_tasks). A task contains one of the following requests: query request (RPC event or Language event), a prelogin request (prelogin event), a login request (connect event), a logout request (disconnect event), a query cancellation request (an Attention event), a bulk load request (bulk load event), a distributed transaction request (transaction manager event). A task is what the Master is about – it is what defines its existence. Note these are tracked at the SOS scheduler layer (thus dm_OS_tasks) **Worker** (worker thread) This is the logical SQL Server representation of a thread (think of it as a wrapper on top of the OS thread). It is a structure within the Scheduler which maintains SQL Server-specific information about what a worker thread does. sys.dm_os_workers. Workers are the humble servants who carry out the task assigned to them by the Master (scheduler). **Thread** this is the OS thread sys.dm_os_threads that is created via calls like CreateThread()/_beginthreadex(). A Worker is mapped 1-to-1 to a Thread. **Request** Request is the logical representation of a query request made from the client application to SQL Server (sys.dm_exec_requests). This query request has been assigned to a task that the scheduler hands off to a worker to process. This represents query requests as well as system thread operations (like checkpoint, log writer, etc); you will not find login, logouts, attentions and the like here. Also, note that this is a representation at the SQL execution engine level (thus dm_EXEC_requests) not at the SOS Scheduler layer. **Sessions** when the client application connects to SQL Server the two sides establish a "session" on which to exchange information. Strictly speaking a session is not the same as the underlying physical connection, it is a SQL Server logical representation of a connection. But for practical purposes, you can think of this as being a connection (session =~ connection). T-SQL to find the users that are connected to the server and returns the number of sessions for each user: SELECT login_name ,COUNT(session_id) AS session_count FROM sys.dm_exec_sessions GROUP BY login_name; You may sometimes notice a single session repeating multiple times in a DMV output. This happens because of parallel queries. A parallel query uses the same session to communicate with the client, but on the SQL Server side multiple worker (threads) are assigned to service this request. So if you see multiple rows with the same session ID, know that the query request is being serviced by multiple threads. **Connections** this is the actual physical connection established at the lower protocol level with all of its characteristics sys.dm_exec_connections . There is a 1:1 mapping between a Session and a Connection.

References	https://blogs.msdn.microsoft.com/sqlmeditation/2012/12/13/tasks-workers-threads-scheduler-sessions-connections-requests-what-does-it-all-mean/

Database Queueing

Feature ID	72
Feature	Database queuing
Description	When Web-based business applications communicate with each other, **producer** applications **enqueue** messages and **consumer** applications **dequeue** messages. At the most basic level of queuing, one producer enqueues one or more messages into one **queue**. Each **message** is dequeued and processed once by one of the consumers. A message stays in the queue until a consumer dequeues it or the message expires. A producer can stipulate a delay before the message is available to be consumed, and a time after which the message expires. Likewise, a consumer can wait when trying to dequeue a message if no message were available. An agent program or application could act as both a producer and a consumer. Producers can enqueue messages in any sequence. Messages are not necessarily dequeued in the order in which they are enqueued. Messages can be enqueued without being dequeued. At a slightly higher level of complexity, many producers enqueue messages into a queue, all of which are processed by one consumer. Or many producers enqueue messages, each message being processed by a different consumer depending on type and correlation identifier. Enqueued messages are said to be propagated when they are reproduced on another queue, which can be in the same database or in a remote database. Applications often use data in different formats. A transformation defines a mapping from one data type to another. The transformation is represented by a SQL function that takes the source data type as input and returns an object of the target data type. You can arrange transformations to occur when a message is enqueued, when it is dequeued, or when it is propagated to a remote subscriber.
Category	SQL
To find Feature Enabled	`SELECT * FROM ALL_QUEUES;`

Feature usage	`SELECT * FROM USER_QUEUES WHERE QUEUE_TYPE='EXCEPTION_QUEUE';`
	`SELECT * FROM USER_QUEUES WHERE QUEUE_TYPE='NON_PERSISTENT_QUEUE';`
	`SELECT * FROM USER_QUEUES WHERE QUEUE_TYPE='NORMAL_QUEUE';`

Recommendation	**Feature Description:** With Service Broker, a feature in Microsoft SQL Server, internal or external processes can send and receive guaranteed, asynchronous messages by using extensions to Transact-SQL Data Manipulation Language (DML). Messages can be sent to a queue in the same database as the sender, to another database in the same SQL Server instance, or to another SQL Server instance either on the same server or on a remote server. To better understand Service Broker, familiarity with the key concepts of queues, dialogs, conversation groups, and activation is helpful. These are discussed briefly in this section.

	Feature Comparison: Service broker in SQL server is equivalent to database queuing in Oracle. Application systems and database processes must be redesigned or reconfigured to use SQL Broker.
Migration Approach	There is no migration approach available. This needs to be done manually. However, there are documentations available for service broker in SQL Server 2016 which is a similar feature like databsse queueing in oracle. There are different sections for this: • Data Definition statements: for CREATE, ALTER, and DROP statements • Service Broker Statements: SQL Server provides various service broker statements such as BEGIN CONVERSATION TIMER, BEGIN DIALOG CONVERSATION, END CONVERSATION, GET CONVERSATION GROUP etc. • Service Broker Catalog Views: Such as sys.conversation_endpoints, sys.conversation_groups, sys.conversation_priorities (Transact-SQL) etc. • Service Broker Related Dynamic Management Views: Such as sys.dm_broker_activated_tasks, sys.dm_broker_forwarded_messages etc. • ssbdisgnose Utiliy : The **ssbdiagnose** utility reports issues in Service Broker conversations or the configuration of Service Broker services. Configuration checks can be made for either two services or a single service. Issues are reported either in the command prompt window as human-readable text, or as formatted XML that can be redirected to a file or another program. For more information on different Service broker features, you can use the following url: https://msdn.microsoft.com/en-GB/Library/bb522893(v=sql.105).aspx
References	http://docs.oracle.com/cd/B10501_01/appdev.920/a96587/qintro.htm https://msdn.microsoft.com/en-us/library/bb522893.aspx

