Delivering a Relational Data Warehouse

Week 2 – Designing a Relational Data Warehouse Schema

Module 06

Exploring Additional Schema Design Concepts

Microsoft

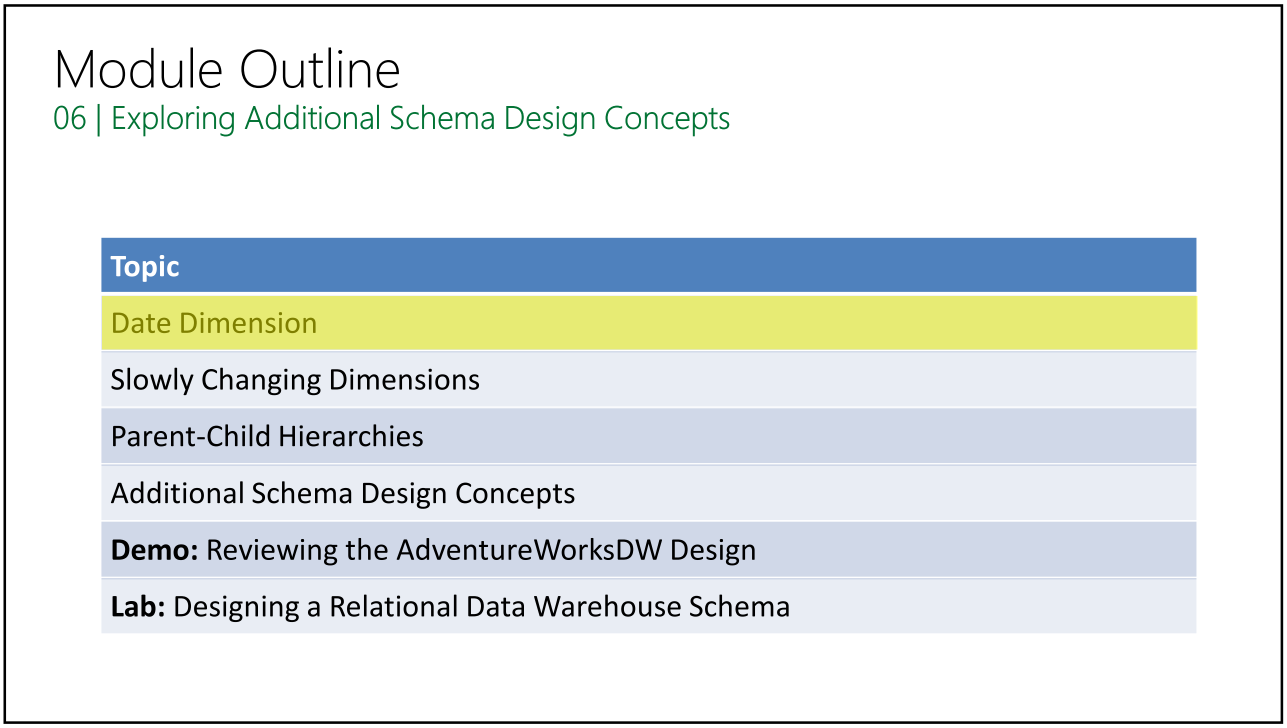
Module Outline

06 | Exploring Additional Schema Design Concepts

	Topic
▶	Date Dimension
▶	Slowly Changing Dimensions
▶	Parent-Child Hierarchies
▶	Additional Schema Design Concepts
▶	**Demo:** Reviewing the AdventureWorksDW Design
▶	**Lab:** Designing a Relational Data Warehouse Schema

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

06 | Exploring Additional Schema Design Concepts

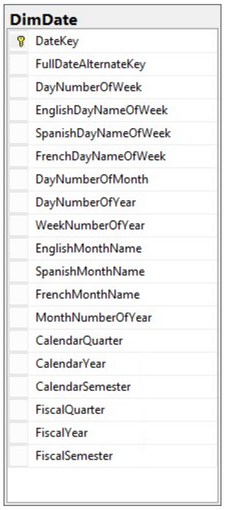
Topic
Date Dimension
Slowly Changing Dimensions
Parent-Child Hierarchies
Additional Schema Design Concepts
Demo: Reviewing the AdventureWorksDW Design
Lab: Designing a Relational Data Warehouse Schema

Date Dimension

- The **Date** dimension is the most common dimension used in analysis
- Provides more efficient and flexible analysis over time, rather than using a date value in the fact table
- Conformed for consistent use across all fact tables
  - And, ensures consistent date analysis when referenced by Self-Service BI solutions
- May be referenced multiple times by a single fact table
  - Role playing: Order date, Due date, Ship date, etc.

