Delivering a Relational Data Warehouse

Week 3 – Optimizing a Data Warehouse for Scale and Performance

Module 09

Improving Operational Performance

Microsoft

Module Outline

09 | Improving Operational Performance

Topic
Data Warehouse Loading
Data Warehouse Maintenance
Demo: Optimizing the Loading of the Data Warehouse
Lab: Optimizing a Data Warehouse for Scale and Performance

Microsoft

Module Outline

09 | Improving Operational Performance

Topic
Data Warehouse Loading
Data Warehouse Maintenance
Demo: Optimizing the Loading of the Data Warehouse
Lab: Optimizing a Data Warehouse for Scale and Performance

Data Warehouse Loading

- Data warehouse loading is concerned with:
  - Loading tables
  - Maintaining indexes
  - Updating statistics
  - Processing related data assets

Data Warehouse Loading
Loading Tables

- Often, the objective is to efficiently load data into data warehouse table as quickly as possible
- SQL Server supports various techniques and methodologies to performance tune and optimize the loading processes
- Consideration needs to be given to:
  - Logging
  - Locking
  - Degree of parallelism

Data Warehouse Loading
Loading Tables ► Logging

- To support high-volume data loading scenarios, SQL Server implements minimally logged operations
  - Minimally logged operations keep track of extent allocations and metadata changes only
- This results in:
  - Less information tracked in the transaction log
  - Faster loading, involving fewer writes to the transaction log
- Minimally logged operations are available only if the database is in bulk-logged or simple recovery mode

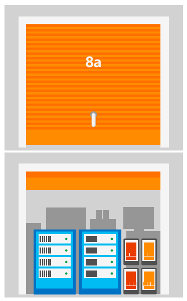
Data Warehouse Loading
Loading Tables ► Locking

- SQL Server will apply locks to tables during loading
- To minimize the cost of locking, SQL Server locks resources automatically at a level appropriate to the task
  - Locking at a smaller granularity, increases concurrency, but has a higher overhead
  - Locking at a larger granularity is expensive in terms of concurrency, but has a lower overhead because fewer locks are being maintained
- Different loading techniques manage locking differently

Data Warehouse Loading

Loading Tables ► Degree of Parallelism

- Strive to load as many tables in parallel as there are CPUs, if there is no I/O bottleneck
- If I/O bandwidth is limited, consider serializing all—or part—of the loading process



Data Warehouse Loading

Loading Tables ► Methods

- SQL Server supports several loading methods:
  - Integration Services (SSIS) data destinations
  - BCP (Bulk Copy Program) utility
  - BULK INSERT
  - INSERT … SELECT
  - SELECT INTO
  - MERGE

T-SQL

Data Warehouse Loading

Loading Tables ► BCP Utility

- The BCP Utility can be used to both extract from, and import text files into, SQL Server
  - Additionally, the utility can export data from SQL Server tables or queries into text files
  - The BATCHSIZE (-b) switch allows specifying the number of rows per batch of imported data, with each batch imported and logged as a separate transaction

Data Warehouse Loading

Loading Tables ► BULK INSERT

- The BULK INSERT command is the in-process method for bringing data from a text file into SQL Server—on the same server
- It is invoked from T-SQL, which makes it ideally suited for use in stored procedures or T-SQL based ETL
- It has the same abilities as the BCP Utility, except that it cannot export data

Data Warehouse Loading

Loading Tables ► INSERT … SELECT

- The INSERT … SELECT command can perform minimally logged insert operations, allowing T-SQL based INSERT statements
- There are some limitations:
  - None of the bulk load parameters, like commit size, check constraints, and trigger firing can be used with this method
  - The target table is locked with an exclusive (X) lock, while the other bulk load methods issue the bulk update (BU) lock

Data Warehouse Loading

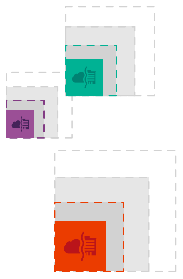
Loading Tables ► SELECT INTO

- The SELECT INTO statement produces a new table based on the result of a SELECT statement
  - The newly created rows are minimally logged, providing a very fast way to load new data
  - There is no way to control BATCHSIZE for this statement
  - The target table must reside on the default filegroup