Oracle resource profiler

Feature ID	103
Feature	Oracle resource profiler
Description	The DBMS_PROFILER package provides an interface to profile existing PL/SQL applications and identify performance bottlenecks. You can then collect and persistently store the PL/SQL profiler data. This package enables the collection of profiler (perfoprmance) data for performance improvement or for determining code coverage for PL/SQL applications. Application developers can use code coverage data to focus their incremental testing efforts. With this interface, you can generate profiling information for all named library units that are executed in a session. The profiler gathers information at the PL/SQL virtual machine level. This information includes the total number of times each line has been executed, the total amount of time that has been spent executing that line, and the minimum and maximum times that have been spent on a particular execution of that line.
Category	SQL
To Find Feature Enablement	`SELECT profile, resource_name, limit FROM dba_profiles ORDER BY 1,2;` `SELECT username, profile FROM dba_users WHERE account_status = 'OPEN' ORDER BY 1;`
Feature usage	`SELECT profile, resource_name, limit FROM dba_profiles ORDER BY 1,2;` `SELECT username, profile FROM dba_users WHERE account_status = 'OPEN' ORDER BY 1;`
Recommendation	**Feature Description:** Microsoft SQL Server Profiler is a graphical user interface to SQL Trace for monitoring an instance of the Database Engine or Analysis Services. You can capture and save data about each event to a file or table to analyze later. For example, you can monitor a production environment to see which stored procedures are affecting performance by executing too slowly. SQL Server Profiler is used for activities such as: • Stepping through problem queries to find the cause of the problem. • Finding and diagnosing slow-running queries. • Capturing the series of Transact-SQL statements that lead to a problem. The saved trace can then be used to replicate the problem on a test server where the problem can be diagnosed. • Monitoring the performance of SQL Server to tune workloads. For information about tuning the physical database design for database workloads, see Database Engine Tuning Advisor. • Correlating performance counters to diagnose problems. SQL Server Profiler also supports auditing the actions performed on instances of SQL Server. Audits record security-related actions for later review by a security administrator.

	Feature Comparison: SQL server profiler is equivalent to Oracle resource profiler.
Migration Approach	No migration approach available for this feature. • However, in SQL Server, you can start SQL Server Profiler in several different ways to support gathering trace output in a variety of scenarios. You can start SQL Server Profiler include from the **Start** menu, from the **Tools** menu in Database Engine Tuning Advisor, and from several locations in SQL Server Management Studio. • When you first start SQL Server Profiler and select **New Trace** from the **File** menu, the application displays a **Connect to Server** dialog box where you can specify the SQL Server instance to which you want to connect. **To start SQL Server Profiler from the Start menu** 1. On the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Performance Tools**, and then click **SQL Server Profiler**. **To start SQL Server Profiler in Database Engine Tuning Advisor** 1. On the Database Engine Tuning Advisor **Tools** menu, click **SQL Server Profiler**. **Starting SQL Server Profiler in Management Studio** SQL Server Management Studio starts each profiler session in its own instance and continues to run if you shutdown SQL Server Management Studio. You can start SQL Server Profiler from several locations in SQL Server Management Studio, as illustrated in the following procedures. When SQL Server Profiler starts, it loads the connection context, trace template, and filter context of its launch point. *To start SQL Server Profiler from the Tools menu* 1. In the SQL Server Management Studio **Tools** menu, click **SQL Server Profiler**. *To start SQL Server Profiler from the Query Editor* 1. On the SQL Server Management Studio menu bar, click **New Query**. 2. In Query Editor, right-click and then select **Trace Query in SQL Server Profiler**. *To start SQL Server Profiler from Activity Monitor* 1. In Object Explorer, right-click an instance of SQL Server, and then click **Activity Monitor**. 2. Click the **Processes** pane, right-click the process that you want to profile, and then click **Trace Process in SQL Server Profiler**. For more information on how to use SQL Server profiler use the following url: https://msdn.microsoft.com/en-in/library/ff650699.aspx

Bitmap indexes

Feature ID	83
Feature	Bitmap indexes
Description	Oracle bitmap indexes are very different from standard b-tree indexes. In bitmap structures, a two-dimensional array is created with one column for every row in the table being indexed. Each column represents a distinct value within the bitmapped index. This two-dimensional array represents each value within the index multiplied by the number of rows in the table. At row retrieval time, Oracle decompresses the bitmap into the RAM data buffers so it can be rapidly scanned for matching values. These matching values are delivered to Oracle in the form of a Row-ID list, and these Row-ID values may directly access the required information. The real benefit of bitmapped indexing occurs when one table includes multiple bitmapped indexes. Each individual column may have low cardinality. The creation of multiple bitmapped indexes provides a very powerful method for rapidly answering difficult SQL queries.
Category	SQL
To Find Feature Enablement	`SELECT profile, resource_name, limit FROM dba_profiles ORDER BY 1,2;` `SELECT username, profile FROM dba_users WHERE account_status = 'OPEN' ORDER BY 1;`

Feature Usage	`SELECT *`
	`  FROM user_indexes`
	`WHERE index_type IN ('BITMAP', 'FUNCTION-BASED BITMAP' );`

Recommendation	**Feature comparison:** There is no bitmap index in SQL Server rather SQL server bitmap filtering which is not direct equivalent. The bitmap filter compares favorably to the bitmap index. A bitmap index is an alternate form for representing row ID (RID) lists in a value-list index using one or more bit vectors indicating which row in a table contains a certain column value. Both can be very effective in removing unnecessary rows from result processing; however, there are important differences between a bitmap filter and a bitmap index. First, bitmap filters are in-memory structures, thus eliminating any index maintenance overhead due to data manipulation language (DML) operations made to the underlying table. In addition, bitmap filters are very small and, unlike existing on-disk indexes that typically depend on the size of the table on which they are built, bitmap filters can be created dynamically with minimal impact on query processing time. Other than bitmap filtering there is also optimized bitmap filtering. Bitmap filtering and optimized bitmap filtering are implemented in the query plan by using the bitmap showplan operator. Bitmap filtering is applied only in parallel query plans in which hash or merge joins are used. Optimized bitmap filtering is applicable only to parallel query plans in which hash joins are used. Optimized bitmap filters have the following advantages: • Filtering from several dimension tables is supported. • Multiple filters can be applied to a single operator.

	• Optimized bitmap filters can be applied to more operator types. These include exchange operators such as the Distribute Streams and Repartition Streams operators, table or index scan operators, and filter operators. • Filtering is applicable to SELECT statements and the read-only operators used in INSERT, UPDATE, DELETE, and MERGE statements. • Filtering is applicable to the creation of indexed views in the operators used to populate the index. • The optimizer uses cardinality and cost estimates to determine if optimized bitmap filtering is appropriate. • The optimizer can consider more plans.

Migration approach	Bitmap filtering and optimized bitmap filtering need to be considered for this feature. This needs to be done manually.
	
	Implementing optimized bitmap filter:
	A bitmap filter is useful only if it is selective. The query optimizer determines when an optimized bitmap filter is selective enough to be useful and to which operators the filter is applied. The optimizer places the optimized bitmap filters on all branches of a star join and uses costing rules to determine whether the plan provides the smallest estimated execution cost. When the optimized bitmap filter is nonselective, the cost estimate is usually too high and the plan is discarded. When considering where to place optimized bitmap filters in the plan, the optimizer looks for hash join variants such as a right-deep stack of hash joins. Joins with dimension tables are implemented to execute the likely most selective join first.
	The operator in which the optimized bitmap filter is applied contains a bitmap predicate in the form of PROBE([Opt_Bitmap1001], {[column_name]} [, 'IN ROW']). The bitmap predicate reports on the following information:
	• The bitmap name that corresponds to the name introduced in the Bitmap operator. The prefix 'Opt_' indicates an optimized bitmap filter is used.
	• The column probed against. This is the point from which the filtered data flows through the tree.
	• Whether the bitmap probe uses in-row optimization. When it is, the bitmap probe is invoked with the IN ROW parameter. Otherwise, this parameter is missing.
	