Date Dimension

(Continued)



DimDate

DateKey
FullDateAlternateKey
DayNumberOfWeek
EnglishDayNameOfWeek
SpanishDayNameOfWeek
FrenchDayNameOfWeek
DayNumberOfMonth
DayNumberOfYear
WeekNumberOfYear
EnglishMonthName
SpanishMonthName
FrenchMonthName
MonthNumberOfYear
CalendarQuarter
CalendarYear
CalendarSemester
FiscalQuarter
FiscalYear
FiscalSemester

- Stores one row per date (i.e. day grain)
- Includes useful attributes to enable time period analysis
  - For example, Year, Quarter, Month, Week, and Day
  - Attributes are organized into hierarchies, such as calendar or fiscal, for navigation and summarization

Date Dimension

Recommended Practices

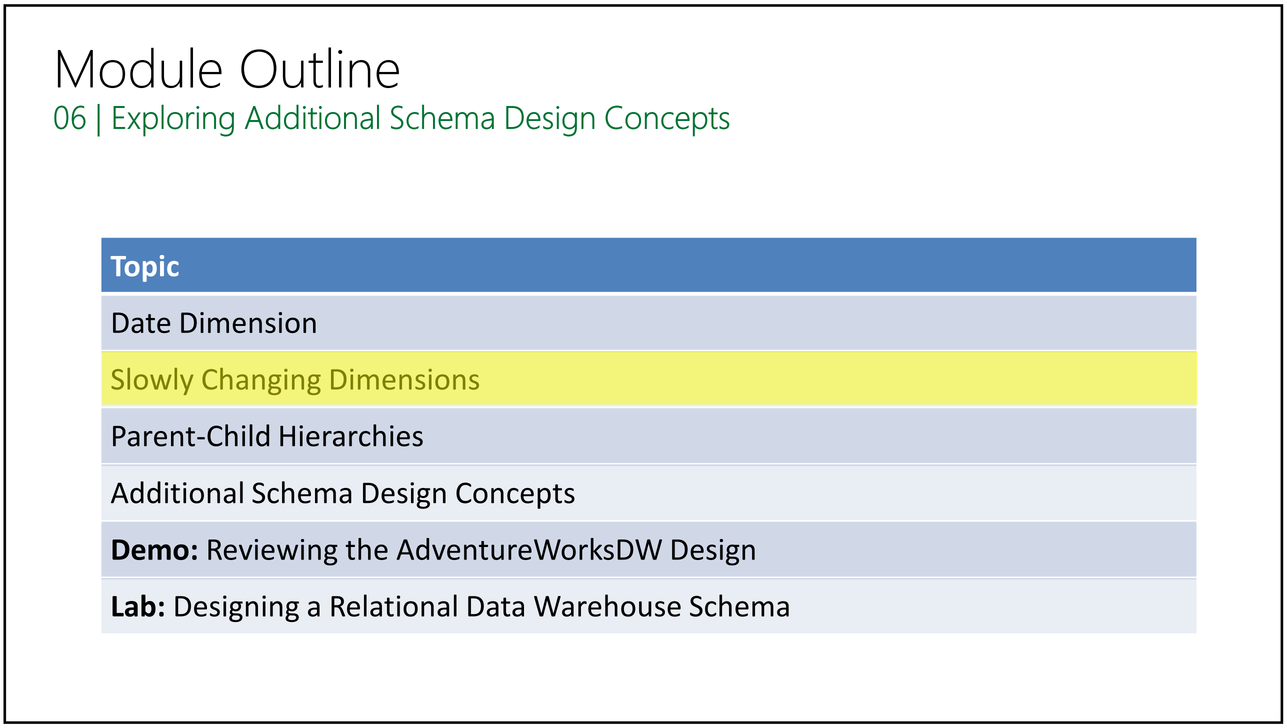
- Single table design (never snowflake)
- Define a key integer using YYYYMMDD format (i.e. 20160622)
  - The key values are human-readable
  - This is reduces the likelihood of error when configuring partitions, which are typically time-based
- Some designs name this the Time dimension
  - However, if there is need to store facts at hour/minute/second grain, then a separate table storing all possible periods within a day should be considered—name this the Time dimension

Date Dimension

Recommended Practices (Continued)

- To enable Time Intelligence calculations in SSAS tabular models, the table must:
  - Include a column of type Date
  - Have no gaps between the first (min) and last (max) dates

©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

06 | Exploring Additional Schema Design Concepts

Topic
Date Dimension
Slowly Changing Dimensions
Parent-Child Hierarchies
Additional Schema Design Concepts
Demo: Reviewing the AdventureWorksDW Design
Lab: Designing a Relational Data Warehouse Schema