Data Warehouse Loading

Loading Tables ► MERGE

- The MERGE statement performs insert, update, and/or delete operations on a target table based on the results of a join with a source table
- This is often useful for loading dimension tables
- Consider two statements:
  1. Insert new, and overwrite Type 1 changes
  2. Handle Type 2 changes



Data Warehouse Loading

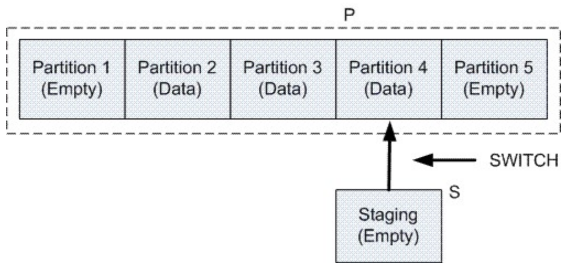
Loading Tables ► MERGE ► Syntax

```
[ WITH <common_table_expression> [,...n] ]
MERGE
    [ TOP ( expression ) [ PERCENT ] ]
    [ INTO ] <target_table> [ WITH ( <merge_hint> ) ] [ [ AS ] table_alias ]
    USING <table_source>
    ON <merge_search_condition>
    [ WHEN MATCHED [ AND <clause_search_condition> ]
        THEN <merge_matched> ] [ ...n ]
    [ WHEN NOT MATCHED [ BY TARGET ] [ AND <clause_search_condition> ]
        THEN <merge_not_matched> ]
    [ WHEN NOT MATCHED BY SOURCE [ AND <clause_search_condition> ]
        THEN <merge_matched> ] [ ...n ]
    [ <output_clause> ]
    [ OPTION ( <query_hint> [ ,...n ] ) ]
;
```

Data Warehouse Loading

Loading Tables ► Partitioned Tables

- Use partition switching to efficiently load new data
- Always keep empty partitions at both ends of the partition range to guarantee a partition split
- Use partition-aligned indexes in switching operations



Data Warehouse Loading

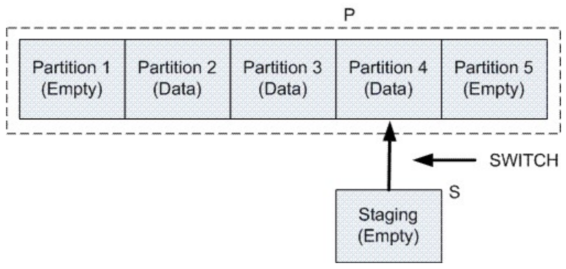
Loading Tables ► Partitioned Tables ► Methodology

1. Create a staging table for the new file group (just a regular table, on the filegroup you want)
2. If the partitioned table has a primary key, add it also to the staging table
3. Create any non-clustered (aligned) indexes on the staging table
4. Create CHECK constraints using WITH CHECK on the staging table that verify the data is in the correct range
5. Alter the partition scheme with NEXT USED
6. Alter the partition function to split the range
7. Switch in the staging table

Data Warehouse Loading

Loading Tables ► Partitioned Tables (Continued)

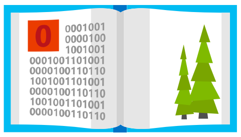
- Note that it is faster to load a complete partition at a time
  - Daily partitions for daily loads may be an attractive option
  - However, keep in mind that a table can have a maximum of 15,000 partitions