	The following example represents a query against a simple star schema. The two dimension tables DimProduct and DimCustomer join to the fact table FactInternetSales using a primary-key-to-foreign-key join on a single integer column.
	
	`USE AdventureWorksDW2008R2;`
	`GO`
	`SELECT *`
	`FROM dbo.FactInternetSales AS F`
	`INNER JOIN dbo.DimProduct AS D1 ON F.ProductKey = D1.ProductKey`
	`INNER JOIN dbo.DimCustomer AS D2 ON F.CustomerKey = D2.CustomerKey`
	`WHERE D1.StandardCost <= 30 AND D2.YearlyIncome <= 50000;`

References	https://msdn.microsoft.com/en-us/library/ms181091.aspx https://technet.microsoft.com/en-us/library/bb522541(v=sql.105).aspx

Oracle parallel query

Feature ID	84
Feature	Oracle parallel query
Description	Without the parallel query feature, the processing of a SQL statement is always performed by a single server process. With the parallel query feature, multiple processes can work together simultaneously to process a single SQL statement. This capability is called parallel query processing. By dividing the work necessary to process a statement among multiple server processes, the Oracle Server can process the statement more quickly than if only a single server process processed it. The parallel query feature can dramatically improve performance for data-intensive operations associated with decision support applications or very large database environments. Symmetric multiprocessing (SMP), clustered, or massively parallel systems gain the largest performance benefits from the parallel query feature because query processing can be effectively split up among many CPUs on a single system. It is important to note that the query is parallelized dynamically at execution time. Thus, if the distribution or location of the data changes, Oracle automatically adapts to optimize the parallelization for each execution of a SQL statement.
Category	SQL
To Find Feature Enablement	`SELECT * FROM GV$SYSSTAT WHERE name LIKE 'Parallel operation%';`
Feature Usage	`SELECT * FROM GV$SYSSTAT WHERE name LIKE 'Parallel operation%' AND CLASS=1;`
Recommendation	**Feature Comparison**: In SQL Server, parallel query is possible. Parallelism can be achieved by parallel plan generated for the queries to run parallel. Migrating from Oracle would take significant effort. **Feature Description**: SQL Server provides parallel queries to optimize query execution and index operations for computers that have more than one microprocessor (CPU). Because SQL Server can perform a query or index operation in parallel by using several operating system threads, the operation can be completed quickly and efficiently. During query optimization, SQL Server looks for queries or index operations that might benefit from parallel execution. For these queries, SQL Server inserts exchange operators into the query execution plan to prepare the query for parallel execution. An exchange operator is an operator in a query execution plan that provides process management, data redistribution, and flow control. The exchange operator includes the Distribute Streams, Repartition Streams, and Gather Streams logical operators as subtypes, one or more of which can appear in the Showplan output of a query plan for a parallel query.
Migration Approach	PARALLEL_ENABLE will not be considered by SSMA. There is not automatic migration for parallel queries. However, SQL Server uses parallel query processing to optimize its performance.

	During query optimization, SQL Server looks for queries or index operations that might benefit from parallel execution. For these queries, SQL Server inserts exchange operators into the query execution plan to prepare the query for parallel execution. An exchange operator is an operator in a query execution plan that provides process management, data redistribution, and flow control. The exchange operator includes the Distribute Streams, Repartition Streams, and Gather Streams logical operators as subtypes, one or more of which can appear in the showplan output of a query plan for a parallel query. After exchange operators are inserted, the result is a parallel-query execution plan. A parallel-query execution plan can use more than one thread. A serial execution plan, used by a nonparallel query, uses only one thread for its execution. The actual number of threads used by a parallel query is determined at query plan execution initialization and is determined by the complexity of the plan and the degree of parallelism. Degree of parallelism determines the maximum number of CPUs that are being used; it does not mean the number of threads that are being used. The degree of parallelism value is set at the server level and can be modified by using the sp_configure system stored procedure. You can override this value for individual query or index statements by specifying the MAXDOP query hint or MAXDOP index option.
References	http://docs.oracle.com/cd/B19306_01/server.102/b14223/usingpe.htm https://technet.microsoft.com/en-us/library/ms178065(v=sql.105).aspx

Oracle Streams

Feature ID	90
Feature	Oracle Streams
Description	Oracle Streams enables information sharing. Using Oracle Streams, each unit of shared information is called a **message**, and you can share these messages in a stream. The stream can propagate information within a database or from one database to another. The stream routes specified information to specified destinations. The result is a feature that provides greater functionality and flexibility than traditional solutions for capturing and managing messages, and sharing the messages with other databases and applications. Oracle Streams provides the capabilities needed to build and operate distributed enterprises and applications, data warehouses, and high availability solutions. You can use all the capabilities of Oracle Streams at the same time. If your needs change, then you can implement a new capability of Oracle Streams without sacrificing existing capabilities. Using Oracle Streams, you control what information is put into a stream, how the stream flows or is routed from database to database, what happens to messages in the stream as they flow into each database, and how the stream terminates. By configuring specific capabilities of Oracle Streams, you can address specific requirements. Based on your specifications, Oracle Streams can capture, stage, and manage messages in the database automatically, including, but not limited to, data manipulation language (DML) changes and data definition language (DDL) changes. You can also put user-defined messages into a stream, and Oracle Streams can propagate the information to other databases or applications automatically. When messages reach a destination, Oracle Streams can consume them based on your specifications.
Category	SQL

Find Feature Enabled	`SELECT q.OWNER, q.NAME, t.QUEUE_TABLE, q.USER_COMMENT`
	`FROM DBA_QUEUES q, DBA_QUEUE_TABLES t`
	`WHERE t.OBJECT_TYPE = 'SYS.ANYDATA' AND`
	`q.QUEUE_TABLE = t.QUEUE_TABLE AND`
	`q.OWNER = t.OWNER;`

Feature Usage	`SELECT * FROM USER_OBJECTS WHERE OBJECT_TYPE IN ('PACKAGE') and OBJECT_NAME='DBMS_STREAMS_ADM';`
Recommendation	The equivalent feature in SQL server is Server Service Broker.