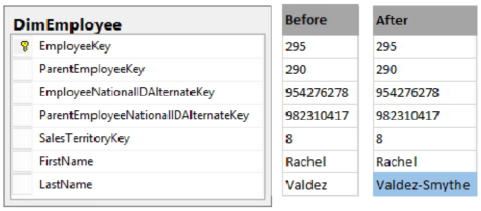
Slowly Changing Dimensions

- Support a primary role of data warehouse to describe the past accurately
- Maintain historical context as new, or changed data, is loaded into dimension tables
- Implement changes by Slowly Changing Dimension (SCD) type:
  - Type 1: Overwrite the existing dimension record
  - Type 2: Insert a new 'versioned' dimension record
  - Type 3: Track limited history with attributes

Slowly Changing Dimensions

Type 1

- Existing record is updated
  - History is not preserved
  - Common form of Slowly Changing Dimension



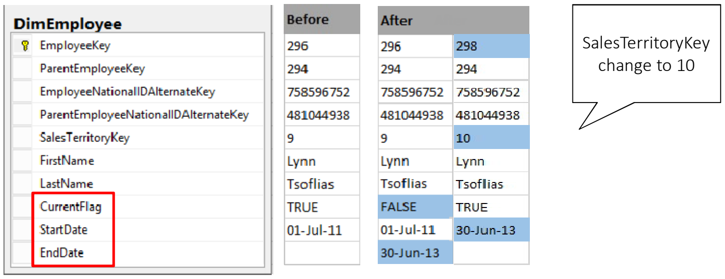
DimEmployee	Before	After
EmployeeKey	295	295
ParentEmployeeKey	290	290
EmployeeNationalIDAlternateKey	954276278	954276278
ParentEmployeeNationalIDAlternateKey	982310417	982310417
SalesTerritoryKey	8	8
FirstName	Rachel	Rachel
LastName	Valdez	Valdez-Smythe

LastName change to Valdez-Smythe

Slowly Changing Dimensions

Type 2

- Existing record is 'expired' and new record inserted
  - History is preserved
  - Surrogate key is required
  - Common form of Slowly Changing Dimension



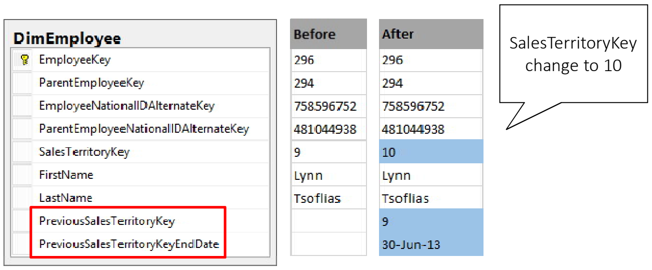
DimEmployee	Before	After	
EmployeeKey	296	296	298
ParentEmployeeKey	294	294	294
EmployeeNationalIDAlternateKey	758596752	758596752	758596752
ParentEmployeeNationalIDAlternateKey	481044938	481044938	481044938
SalesTerritoryKey	9	9	10
FirstName	Lynn	Lynn	Lynn
LastName	Tsoflias	Tsoflias	Tsoflias
CurrentFlag	TRUE	FALSE	TRUE
StartDate	01-Jul-11	01-Jul-11	30-Jun-13
EndDate		30-Jun-13	