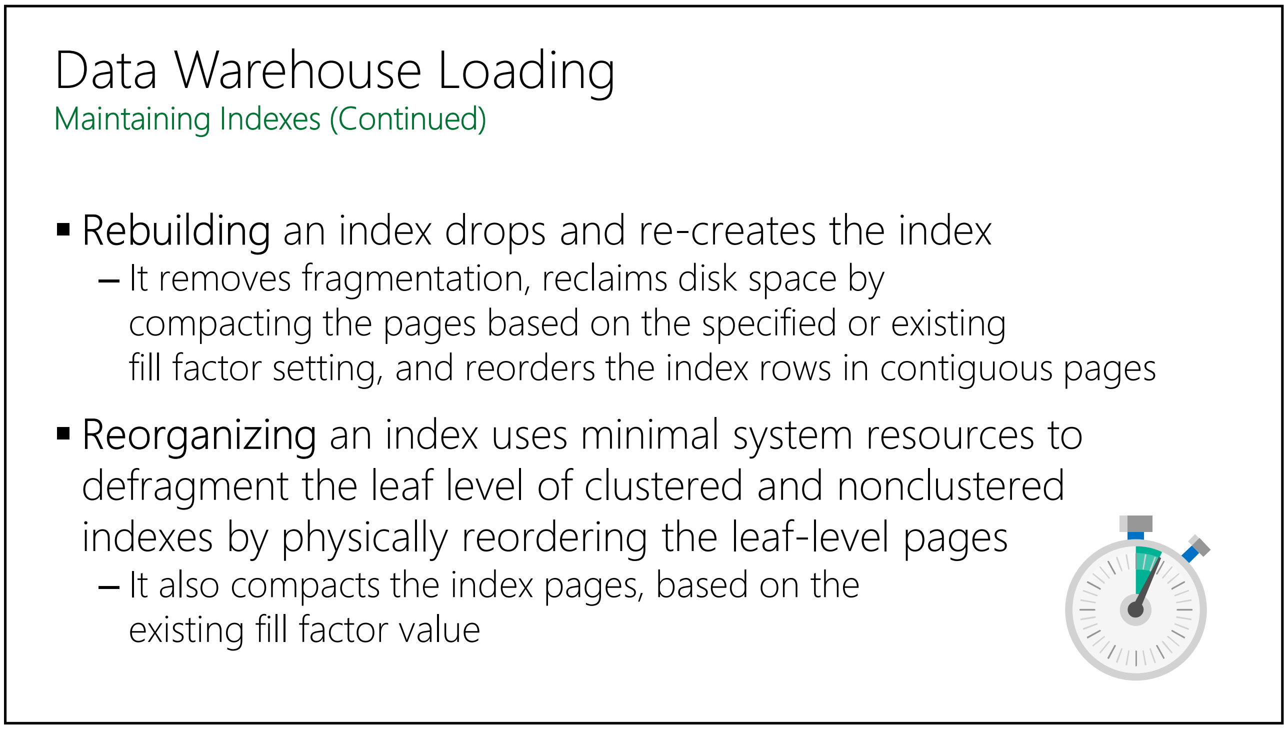


Data Warehouse Loading

Maintaining Indexes

- SQL Server automatically synchronizes indexes whenever table insert, update, or delete operations are made
- Over time these modifications can cause the information in the index to become scattered in the database (fragmented)
- Heavily fragmented indexes can degrade query performance
- To remedy index fragmentation, reorganize or rebuild indexes