	SQL Server Service Broker provides native support for messaging and queuing applications in the SQL Server Database Engine. This makes it easier for developers to create sophisticated applications that use the Database Engine components to communicate between disparate databases. Developers can use Service Broker to easily build distributed and reliable applications. Application developers who use Service Broker can distribute data workloads across several databases without programming complex communication and messaging internals. This reduces development and test work because Service Broker handles the communication paths in the context of a conversation. It also improves performance. For example, front-end databases supporting Web sites can record information and send process intensive tasks to queue in back-end databases. Service Broker ensures that all tasks are managed in the context of transactions to assure reliability and technical consistency.
Migration Approach	There is no direct migration approach available this needs to be manually rewritten with Server service broker feature. There are documentations available for service broker in SQL Server 2016. There are different sections for this: • Data Definition statements: for CREATE, ALTER, and DROP statements • Service Broker Statements: SQL Server provides various service broker statements such as BEGIN CONVERSATION TIMER, BEGIN DIALOG CONVERSATION, END CONVERSATION, GET CONVERSATION GROUP etc. • Service Broker Catalog Views: Such as sys.conversation_endpoints, sys.conversation_groups, sys.conversation_priorities (Transact-SQL) etc. • Service Broker Related Dynamic Management Views: Such as sys.dm_broker_activated_tasks, sys.dm_broker_forwarded_messages etc. • ssbdisgnose Utiliy : The **ssbdiagnose** utility reports issues in Service Broker conversations or the configuration of Service Broker services. Configuration checks can be made for either two services or a single service. Issues are reported either in the command prompt window as human-readable text, or as formatted XML that can be redirected to a file or another program. For more information on different Service broker features, you can use the following url: https://msdn.microsoft.com/en-GB/Library/bb522893(v=sql.105).aspx
References	https://docs.oracle.com/cd/B28359_01/server.111/b28321/strms_over.htm#strms_overview https://msdn.microsoft.com/en-us/library/bb522893.aspx

SQL Optimizer Plan Stability (Stored Outlines)

Feature ID	79
Feature	SQL Optimizer plan stability (stored outlines)
Description	A stored outline is a collection of hints associated with a specific SQL statement that allows a standard execution plan to be maintained, regardless of changes in the system environment or associated statistics. Plan stability is based on the preservation of execution plans at a point in time where the performance of a statement is considered acceptable. The outlines are stored in the OL$, OL$HINTS, and OL$NODES tables, but the [USER|ALL|DBA]_OUTLINES and [USER|ALL|DBA]_OUTLINE_HINTS views should be used to display information about existing outlines. All of the caveats associated with optimizer hints apply equally to stored outlines. Under normal running the optimizer chooses the most suitable execution plan for the current circumstances. By using a stored outline, you may be forcing the optimizer to choose a substandard execution plan, so you should monitor the affects of your stored outlines over time to make sure this isn't happening.
Category	Admin
To find feature enablement	`SELECT OUTLINE_CATEGORY, OUTLINE_SID, USERS_EXECUTING` `  FROM V$SQL;`
Feature Usage	`SELECT count(*)` `  FROM V$SQL;`
Recommendation	When tuning an individual query, you usually start by looking at the execution plan of that query. The execution plan describes the sequence of operations, physical and logical, that SQL ServerTM will perform in order to fulfill the query and produce the desired resultset. The execution plan is produced by a database engine component called Query Optimizer during the optimization phase of query processing. It considers many different factors, such as the search predicates used in the query, the tables involved and their join conditions, the list of columns returned, and the presence of useful indexes that can serve as efficient access paths to the data. For complex queries, the number of all possible permutations can be huge, so the query optimizer does not evaluate all possibilities but instead tries to find a plan that is "good enough" for a given query. This is because finding a perfect plan may not always be possible; and even when it is possible, the cost of evaluating all the possibilities to find the perfect plan could easily outweigh any performance gains. From a DBA point of view, it is important to understand the process and its limitations. There are several ways to retrieve an execution plan for a query: • Management Studio provides Display Actual Execution Plan and Display Estimated Execution Plan features, which present the plan in a graphical way. These features offer the most suitable solution for direct examination and are by far the most often-used approach to display and analyze execution plans. (In this article, I will use graphical plans generated in this way to illustrate my examples.)

	• Various SET options, such as SHOWPLAN_XML and SHOWPLAN_ALL, return the execution plan as either an XML document describing the plan using a special schema or a rowset with textual description of each of the operations in the execution plan. • SQL Server Profiler event classes, such as Showplan XML, allow you to gather execution plans of statements collected by a trace.

Migration Approach	There is no direct migration approach available. You could utilize Database Engine Tuning Advisor in SQL Server.
	
	Initialize the Database Engine Tuning Advisor:
	On first use, a user who is member of the sysadmin fixed server role must initialize the Database Engine Tuning Advisor. This is because several system tables must be created in the msdb database to support tuning operations. Initialization also enables users that are members of the db_owner fixed database role to tune workloads on tables in databases that they own.
	
	A workload is a set of Transact-SQL statements that execute against a database or databases that you want to tune. Database Engine Tuning Advisor analyzes these workloads to recommend indexes or partitioning strategies that will improve your server's query performance.
	
	The dta utility provides a command prompt executable file that you can use to tune databases. It enables you to use Database Engine Tuning Advisor functionality in batch files and scripts. The dta utility takes plan cache entries, trace files, trace tables, and Transact-SQL scripts as workloads. It also takes XML input that conforms to the Database Engine Tuning Advisor XML schema
	
	To tune a database by using the plan cache:
	1. Specify the -ip option. The top 1,000 plan cache events for the selected databases are analyzed.
	`dta -E -D DatabaseName -ip -s SessionName`
	2. To modify the number of events to use for analysis, specify the –n option. The following example increases the number of cache entries to 2,000.
	`dta -E -D DatabaseName -ip –n 2000-s SessionName1`
	3. To analyze events for all databases in the instance, specify the -ipf option.
	`dta -E -D DatabaseName -ip –ipf –n 2000 -s SessionName2`

References	https://technet.microsoft.com/en-us/library/2007.11.sqlquery.aspx https://oracle-base.com/articles/misc/outlines