SalesTerritoryKey change to 10

Slowly Changing Dimensions

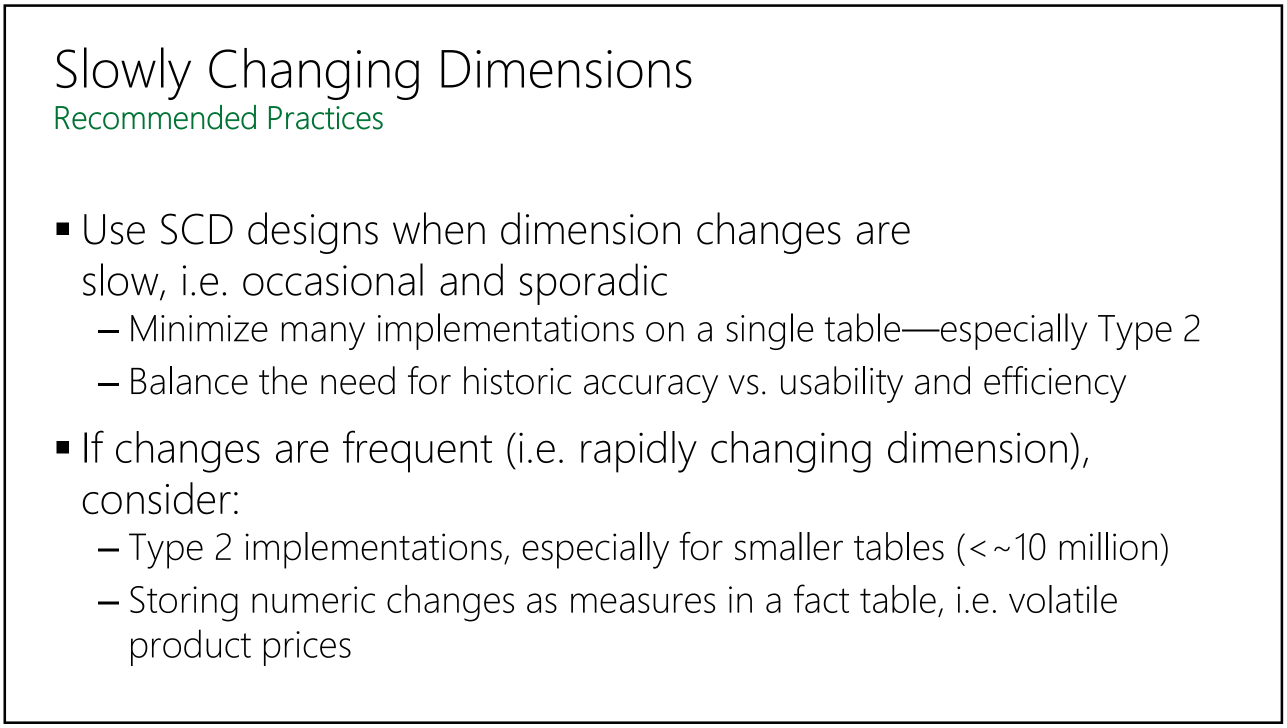
Type 3

- Existing record is updated
  - Limited history is preserved
  - Implementations are uncommon



DimEmployee	Before	After
EmployeeKey	296	296
ParentEmployeeKey	294	294
EmployeeNationalIDAlternateKey	758596752	758596752
ParentEmployeeNationalIDAlternateKey	481044938	481044938
SalesTerritoryKey	9	10
FirstName	Lynn	Lynn
LastName	Tsoflias	Tsoflias
PreviousSalesTerritoryKey		9
PreviousSalesTerritoryKeyEndDate		30-Jun-13

SalesTerritoryKey change to 10



Slowly Changing Dimensions

Recommended Practices

- Use SCD designs when dimension changes are slow, i.e. occasional and sporadic
  - Minimize many implementations on a single table—especially Type 2
  - Balance the need for historic accuracy vs. usability and efficiency
- If changes are frequent (i.e. rapidly changing dimension), consider:
  - Type 2 implementations, especially for smaller tables (<~10 million)
  - Storing numeric changes as measures in a fact table, i.e. volatile product prices

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.

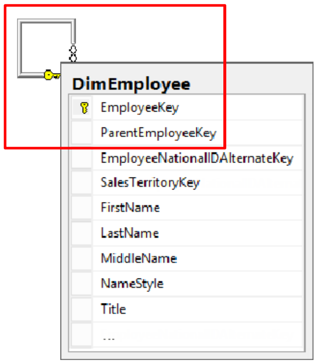
Module Outline

06 | Exploring Additional Schema Design Concepts

Topic
Date Dimension
Slowly Changing Dimensions
Parent-Child Hierarchies
Additional Schema Design Concepts
Demo: Reviewing the AdventureWorksDW Design
Lab: Designing a Relational Data Warehouse Schema

Parent-Child Hierarchies

- A dimension that defines a recursive relationship can be used to generate a parent-child hierarchy
- These hierarchies are usually ragged
  - Leaf (bottom) members at varying depths
- Common business examples include:
  - Organization charts
  - General Ledger structures
  - Bill of materials



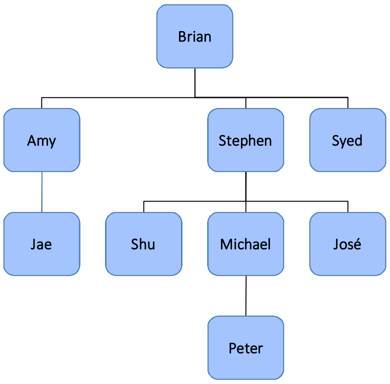
DimEmployee

EmployeeKey
ParentEmployeeKey
EmployeeNationalIDAlternateKey
SalesTerritoryKey
FirstName
LastName
MiddleName
NameStyle
Title
...

Parent-Child Hierarchies

Example ► Organization Chart

EmployeeKey	ParentEmployeeKey	Employee
277	277	Brian Welcker
290	277	Amy Alberts
272	277	Stephen Jiang
294	277	Syed Abbas
287	272	Shu Ito
281	272	Michael Blythe
288	272	José Saraiva
291	290	Jae Pak
299	281	Peter Myers



Parent-Child Hierarchies

Characteristics

- Parent-child hierarchies differ fundamentally from regular "fixed level" hierarchies
- For regular hierarchies:
  - Members of a level (siblings) are of the same type (year, quarter, etc.)
  - Facts are attached at a lower level, and aggregated (rolled up) to higher levels

Parent-Child Hierarchies

Characteristics (Continued)