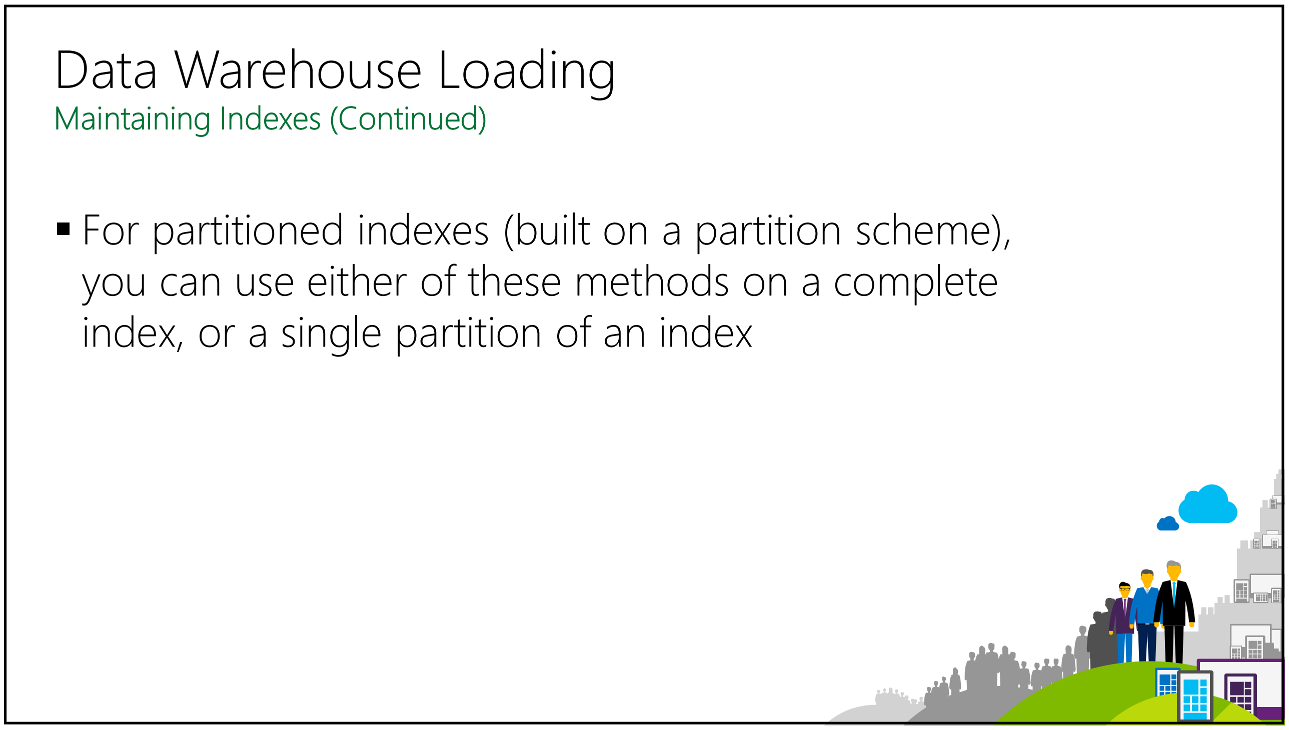




Data Warehouse Loading

Maintaining Indexes (Continued)

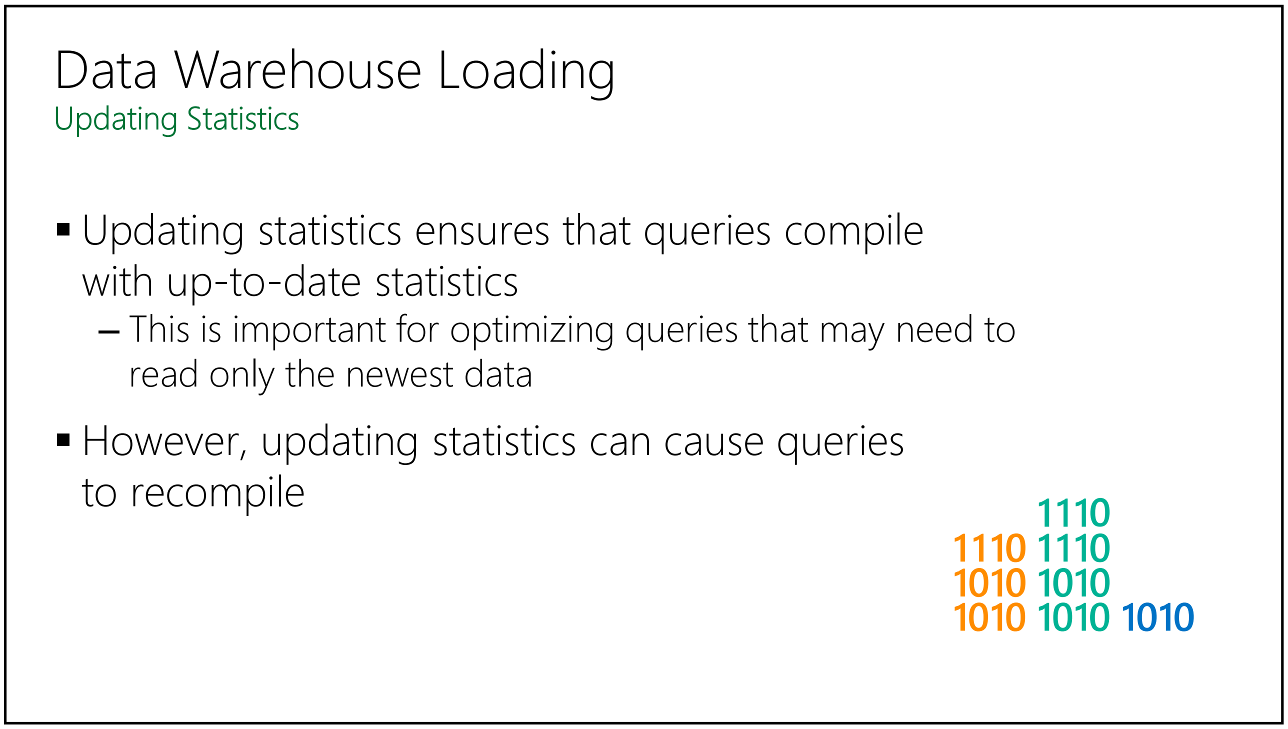
- Rebuilding an index drops and re-creates the index
  - It removes fragmentation, reclaims disk space by compacting the pages based on the specified or existing fill factor setting, and reorders the index rows in contiguous pages
- Reorganizing an index uses minimal system resources to defragment the leaf level of clustered and nonclustered indexes by physically reordering the leaf-level pages
  - It also compacts the index pages, based on the existing fill factor value