Online index rebuilds

Feature ID	80
Feature	Online index rebuild
Description	Altering existing index When you rebuild an index, you use an existing index as the data source. Creating an index in this manner enables you to change storage characteristics or move to a new tablespace. Rebuilding an index based on an existing data source removes intra-block fragmentation. Rebuilding online enables you to update base tables at the same time that you are rebuilding. Creating New Index You can create and rebuild indexes online. This enables you to update base tables at the same time you are building or rebuilding indexes on that table. You can perform DML operations while the index build is taking place, but DDL operations are not allowed. Parallel execution is not supported when creating or rebuilding an index online.
Category	SQL
Feature Usage	`SELECT * FROM USER_INDEXES WHERE INDEX_TYPE='NORMAL';`
Recommendation	The same feature available for SQL Server. The online index feature provides a powerful way to perform maintenance operations such as rebuilding or creating indexes in a production system without sacrificing DML concurrency. Users are not blocked from querying and updating the underlying table during the index operation. The SQL Server Database Engine automatically maintains indexes whenever insert, update, or delete operations are made to the underlying data. Over time these modifications can cause the information in the index to become scattered in the database (fragmented). Fragmentation exists when indexes have pages in which the logical ordering, based on the key value, does not match the physical ordering inside the data file. Heavily fragmented indexes can degrade query performance and cause your application to respond slowly. You can remedy index fragmentation by reorganizing or rebuilding an index. For partitioned indexes built on a partition scheme, you can use either of these methods on a complete index or a single partition of an index. Rebuilding an index drops and re-creates the index. This removes fragmentation, reclaims disk space by compacting the pages based on the specified or existing fill factor setting, and reorders the index rows in contiguous pages. When ALL is specified, all indexes on the table are dropped and rebuilt in a single transaction. In ONLINE mode the new index is built while the old index is accessible to reads and writes. any update on the old index will also get applied to the new index.

	Online index rebuilds are less intrusive when it comes to locking tables. Offline rebuilds cause heavy locking of tables which can cause significant blocking issues for things that are trying to access the database while the rebuild takes place. In OFFLINE mode the table is locked upfront for any read or write, and then the new index gets built from the old index, while holding a lock on the table. No read or write operation is permitted on the table while the index is being rebuilt. Only when the operation is done is the lock on the table released and reads and writes are allowed again. OFFLINE index rebuild is faster than ONLINE rebuild.

Migration Approach	All the index rebuild queries needs to be written manually using T-SQL.
	
	'Reorganize' and 'Rebuild' are two different operations that each reduce fragmentation in an index.
	
	Rebuild: An index 'rebuild' creates a fresh, sparkling new structure for the index. If the index is disabled, rebuilding brings it back to life. You can apply a new fillfactor when you rebuild an index. If you cancel a rebuild operation midway, it must roll back (and if it's being done offline, that can take a while).
	
	Reorganize: This option is more lightweight. It runs through the leaf level of the index, and as it goes it fixes physical ordering of pages and also compacts pages to apply any previously set fillfactor settings. This operation is always online, and if you cancel it then it's able to just stop where it is (it doesn't have a giant operation to rollback).
	
	The syntax for rebuilding indexes is very simple, we just add the "WITH ONLINE=ON" clause to the ALTER INDEX command.
	`ALTER INDEX [IX_Test] ON [dbo].[Test] REBUILD WITH (ONLINE = ON);`
	
	T-SQL to reorganize all indexes in a table:
	`ALTER INDEX ALL ON [dbo].[Test] REBUILD WITH (ONLINE = ON);`

References	https://docs.oracle.com/cd/B28359_01/server.111/b28310/indexes004.htm#ADMIN12318 https://msdn.microsoft.com/en-us/library/ms188388.aspx (ALTER INDEX) https://technet.microsoft.com/en-us/library/ms189858.aspx (Reorganize and Rebuild Indexes) https://www.brentozar.com/archive/2013/09/index-maintenance-sql-server-rebuild-reorganize/

Parallel DML

Feature ID	85
Feature	Parallel DML
Description	A DML statement can be parallelized only if you have explicitly enabled parallel DML in the session or in the SQL statement. To enable parallel DML mode in a specific SQL statement, include the ENABLE_PARALLEL_DML SQL hint.
Category	SQL
Recommendation	In SQL server parallel processing are determined by the Server automatically, however parallelism settings available to have some control on parallel executions. A lot of calculations are required to determine whether parallel processing should be used. Generally, SQL Server processes queries in parallel in the following cases: • When the number of CPUs is greater than the number of active connections. • When the estimated cost for the serial execution of a query is higher than the query plan threshold (The estimated cost refers to the elapsed time in seconds required to execute the query serially.) Certain types of statements cannot be processed in parallel unless they contain clauses, however. For example, UPDATE, INSERT, and DELETE are not normally processed in parallel even if the related query meets the criteria. But if the UPDATE or DELETE statements contain a WHERE clause, or an INSERT statement contains a SELECT clause, WHERE and SELECT can be executed in parallel. Changes are applied serially to the database in these cases. To configure parallel processing, simply do the following: 1. In the Server Properties dialog box, go to the Advanced page. 2. By default, the Max Degree Of Parallelism setting has a value of 0, which means that the maximum number of processors used for parallel processing is controlled automatically. Essentially, SQL Server uses the actual number of available processors, depending on the workload. To limit the number of processors used for parallel processing to a set amount (up to the maximum supported by SQL Server), change the Max Degree Of Parallelism setting to a value greater than 1. A value of 1 tells SQL Server not to use parallel processing. 3. Large, complex queries usually can benefit from parallel execution. However, SQL Server performs parallel processing only when the estimated number of seconds required to run a serial plan for the same query is higher than the value set in the cost threshold for parallelism. Set the cost estimate threshold using the Cost Threshold For Parallelism box on the Advanced page of the Server Properties dialog box. You can use any value from 0 through 32,767. On a single CPU, the cost threshold is ignored. 4. Click OK. These changes are applied immediately. You do not need to restart the server.
Migration Approach	SSMA wont handle parallel queries while migration. All the queries needs to be migrated manually using query hint

References	https://docs.oracle.com/database/121/VLDBG/GUID-5EB01FA8-030B-45BB-9B16-2D13881F6010.htm https://technet.microsoft.com/en-us/library/dd320292.aspx

Parallel Backup and recovery

Feature ID	88
Feature	Parallel Backup and recovery
Description	With RMAN's RESTORE and RECOVER commands, Oracle Database automatically makes parallel the following three stages of recovery: Restoring Datafiles When restoring datafiles, the number of channels you allocate in the RMAN recover script effectively sets the parallelism that RMAN uses. For example, if you allocate five channels, you can have up to five parallel streams restoring datafiles. Applying Incremental Backups Similarly, when you are applying incremental backups, the number of channels you allocate determines the potential parallelism. Applying Archived Redo Logs With RMAN, the application of archived redo logs is performed in parallel. Oracle Database automatically selects the optimum degree of parallelism based on available CPU resources.
Category	SQL
Recommendation	There is no exact similar feature available in SQL Server. Because SQL Server decides on Parallelism based on the execution. So there is no exact similar settings. The SQL Server backup and restore component provides an essential safeguard for protecting critical data stored in your SQL Server databases. To minimize the risk of catastrophic data loss, you need to back up your databases to preserve modifications to your data on a regular basis. A well-planned backup and restore strategy helps protect databases against data loss caused by a variety of failures. Test your strategy by restoring a set of backups and then recovering your database to prepare you to respond effectively to a disaster. In SQL server parallel processing are determined by the Server automatically, however parallelism settings available to have some control on parallel executions. A lot of calculations are required to determine whether parallel processing should be used. Generally, SQL Server processes queries in parallel in the following cases: • When the number of CPUs is greater than the number of active connections. • When the estimated cost for the serial execution of a query is higher than the query plan threshold (The estimated cost refers to the elapsed time in seconds required to execute the query serially.) Certain types of statements cannot be processed in parallel unless they contain clauses, however. For example, UPDATE, INSERT, and DELETE are not normally processed in parallel even if the related query meets the criteria. But if the UPDATE or DELETE statements contain a WHERE clause, or an INSERT

