

## Delivering a Relational Data Warehouse

Week 3 – Optimizing a Data Warehouse for Scale and Performance

Module 07

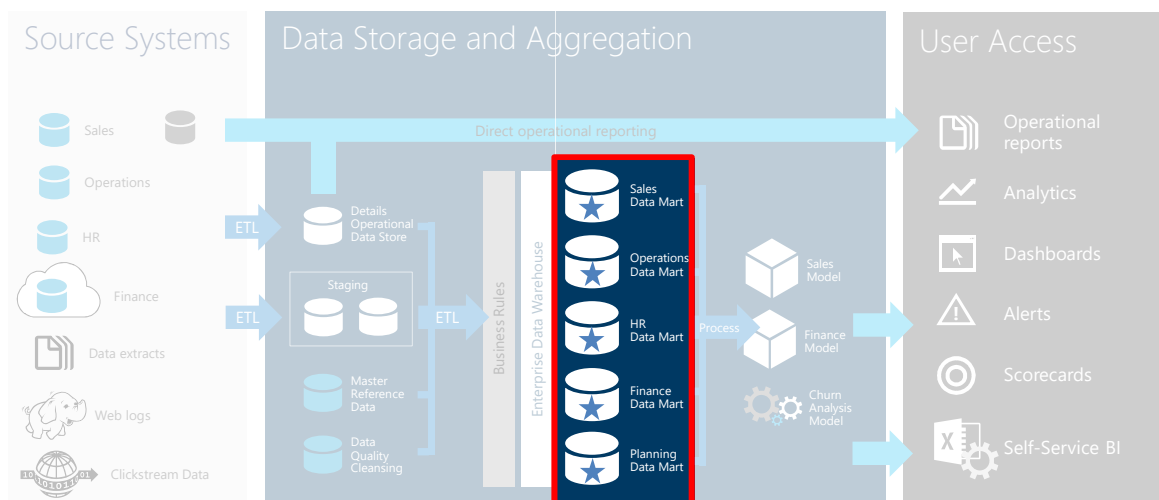
### Designing a Physical Database Architecture



CR1

## Week Outline

### 3 | Optimizing a Data Warehouse for Scale and Performance



## Slide 2

---

**CR1** Just to be picky here, I think the red outline should enclose the label "Enterprise Data Warehouse" here to make clear the scope is more than the marts.

Chris Randall, 6/7/2016

# Module Outline

## 07 | Designing a Physical Database Architecture

Topic	
▶	Physical Table Designs
▶	Table Partitioning
▶	<b>Demo:</b> Developing Optimized Table Designs



# Microsoft

©2016 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

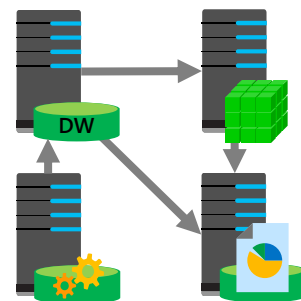
# Module Outline

## 07 | Designing a Physical Database Architecture

Topic
Physical Table Designs
Table Partitioning
<b>Demo:</b> Developing Optimized Table Designs

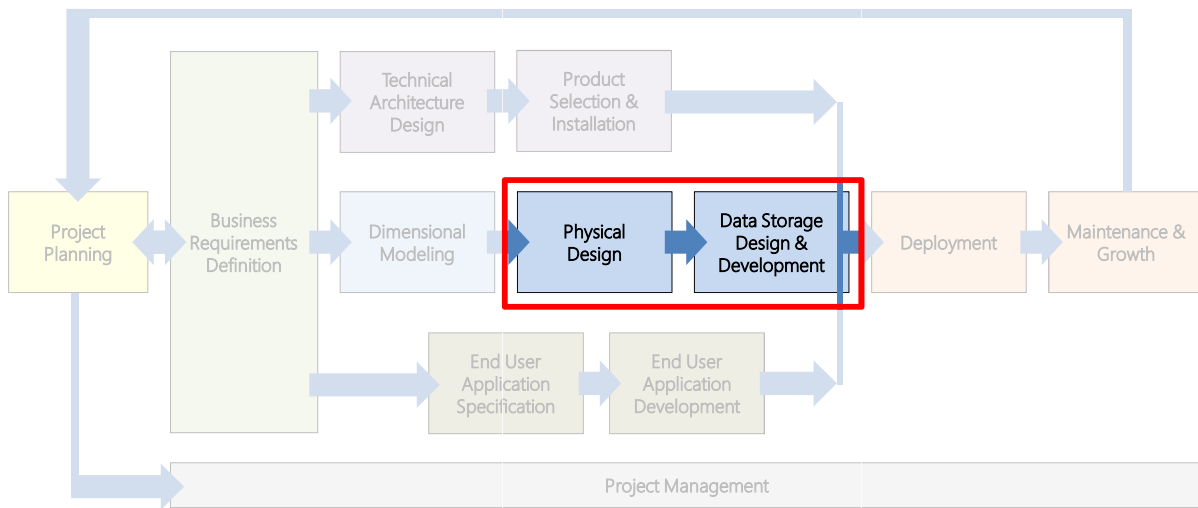
## Physical Table Designs

- Developing a large scale relational data warehouse is often a complex task
- As a relational data warehouse developer, you will need to design and develop databases and supporting schemas to:
  - Store large volumes of data
  - Enable efficient:
    - Data loading
    - Data querying
    - Maintenance