Data Warehouse Loading

Maintaining Indexes (Continued)

- 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

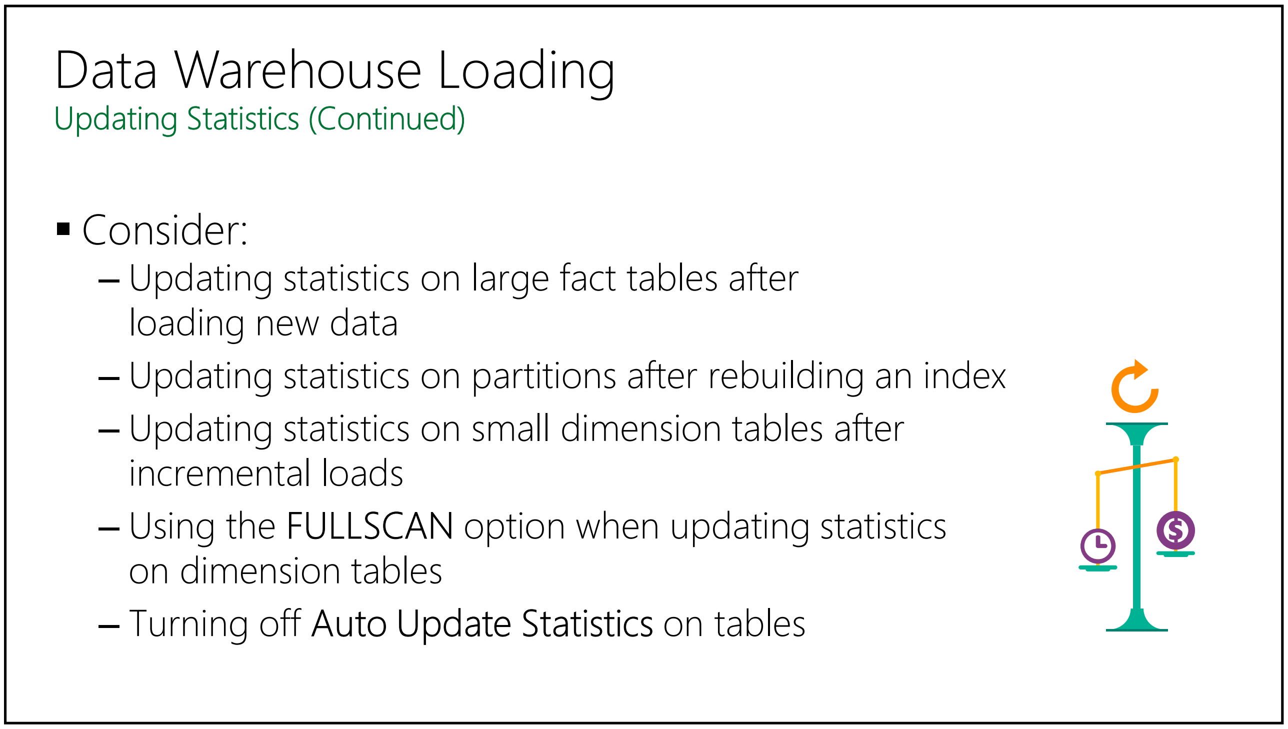


Data Warehouse Loading

Updating Statistics

- Updating statistics ensures that queries compile with up-to-date statistics
  - This is important for optimizing queries that may need to read only the newest data
- However, updating statistics can cause queries to recompile

1110
1110 1110
1010 1010
1010 1010 1010



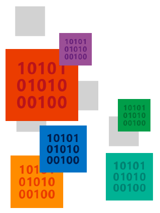
Data Warehouse Loading

Updating Statistics (Continued)

- Consider:
  - Updating statistics on large fact tables after loading new data
  - Updating statistics on partitions after rebuilding an index
  - Updating statistics on small dimension tables after incremental loads
  - Using the FULLSCAN option when updating statistics on dimension tables
  - Turning off Auto Update Statistics on tables