	statement contains a SELECT clause, WHERE and SELECT can be executed in parallel. Changes are applied serially to the database in these cases. To configure parallel processing, simply do the following: 1. In the Server Properties dialog box, go to the Advanced page. 2. By default, the Max Degree Of Parallelism setting has a value of 0, which means that the maximum number of processors used for parallel processing is controlled automatically. Essentially, SQL Server uses the actual number of available processors, depending on the workload. To limit the number of processors used for parallel processing to a set amount (up to the maximum supported by SQL Server), change the Max Degree Of Parallelism setting to a value greater than 1. A value of 1 tells SQL Server not to use parallel processing. 3. Large, complex queries usually can benefit from parallel execution. However, SQL Server performs parallel processing only when the estimated number of seconds required to run a serial plan for the same query is higher than the value set in the cost threshold for parallelism. Set the cost estimate threshold using the Cost Threshold For Parallelism box on the Advanced page of the Server Properties dialog box. You can use any value from 0 through 32,767. On a single CPU, the cost threshold is ignored. 4. Click OK. These changes are applied immediately. You do not need to restart the server.
Migration Approach	Stripped backup can be used while migrating. Since there is no direct approach available. This needs to be done manually.
References	https://docs.oracle.com/cd/B28359_01/rac.111/b28254/backup.htm#RACAD066

Parallel Index rebuilding

Feature ID	86
Feature	Parallel Index rebuilding
Description	The CREATE INDEX and ALTER INDEX ... REBUILD statements can be parallelized only by a PARALLEL clause or an ALTER SESSION FORCE PARALLEL DDLstatement. The ALTER INDEX ... REBUILD statement can be parallelized only for a nonpartitioned index, but ALTER INDEX ... REBUILD PARTITION can be parallelized by a PARALLEL clause or an ALTER SESSION FORCE PARALLEL DDL statement. The scan operation for ALTER INDEX ... REBUILD (nonpartitioned), ALTER INDEX ... REBUILD PARTITION, and CREATE INDEX has the same parallelism as the REBUILD or CREATE operation and uses the same DOP. If the DOP is not specified for REBUILD or CREATE, the default is the number of CPUs
Category	SQL
Recommendation	SQL Server uses the same algorithms to determine the degree of parallelism (the total number of separate threads to run) for index operations as it does for other queries. The maximum degree of parallelism for an index operation is subject to the max degree of parallelism server configuration option. You can override the max degree of parallelism value for individual index operations by setting the MAXDOP index option in the CREATE INDEX, ALTER INDEX, DROP INDEX, and ALTER TABLE statements. When the Database Engine builds an index execution plan, the number of parallel operations is set to the lowest value from among the following: • The number of microprocessors, or CPUs in the computer. • The number specified in the max degree of parallelism server configuration option. • The number of CPUs not already over a threshold of work performed for SQL Server threads. The main phases of a parallel index operation include the following: • A coordinating thread quickly and randomly scans the table to estimate the distribution of the index keys. The coordinating thread establishes the key boundaries that will create a number of key ranges equal to the degree of parallel operations, where each key range is estimated to cover similar numbers of rows. • The coordinating thread dispatches a number of threads equal to the degree of parallel operations and waits for these threads to complete their work. Each thread scans the base table using a filter that retrieves only rows with key values within the range assigned to the thread. Each thread builds an index structure for the rows in its key range. In the case of a partitioned index, each thread builds a specified number of partitions. Partitions are not shared among threads. • After all the parallel threads have completed, the coordinating thread connects the index subunits into a single index. This phase applies only to offline index operations.

	Individual CREATE TABLE or ALTER TABLE statements can have multiple constraints that require that an index be created. These multiple index creation operations are performed in series, although each individual index creation operation may be a parallel operation on a computer that has multiple CPUs.

Migration Approach	There is not direct migration approach available in SSMA tool. To achieve parallel index rebuild queries should be rewritten using query hint in SQL server.
	However there are some steps to set maximum degree of parallelism on an index.
	Using SQL Server management studios:
	1. In Object Explorer, click the plus sign to expand the database that contains the table on which you want to set max degree of parallelism for an index.
	2. Expand the **Tables** folder.
	3. Click the plus sign to expand the table on which you want to set max degree of parallelism for an index.
	4. Expand the **Indexes** folder.
	5. Right-click the index for which you want to set the max degree of parallelism and select **Properties**.
	6. Under **Select a page**, select **Options**.
	7. Select **Maximum degree of parallelism**, and then enter some value between 1 and 64.
	8. Click **OK.**
	Using Transact-SQL:
	To set max degree of parallelism on an existing index
	1. In **Object Explorer**, connect to an instance of Database Engine.
	2. On the Standard bar, click **New Query**.
	3. Copy and paste the following example into the query window and click **Execute**.
	
	`USE AdventureWorks2012;`
	`GO`
	`/*Alters the IX_ProductVendor_VendorID index on the Purchasing.ProductVendor table so that, if the server has eight or more processors, the Database Engine will limit the execution of the index operation to eight or fewer processors. */`
	`ALTER INDEX IX_ProductVendor_VendorID ON Purchasing.ProductVendor`
	`REBUILD WITH (MAXDOP=8);`
	`GO`
	Set max degree of parallelism on a new index
	1. In **Object Explorer**, connect to an instance of Database Engine.
	2. On the Standard bar, click **New Query**.
	3. Copy and paste the following example into the query window and click **Execute**.
	