- For parent-child hierarchies:
  - Each member is of the same type
    - For example, in an organization chart, every member is an employee
    - As another example, in a bill of materials, each member is a product
  - Facts can be attached at any level of the hierarchy, and so a member represents its own value rolled up with its descendants' values

Parent-Child Hierarchies

Recommended Practices

- Avoid implementing SCD Type 2 changes
  - Consider what happens when a non-leaf member changes... all descendants must be versioned also!
- The root member(s) should be either:
  - NULL (if allowed)
  - Or, define the parent key by using their key

Parent-Child Hierarchies

Recommended Practices (Continued)

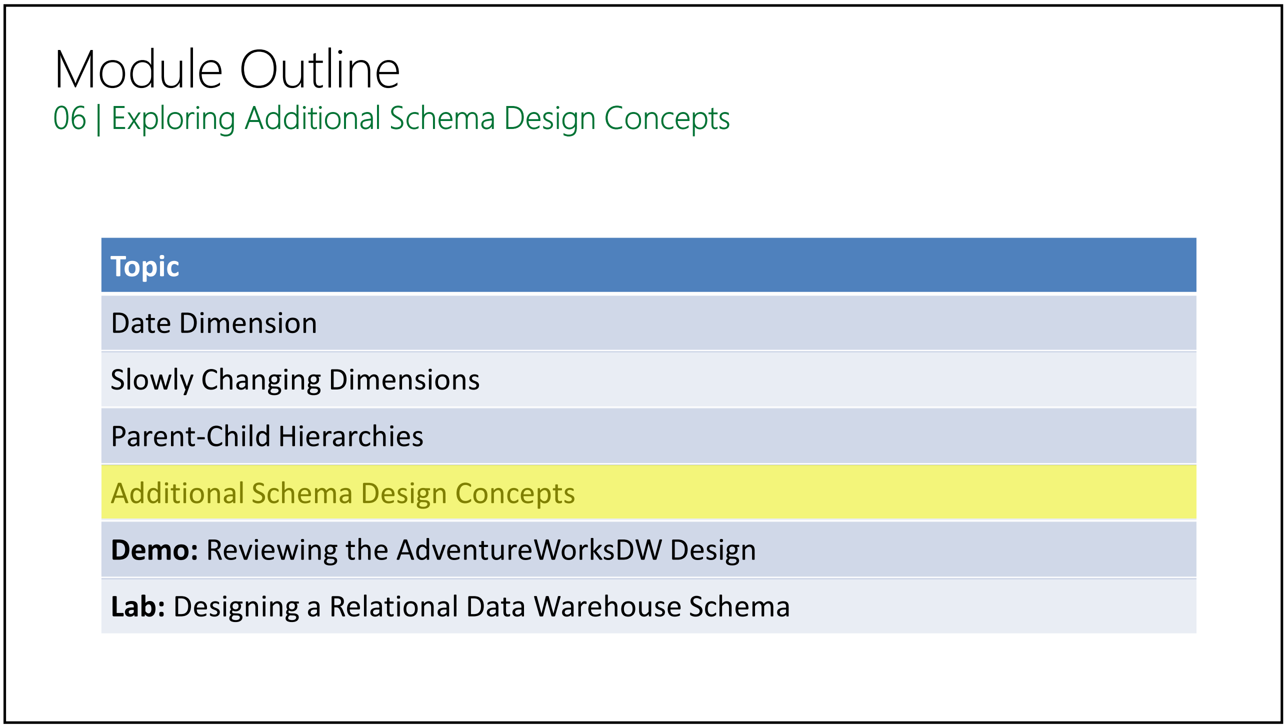
- Develop SSAS multidimensional models
  - There is native support to develop only one parent-child hierarchy per dimension
    - Note: Aggregations for parent-child hierarchies cannot be pre-computed
  - Advanced configurations can leverage the UnaryOperator property to control rollup behavior, and will require that values (+, -, ~) be stored in the dimension table

Parent-Child Hierarchies

Recommended Practices (Continued)

- If developing a SSAS tabular model, consider naturalizing the recursive relationship into fixed dimension columns (Level1, Level2, etc.)
  - SSAS tabular models cannot express a recursive relationship as a hierarchy
  - Data Analysis Expression (DAX) does include a set of PATH functions that can be used to naturalize the recursive relationship
  - However, there is no native support for the unary operator, and the way a ragged hierarchy is expressed is often confusing to users (i.e. blanks are shown where no member actually exists)

©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

06 | Exploring Additional Schema Design Concepts

Topic
Date Dimension
Slowly Changing Dimensions
Parent-Child Hierarchies
Additional Schema Design Concepts
Demo: Reviewing the AdventureWorksDW Design
Lab: Designing a Relational Data Warehouse Schema