## Physical Table Designs

### Kimball Business Dimensional Lifecycle



Source: The Data Warehouse Lifecycle Toolkit

## Physical Table Designs

### Design Considerations

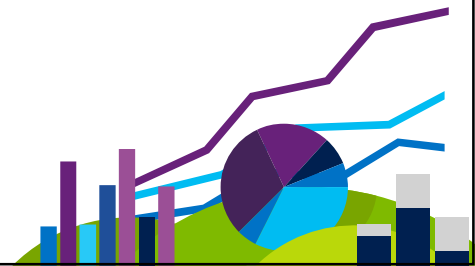
- Considerations must be given to the technical architecture and product selection
  - Hardware
  - Product selection
  - Number, and placement, of database servers
  - Database server subsystems:
    - Processors
    - Memory
    - Disk
    - Networking



## Physical Table Designs

### Design Considerations (Continue)

- The theory presented this week is largely applicable to SQL Server database design, whether on-premises, IaaS or PaaS
  - Any differences in design approaches or functionality will be called out



## Physical Table Designs

### Table Design Fundamentals

- When creating tables, give consideration to the role of the table: staging, dimension or fact
- In general:
  - Use the smallest possible column data types
  - Avoid adding columns that allow storing NULL
  - Add foreign key constraints, but do not necessarily enable them
  - Apply a balanced index design
- Strive to store data as densely as possible, and spread the data across physical storage devices to improve I/O throughput

## Physical Table Designs

### Table Design Fundamentals ► Staging Tables

- **Staging Tables** are often truncated and reloaded with each ETL process
  - To allow TRUNCATE TABLE, do not define foreign keys
- Consider table designs that map closely to the structure of the data source(s)
- Consider also defining column data types that are resilient to poor data quality
  - For example, define data and numeric columns as VARCHAR if you suspect invalid values may be sourced

## Physical Table Designs

### Table Design Fundamentals ► Dimension Tables

- **Dimensions Tables** are loaded incrementally
  - Resetting surrogate key values would otherwise require reloading all related fact tables
- Column types:
  - Surrogate key
  - Business key(s)
  - Dimension attributes
  - Slowly Changing Dimension (SCD) tracking
  - ETL tracking, and lineage

## Physical Table Designs

### Table Design Fundamentals ► Dimension Tables (Continued)

- Define a surrogate key
  - To keep fact tables narrow, use the smallest possible integer type
  - This also helps to take advantage of bitmap filtering for star joins
- With the exception of **DateKey**, use an IDENTITY column
  - Set the identity seed as the minimum data type value

Data Type	Range	Storage
bigint	$-2^{63}$ (-9,223,372,036,854,775,808) to $2^{63}-1$ (9,223,372,036,854,775,807)	8 Bytes
int	$-2^{31}$ (-2,147,483,648) to $2^{31}-1$ (2,147,483,647)	4 Bytes
smallint	$-2^{15}$ (-32,768) to $2^{15}-1$ (32,767)	2 Bytes
tinyint	0 to 255	1 Byte

## Physical Table Designs

### Table Design Fundamentals ► Dimension Tables (Continued)



- Always store the business key(s)
- When required, include SCD tracking columns:
  - StateDateKey (int)
  - EndDateKey (int)
  - IsCurrent (bit)
  - IsInferredMember (bit)



## Physical Table Designs

### Table Design Fundamentals ► Fact Tables

- **Fact Tables** are usually loaded incrementally
- Column types:
  - Dimension (surrogate) key
  - Measures
  - ETL tracking, and lineage
- Avoid NULL dimension key values
  - Instead, reference "Unknown" dimension members
- A primary key may serve no useful purpose

## Physical Table Designs

### Table Design Fundamentals ► Data Compression



- SQL Server data compression helps to reduce the size of the database, and improve query performance
  - Table data can be compressed at row or page level
  - It functions transparently, and compressed tables can be queried in the same way as non-compressed tables

## Physical Table Designs

### Table Design Fundamentals ► Data Compression (Continued)

- As compressed data is stored in fewer pages, queries need to read fewer pages from the disk, thereby improving the performance of I/O intensive workloads
- However, this is at the cost of extra CPU resources are required to compress and decompress the data, while data is exchanged with the application



## Physical Table Designs

### Table Design Fundamentals ► Data Compression (Continued)

- In a data warehouse design, providing there is CPU headroom available, page-level compression is recommended for all types of tables
  - Good compression rates between 75-90% can often be achieved on fact table data



## Physical Table Designs

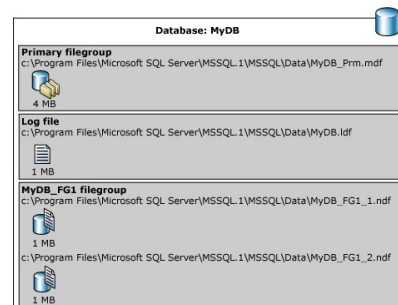
### Table Design Fundamentals ► Table Storage