	`USE AdventureWorks2012;`
	`GO`
	`CREATE INDEX IX_ProductVendor_NewVendorID`
	`ON Purchasing.ProductVendor (BusinessEntityID)`
	`WITH (MAXDOP=8);`
	`GO`

References	https://docs.oracle.com/cd/E11882_01/server.112/e25523/parallel003.htm#BEIBEFDD https://technet.microsoft.com/en-us/library/ms191292(v=sql.105).aspx https://msdn.microsoft.com/en-us/library/ms189329.aspx

Automatic SQL tuning

Feature ID	94
Feature	Automatic SQL tuning
Description	When SQL statements are executed by the Oracle database, the query optimizer is used to generate the execution plans of the SQL statements. The query optimizer operates in two modes: a normal mode and a tuning mode. In normal mode, the optimizer compiles the SQL and generates an execution plan. The normal mode of the optimizer generates a reasonable execution plan for the vast majority of SQL statements. Under normal mode, the optimizer operates with very strict time constraints, usually a fraction of a second, during which it must find a good execution plan. In tuning mode, the optimizer performs additional analysis to check whether the execution plan produced under normal mode can be improved further. The output of the query optimizer is not an execution plan, but a series of actions, along with their rationale and expected benefit for producing a significantly superior plan. When running in the tuning mode, the optimizer is referred to as the Automatic Tuning Optimizer.
Category	SQL

To find feature enablement	`SELECT window_name,TO_CHAR(window_next_time,'DD-MON-YY HH24:MI:SS') ,sql_tune_advisor, optimizer_stats, segment_advisor FROM dba_autotask_window_clients;`
	
	`SELECT client_name, status, consumer_group, window_group FROM dba_autotask_client ORDER BY client_name;`

Feature Usage	`SELECT window_name,TO_CHAR(window_next_time,'DD-MON-YY HH24:MI:SS') ,sql_tune_advisor, optimizer_stats, segment_advisor FROM dba_autotask_window_clients;`
Recommendation	SQL server execution plans are equivalent to query tuning in Oracle but it is not automatic as Oracle "Include Actual Execution Plan" option in Microsoft SQL Server Management Studio. It displays detailed information about the execution path taken by the server when running your query. (Note that this works best when there's data in the tables. Of course, without a good bit of test data, any optimization is purely theoretical anyway.) It basically gives three very important things: 1. It tells you which steps take the most processing time and what they're doing at that step.

	2. It tells you which steps carry the most data to the next step, including how many records, which helps identify places where you can be more specific about the data you want and exclude unnecessary records. 3. It gives you a ton of insight into the inner workings of SQL Server and what it does with your queries. This knowledge will help you optimize things a lot over time.
Migration Approach	This feature does not have migration approach, since query execution plan UI is a replacement for this feature
References	http://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm#i37659

Parallel shared pool

Feature ID	96
Feature	Parallel shared pool
Description	Parallel execution requires memory resources in addition to those required by serial SQL execution. Additional memory is used for communication and passing data between query server processes and the query coordinator. Oracle Database allocates memory for query server processes from the shared pool. Tune the shared pool as follows: • Allow for other clients of the shared pool, such as shared cursors and stored procedures. • Remember that larger values improve performance in multiuser systems, but smaller values use less memory. •
Category	SQL
To find feature enablement	`SELECT POOL, NAME, SUM(BYTES) FROM V$SGASTAT WHERE POOL LIKE '%pool%' GROUP BY ROLLUP (POOL, NAME);`
Feature Usage	`SELECT POOL, NAME, SUM(BYTES) FROM V$SGASTAT WHERE POOL LIKE 'shared pool' or POOL LIKE 'large pool' GROUP BY ROLLUP (POOL, NAME);`
Recommendation	There is no relevant configuration available in SQL Server.
Migration Approach	There is no migration available for this feature
References	https://docs.oracle.com/cd/E11882_01/server.112/e25523/parallel005.htm

Parallel buffers

Feature ID	97
Feature	Parallel Buffers
Description	Parallel execution requires memory resources in addition to those required by serial SQL execution. Additional memory is used for communication and passing data between query server processes and the query coordinator. • You can then monitor the number of buffers used by parallel execution and compare the shared pool PX msg pool to the current high water mark reported in output from the view V$PX_PROCESS_SYSSTAT. If you do not have enough memory available, error message 12853 occurs (insufficient memory for PX buffers: current stringK, max needed stringK). This is caused by having insufficient SGA memory available for PX buffers. You must reconfigure the SGA to have at least (MAX - CURRENT) bytes of additional memory.
Category	SQL
To find feature enablement	SELECT POOL, NAME, SUM(BYTES) FROM V$SGASTAT WHERE POOL LIKE '%pool%' GROUP BY ROLLUP (POOL, NAME);
Feature Usage	`SELECT POOL, NAME, SUM(BYTES) FROM V$SGASTAT WHERE POOL LIKE 'shared pool' or POOL LIKE 'large pool' GROUP BY ROLLUP (POOL, NAME);`
Recommendation	**Feature Description:** SQL Server provides parallel queries to optimize query execution and index operations for computers that have more than one microprocessor (CPU). Because SQL Server can perform a query or index operation in parallel by using several operating system threads, the operation can be completed quickly and efficiently. The SQL Server query optimizer does not use a parallel execution plan for a query if any one of the following conditions is true: • The serial execution cost of the query is not high enough to consider an alternative, parallel execution plan. • A serial execution plan is considered faster than any possible parallel execution plan for the particular query. • The query contains scalar or relational operators that cannot be run in parallel. Certain operators can cause a section of the query plan to run in serial mode, or the whole plan to run in serial mode.
Migration Approach	There is no migration available for this feature. However, SQL Server uses parallel query processing to optimize its performance. • During query optimization, SQL Server looks for queries or index operations that might benefit from parallel execution. For these queries, SQL Server inserts exchange operators into the query execution plan to prepare the query for parallel execution. • An exchange operator is an operator in a query execution plan that provides process management, data redistribution, and flow control.

	• The exchange operator includes the Distribute Streams, Repartition Streams, and Gather Streams logical operators as subtypes, one or more of which can appear in the Showplan output of a query plan for a parallel query. • After exchange operators are inserted, the result is a parallel-query execution plan. • A parallel-query execution plan can use more than one thread. A serial execution plan, used by a nonparallel query, uses only one thread for its execution. • The actual number of threads used by a parallel query is determined at query plan execution initialization and is determined by the complexity of the plan and the degree of parallelism. Degree of parallelism determines the maximum number of CPUs that are being used; it does not mean the number of threads that are being used. • The degree of parallelism value is set at the server level and can be modified by using the **sp_configure** system stored procedure. You can override this value for individual query or index statements by specifying the MAXDOP query hint or MAXDOP index option.
References	http://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm#i37659