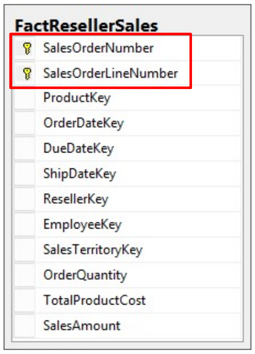
Additional Schema Design Concepts

- Degenerate dimensions
- Junk dimensions
- Factless fact tables

Additional Schema Design Concepts

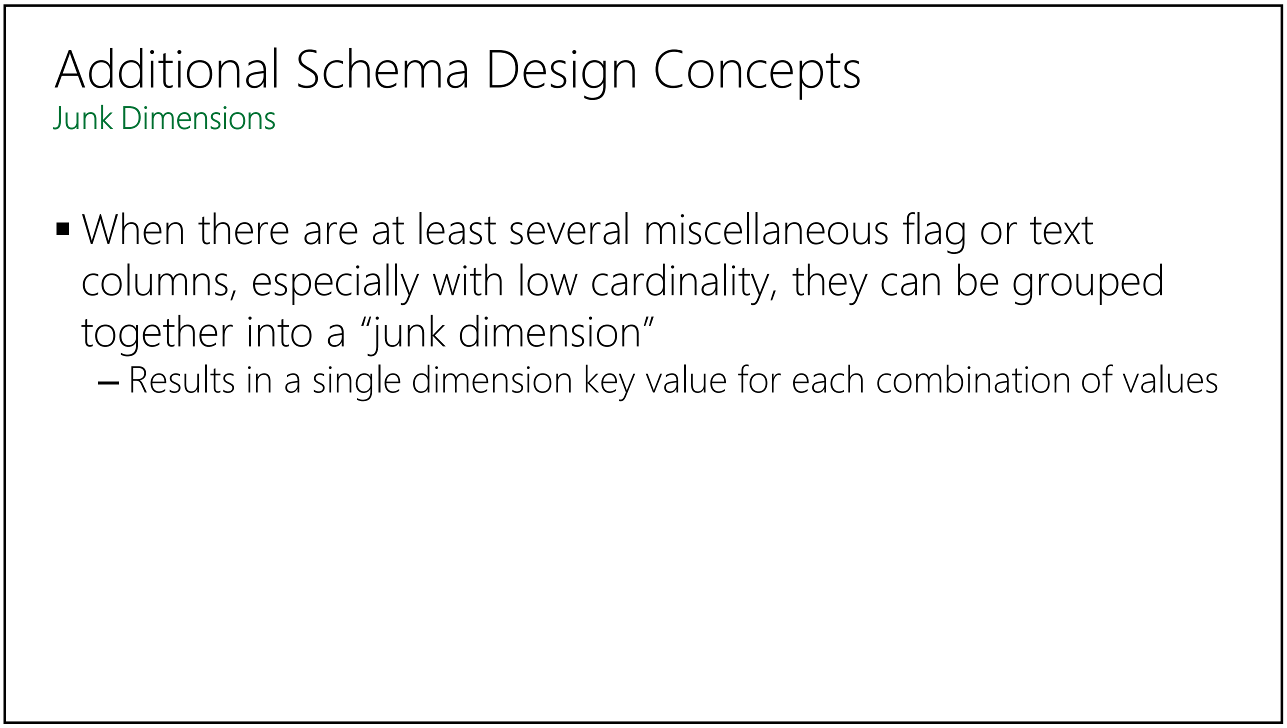
Degenerate Dimensions

- A degenerate dimension is sourced directly from fact table columns
  - It does not make sense to design a dimension that consists of a single attribute (e.g. order number)
  - Common examples include order, invoice or tracking numbers