- Tables (and indexes) are created on:
  - Filegroups, or
  - Partition schemes (table partitioning)
- Specifically, table partitioning addresses the challenges of querying and managing large tables
- The storage mechanism of a table, once created, cannot be subsequently altered

## Physical Table Designs

### Table Design Fundamentals ► Table Storage ► Filegroups

- SQL Server maps a database over a set of operating-system files
- Filegroups are named collections of files used for data placement, and also administration
  - The **PRIMARY** filegroup stores system objects
  - User-defined filegroups can be created to control how data is distributed over multiple storage devices and files





©2016 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

## Module Outline

### 07 | Designing a Physical Database Architecture

Topic
Physical Table Designs
Table Partitioning
<b>Demo:</b> Developing Optimized Table Designs

## Table Partitioning

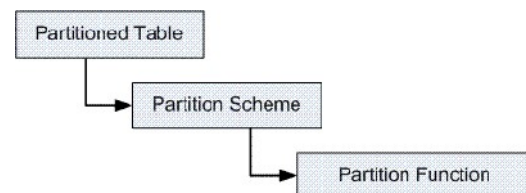
- Table partitioning addresses the challenges of querying and managing large tables
- Benefits:
  - Faster data loading and deleting
  - Faster queries when restricted to a single partition
  - Faster, more granular index maintenance
  - More flexible backup and restore options



## Table Partitioning

(Continued)

- A partitioned table is a unique kind of table in SQL Server
- It depends on two pre-existing objects, used only for partitioned tables and indexes:
  - Partition function
  - Partition scheme
- A partitioned table is queried in exactly the same way as a regular table



## Table Partitioning

### Considerations

- Consider partitioning fact tables that are 50 to 100GB, or larger
- Partitioning a table requires careful analysis to choose the appropriate:
  - Partition column
  - Boundary values for a partition function, and
  - Filegroup placement for the partition scheme
- Partitioning dimension tables is uncommon



## Table Partitioning

### Considerations (Continued)

- Typically, partition fact tables on a date key
  - This enables managing a sliding window scenario
- Choose partition grain carefully
  - Use a consistent grain: year, quarter, month, week or day
  - A maximum of 15,000 partitions is supported for a table or index
  - Partition grain affects query parallelism

## Table Partitioning

### Development Methodology

1. Create filegroups and files
2. Create a partition function to determine assignment of fact records by a date key
3. Create a partition scheme to determine the mapping between partitions and filegroups
4. Create the fact table on the partition scheme
5. Manage data load/delete, and index operations at partition level



## Table Partitioning

### Partitioning Components ► Partition Function

- The **Partition Function** defines the boundary values of the initial set of partitions and the data type of the partitioned column:
  - It makes no reference to any tables or disk storage
  - It forms the basis for one or more partition schemes

## Table Partitioning

Partitioning Components ► Partition Function ► Example

```
CREATE PARTITION FUNCTION [AnnualSales](INT)
AS RANGE RIGHT FOR VALUES
(
    -- Partition 1 -- Before 2010
    20100101 -- Partition 2 -- 2010
    ,20110101 -- Partition 3 -- 2011
    ,20120101 -- Partition 4 -- 2012
    ,20130101 -- Partition 5 -- 2013 (and beyond)
);
```

## Table Partitioning

Partitioning Components ► Partition Scheme

- The **Partition Scheme** maps particular partitions to filegroups
  - A given partition scheme can be used for one or more partitioned tables, indexes, and indexed views



## Table Partitioning

Partitioning Components ► Partition Scheme ► Example

```
CREATE PARTITION SCHEME [AnnualSalesScheme]
AS PARTITION [AnnualSales] TO
(
    [ResellerSales_2009]
    ,[ResellerSales_2010]
    ,[ResellerSales_2011]
    ,[ResellerSales_2012]
    ,[ResellerSales_2013]
);
```

## Table Partitioning

Partitioning Components ► Partitioned Table

- The **Partitioned Table** (or index) is tied to a particular partition scheme when it is created:
  - The partition table has only an indirect relationship, through the partition scheme, to the partition function

# Table Partitioning

Partitioning Components ► Partitioned Table ► Example

```
CREATE TABLE [dbo].[FactResellerSales]
(
    [ProductKey] [smallint] NOT NULL
    , [OrderDateKey] [int] NOT NULL
    , [DueDateKey] [int] NOT NULL
    , [ShipDateKey] [int] NOT NULL
    , [ResellerKey] [int] NOT NULL
    , [EmployeeKey] [int] NOT NULL
    , ...
) ON [AnnualSalesScheme]([OrderDateKey]);
GO
```



# Microsoft

©2016 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

## Module Outline

### 07 | Designing a Physical Database Architecture

Topic
Physical Table Designs
Table Partitioning
<b>Demo: Developing Optimized Table Designs</b>

## Demo

---

### Developing Optimized Table Designs

Demo objectives:

1. Revise the design of the product dimension tables
2. Revise the design of the **FactResellerSales** table



©2016 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.