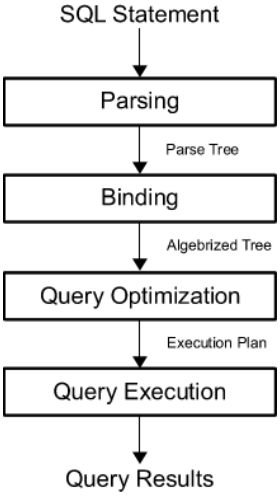
Degree of Parallelism

Feature ID	102
Feature	Degree of parallelism
Description	The number of parallel execution servers associated with a single operation is known as the degree of parallelism (DOP). Parallel execution is designed to effectively use multiple CPUs. Oracle Database parallel execution framework enables you to either explicitly choose a specific degree of parallelism or to rely on Oracle Database to automatically control it. This section contains the following topics: • Manually Specifying the Degree of Parallelism • Automatic Degree of Parallelism • Controlling Automatic Degree of Parallelism • In-Memory Parallel Execution • Adaptive Parallelism • Controlling Automatic DOP_ Parallel Statement Queuing_ and In-Memory Parallel Execution
Category	SQL
Recommendation	SQL Server automatically detects the best degree of parallelism for each instance of a parallel query execution or index data definition language (DDL) operation. It does this based on the following criteria: 1. Whether SQL Server is running on a computer that has more than one microprocessor or CPU, such as a symmetric multiprocessing computer (SMP). Only computers that have more than one CPU can use parallel queries. 2. Whether sufficient threads are available. Each query or index operation requires a certain number of threads to execute. Executing a parallel plan requires more threads than a serial plan , and the number of required threads increases with the degree of parallelism. When the thread requirement of the parallel plan for a specific degree of parallelism cannot be satisfied, the Database Engine decreases the degree of parallelism automatically or completely abandons the parallel plan in the specified workload context. It then executes the serial plan (one thread). 3. The type of query or index operation executed. Index operations that create or rebuild an index, or drop a clustered index and queries that use CPU cycles heavily are the best candidates for a parallel plan. For example, joins of large tables, large aggregations, and sorting of large result sets are good candidates. Simple queries, frequently found in transaction processing applications, find the additional coordination required to execute a query in parallel outweigh the potential performance boost. To distinguish between queries that benefit from parallelism and those that do not benefit, The Database Engine compares the estimated cost of executing the query or index operation with the cost threshold for parallelism value. Although not recommended, users can change the default value of 5 using sp_configure. 4. Whether there are a sufficient number of rows to process. If the query optimizer determines that the number of rows is too low, it does not introduce exchange operators to distribute the rows. Consequently, the operators are executed serially.

	Executing the operators in a serial plan avoids scenarios when the startup, distribution, and coordination costs exceed the gains achieved by parallel operator execution. 5. Whether current distribution statistics are available. If the highest degree of parallelism is not possible, lower degrees are considered before the parallel plan is abandoned. For example, when you create a clustered index on a view, distribution statistics cannot be evaluated, because the clustered index does not yet exist. In this case, the Database Engine cannot provide the highest degree of parallelism for the index operation. However, some operators, such as sorting and scanning, can still benefit from parallel execution.
Migration Approach	While migrating we have evaluate the parallelism defined in existing system. The same level of parallelism can be achieved by server configuration.
References	https://docs.oracle.com/database/121/VLDBG/GUID-68ED8F5E-DD97-4762-985C-4C3AF85F9629.htm#VLDBG0102 https://technet.microsoft.com/en-us/library/ms188611(v=sql.105).aspx

Advanced Rewrite

Feature ID	102
Feature	Advanced rewrite
Description	The optimizer uses a number of different methods to rewrite a query. The first step in determining whether query rewrite is possible is to see if the query satisfies the following prerequisites: • Joins present in the materialized view are present in the SQL. • There is sufficient data in the materialized view(s) to answer the query. After that, it must determine how it will rewrite the query. The simplest case occurs when the result stored in a materialized view exactly matches what is requested by a query. The optimizer makes this type of determination by comparing the text of the query with the text of the materialized view definition. This text match method is most straightforward but the number of queries eligible for this type of query rewrite is minimal. When the text comparison test fails, the optimizer performs a series of generalized checks based on the joins, selections, grouping, aggregates, and column data fetched. This is accomplished by individually comparing various clauses (SELECT, FROM, WHERE, HAVING, or GROUP BY) of a query with those of a materialized view. This section discusses the optimizer in more detail, as well as the following types of query rewrite: • Text Match Rewrite • General Query Rewrite Methods
Category	SQL
Feature Usage	SELECT owner, name FROM dba_rewrite_equivalences;
Recommendation	In SQL server, there is not automatic query rewrite available. The query optimizer gives information regarding the issues related to queries. But there is no automatic rewrite. The job of the Query Optimizer is to create and assess as many candidate execution plans as possible, within certain criteria, in order to arrive at the best possible plan. We define the search space for a given query as the set of all the possible execution plans for that query, and any possible plan in this search space returns the same results. Theoretically, to find the optimum execution plan for a query, a cost-based query optimizer should generate all possible execution plans that exist in that search space and correctly estimate the cost of each plan. However, some complex queries may have thousands or even millions of possible execution plans and, while the SQL Server Query Optimizer can typically consider many candidate execution plans, it cannot perform an exhaustive search of all the possible plans for every query. If it did, then the time taken to assess all the plans would be unacceptably long, and could start to have a major impact on the overall query execution time.
Migration Approach	There is no direct migration approach available in SSMA tool. In SQL Server, query optimization is facilitated by query optimizer. Query optimizers does not rewrite queries.

	The SQL Server Query Optimizer is a cost-based optimizer. It analyzes several candidate execution plans for a given query, estimates the cost of each of these plans and selects the plan with the lowest cost of the choices considered. At the core of the SQL Server Database Engine are two major components: The Storage Engine and the Query Processor, also called the Relational Engine. The Storage Engine is responsible for reading data between the disk and memory in a manner that optimizes concurrency while maintaining data integrity. The Query Processor, as the name suggests, accepts all queries submitted to SQL Server, devises a plan for their optimal execution, and then executes the plan and delivers the required results. for each query it receives, the first job of the query processor is to devise a plan, as quickly as possible, and chooses the best possible way to execute said query (or, at the very least, an efficient way). Its next task is to execute the query per that plan. Each of these tasks is delegated to a separate component within the query processor; the Query Optimizer devises the plan and then passes it along to the Execution Engine, which will execute the plan and get the results from the database. To arrive at what it believes to be the best plan for executing a query, the Query Processor performs a number of different steps; the entire query processing process is shown on figure:  To show an XML plan you can use the following commands: SET SHOWPLAN_XML ON GO SELECT DISTINCT(City) FROM Person.Address GO SET SHOWPLAN_XML OFF Displaying the execution plans for currently-running queries: SELECT query_plan FROM sys.dm_exec_requests CROSS APPLY sys.dm_exec_query_plan(plan_handle) WHERE session_id = 135
References	https://docs.oracle.com/database/121/VLDBG/GUID-68ED8F5E-DD97-4762-985C-4C3AF85F9629.htm#VLDBG0102

	https://technet.microsoft.com/en-us/library/ms188611(v=sql.105).aspx https://www.simple-talk.com/sql/sql-training/the-sql-server-query-optimizer/

Feedback and suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Data Migration Jumpstart Team (askdmjfordmtools@microsoft.com). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the Azure Database Migration Guide.