FactResellerSales

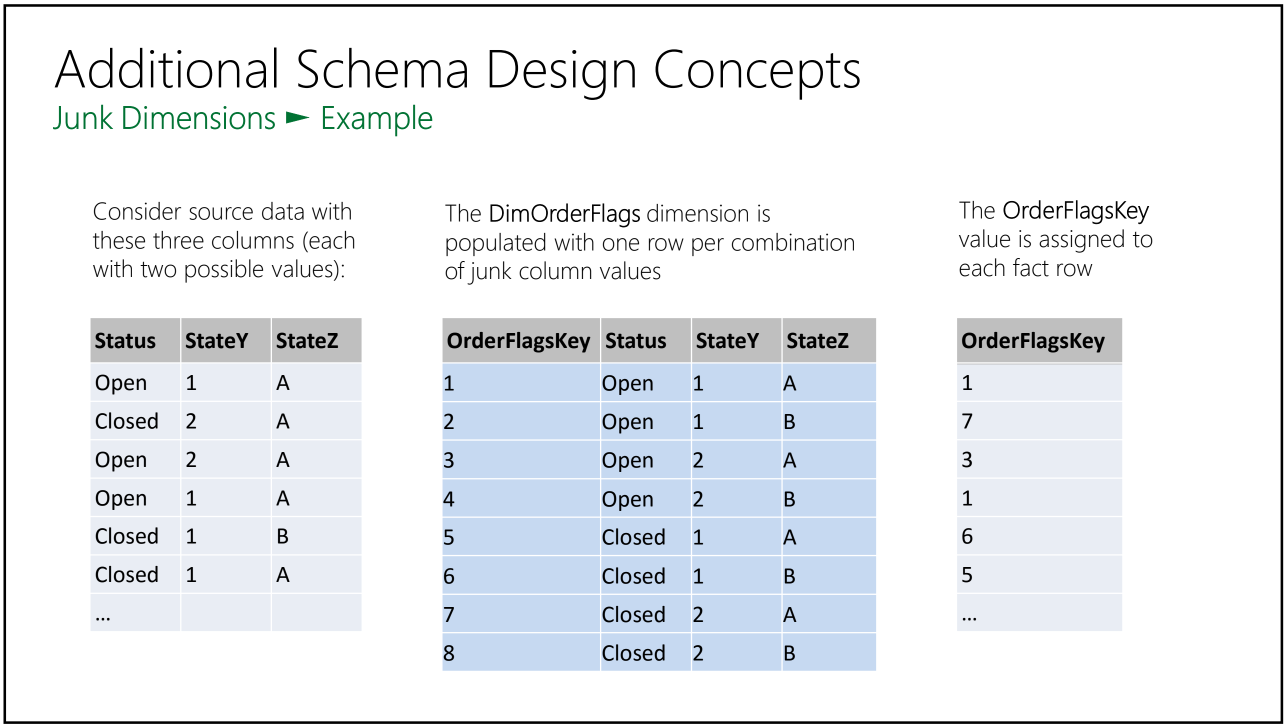
- SalesOrderNumber
- SalesOrderLineNumber
- ProductKey
- OrderDateKey
- DueDateKey
- ShipDateKey
- ResellerKey
- EmployeeKey
- SalesTerritoryKey
- OrderQuantity
- TotalProductCost
- SalesAmount



Additional Schema Design Concepts

Junk Dimensions

- When there are at least several miscellaneous flag or text columns, especially with low cardinality, they can be grouped together into a "junk dimension"
  - Results in a single dimension key value for each combination of values



Additional Schema Design Concepts

Junk Dimensions ► Example

Consider source data with these three columns (each with two possible values):

Status	StateY	StateZ
Open	1	A
Closed	2	A
Open	2	A
Open	1	A
Closed	1	B
Closed	1	A
…		

The DimOrderFlags dimension is populated with one row per combination of junk column values

OrderFlagsKey	Status	StateY	StateZ
1	Open	1	A
2	Open	1	B
3	Open	2	A
4	Open	2	B
5	Closed	1	A
6	Closed	1	B
7	Closed	2	A
8	Closed	2	B

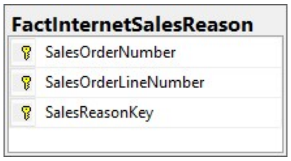
The OrderFlagsKey value is assigned to each fact row

OrderFlagsKey
1
7
3
1
6
5
…

Additional Schema Design Concepts

Factless Fact Tables

- A fact table does not always need facts to measure a process
  - Some processes are measured only by counting events or activities
- A fact table with no measures is called a factless fact table
  - What is stored is a combination of dimension key values



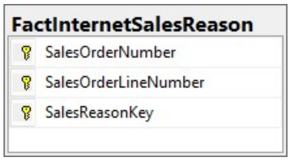
FactInternetSalesReason

- SalesOrderNumber
- SalesOrderLineNumber
- SalesReasonKey

Additional Schema Design Concepts

Factless Fact Tables (Continued)

- Count aggregations can measure the number of events
  - For example, the number of calls received by a call center
- These tables can also capture details of conditions
  - For example, the assignment of salespeople to territories for a month