Data Warehouse Loading
Processing Related Data Assets

- Commonly, the data warehouse contains data models
- Analysis Services (SSAS) data models—multidimensional and tabular—can define partitions to manage storage and processing
- Ideally, the SSAS partitions should be aligned with the fact table partitions
  - Only process SSAS partitions that correspond to loaded fact table partitions



Microsoft

Module Outline
09 | Improving Operational Performance

Topic
Data Warehouse Loading
Data Warehouse Maintenance
Demo: Optimizing the Loading of the Data Warehouse
Lab: Optimizing a Data Warehouse for Scale and Performance

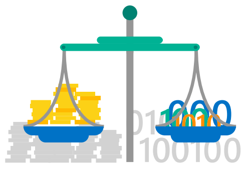
Data Warehouse Maintenance

- Data warehouse maintenance is concerned with:
  - Reviewing indexes
  - Archiving data
  - Backups

Data Warehouse Maintenance

Reviewing Indexes

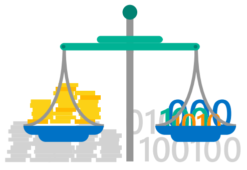
- Periodically, review indexes to ensure that an optimal set of indexes are applied
  - Review usage
  - Respond to slow-running queries
  - Identify new index requirements
  - Considering dropping indexes rarely used



Data Warehouse Maintenance

Reviewing Indexes ► Database Engine Tuning Advisor

- The **Database Engine Tuning Advisor** can help to select and create an optimal set of indexes, indexed views, and partitions
- The advisors accepts these workloads:
  - Plan cache entries
  - Profiler trace files, and
  - T-SQL scripts



Data Warehouse Maintenance
Archiving Data

- Archiving data may not be a requirement or a priority
- If large volumes of data need to be deleted, use partition switching whenever possible

Data Warehouse Maintenance
Archiving Data (Continued)

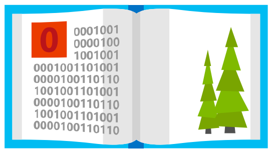
- For non-partitioned tables:
  - Avoid DELETE FROM … WHERE … ; due to excessive locking and logging
- Alternatively:
  - 'Trickle' the record deletions by using the following statements repeatedly in a loop
    - DELETE TOP (1000) … ;
    - COMMIT;

Data Warehouse Maintenance
Archiving Data (Continued)

- It is usually faster to:
  - Insert the records to retain into a new non-indexed table
  - Create index(es) on the new table
  - Delete the original table
  - Rename the new table to replace the original
- Yet another alternative is to update the rows to mark them as logically deleted, and then physically delete them later at a non-critical time

Data Warehouse Maintenance
Backups

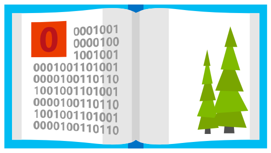
- Implement an efficient backup strategy which is designed for business continuity
- Backing up the entire data warehouse may take significant amount of time and storage space
- Do not assume that the data warehouse can be reconstructed from the source system(s)



Data Warehouse Maintenance

Backups (Continued)

- Consider reducing the volume of data to backup regularly:
  - Filegroups for historical partitions can be marked as read-only
  - Perform a filegroup backup once the filegroup is set to read-only
  - Perform regular backups only on the read-write filegroups
  - Use backup compression, taking into consideration the performance trade-off
- Test backups by periodically restoring them