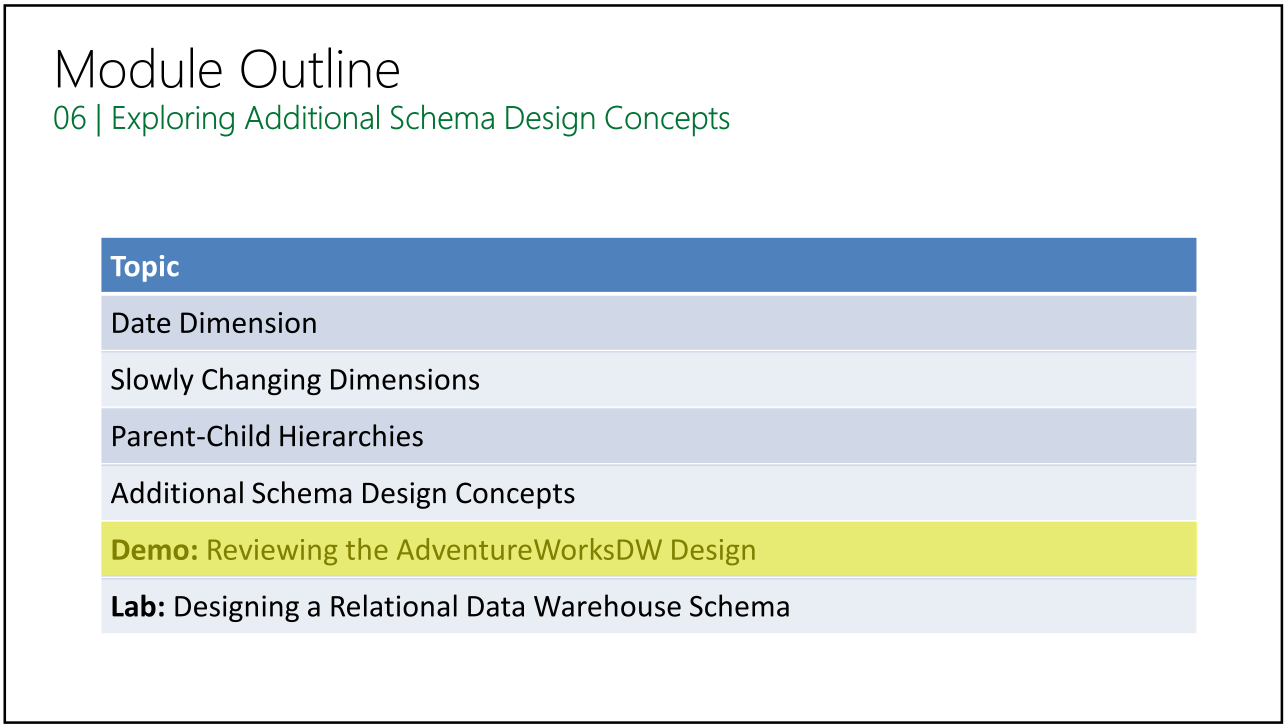


FactInternetSalesReason

- SalesOrderNumber
- SalesOrderLineNumber
- SalesReasonKey

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

06 | Exploring Additional Schema Design Concepts

Topic
Date Dimension
Slowly Changing Dimensions
Parent-Child Hierarchies
Additional Schema Design Concepts
Demo: Reviewing the AdventureWorksDW Design
Lab: Designing a Relational Data Warehouse Schema

Demo

Reviewing the AdventureWorksDW Design

Demo objectives:

1. Explore the DimDate table
2. Explore the FactResellerSales table
3. Explore the FactInternetSalesReason table
4. Review the slowly changing dimension 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

06 | Exploring Additional Schema Design Concepts

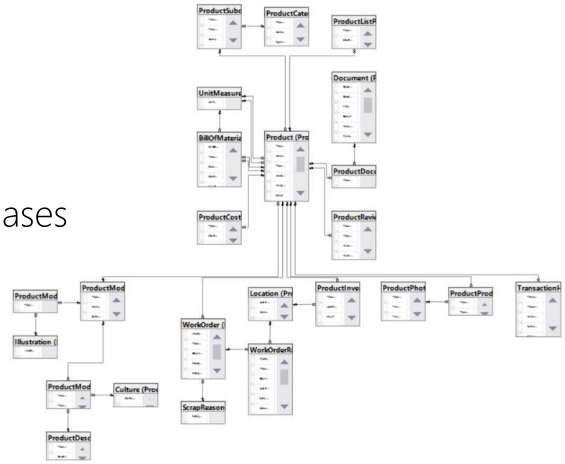
Topic
Date Dimension
Slowly Changing Dimensions
Parent-Child Hierarchies
Additional Schema Design Concepts
Demo: Reviewing the AdventureWorksDW Design
Lab: Designing a Relational Data Warehouse Schema

Lab

01 | Designing a Relational Data Warehouse Schema

Lab exercises:

1. Provisioning an Azure VM
2. Setting Up the Azure VM
3. Exploring the AdventureWorks Databases



Lab

01 | Designing a Relational Data Warehouse Schema

Tips:

- Be sure to read instructions carefully, especially when executing scripts
- There is a lab setup shortcut, and it can be used to reset and try again

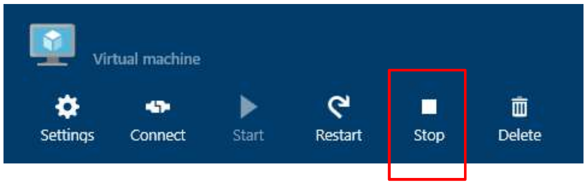
Lab

01 | Designing a Relational Data Warehouse Schema

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

©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.