Data Warehouse Maintenance

Backups ► Azure SQL Databases

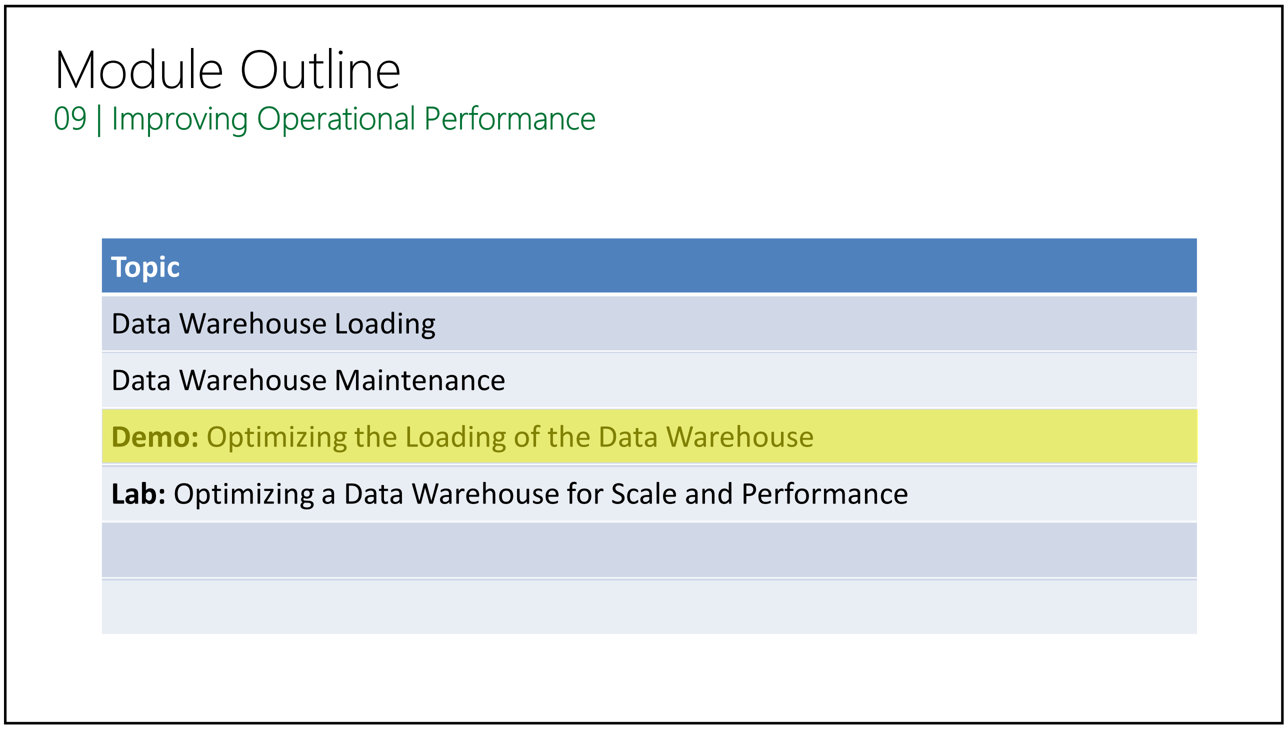
- Backing up and restoring data is different in Azure SQL Database to on-premises SQL Server
  - Azure SQL Databases are replicated to different servers and/or regions for disaster recovery, but are not technically backed up for the purpose of recovering accidental data loss or changes

Data Warehouse Maintenance

Backups ► Azure SQL Databases (Continued)

- Azure SQL Database recovery supports:
  - Point In Time Restore (available for all service tiers, up to 35 days' of history for Premium)
  - Geo-Restore (restores from the geo-replicated backup copy and therefore is resilient to the storage outages in the primary region), and
  - Active Geo-Replication (up to four readable secondaries on servers in different regions)

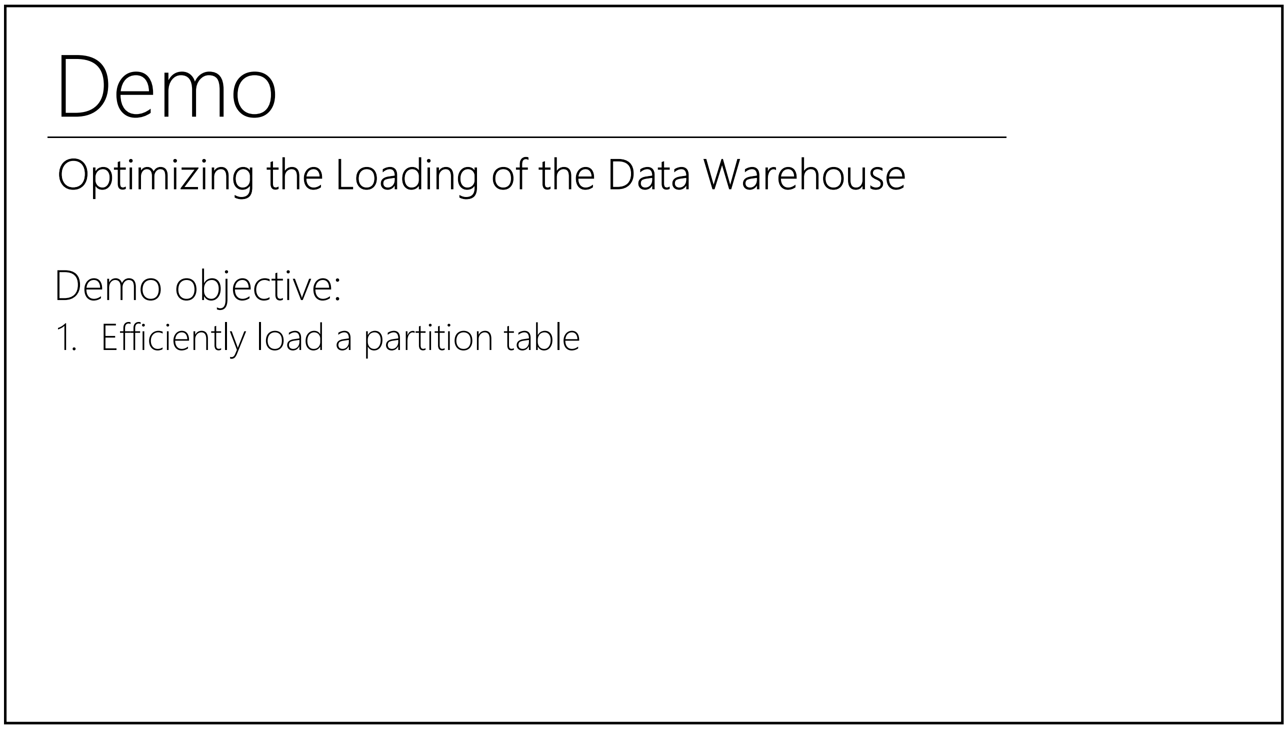
Microsoft



Module Outline

09 | Improving Operational Performance

Topic
Data Warehouse Loading
Data Warehouse Maintenance
Demo: Optimizing the Loading of the Data Warehouse
Lab: Optimizing a Data Warehouse for Scale and Performance



Demo

Optimizing the Loading of the Data Warehouse

Demo objective:

1. Efficiently load a partition table

Microsoft

Module Outline

09 | Improving Operational Performance

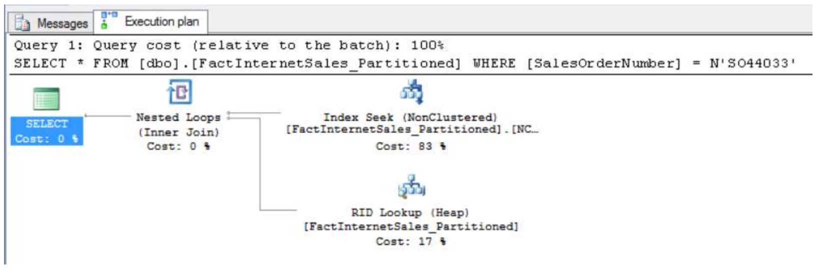
Topic
Data Warehouse Loading
Data Warehouse Maintenance
Demo: Optimizing the Loading of the Data Warehouse
Lab: Optimizing a Data Warehouse for Scale and Performance

Lab

02 | Optimizing a Data Warehouse for Scale and Performance

Lab exercises:

1. Optimizing the snowflake product table design
2. Optimizing the FactInternetSales table design



```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [dbo].[FactInternetSales_Partitioned] WHERE [SalesOrderNumber] = N'SO44033'
```

Lab

02 | Optimizing a Data Warehouse for Scale and Performance

Tips:

- Be sure to read instructions carefully, especially when executing scripts
- If you did not successfully complete last week's lab, there is a lab setup shortcut, but you will still need to provision and setup an Azure VM
- The lab setup shortcut can be used to reset and try again

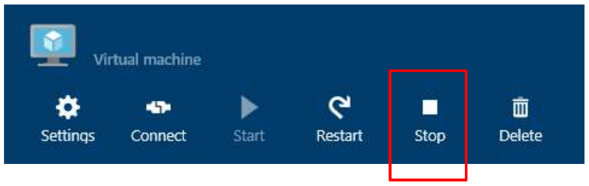
Lab

02 | Optimizing a Data Warehouse for Scale and Performance

Reminder

When you have completed the lab, remember to stop your VM

You are charged when the VM status is Running, but you are not charged when the VM status is Stopped (Deallocated)



Microsoft