Data Warehouse Systems: Design and Implementation

Second Edition

Alejandro VAISMAN
Department of Information Engineering
Instituto Tecnológico de Buenos Aires
avaisman@itba.edu.ar

Esteban ZIMÁNYI
Department of Computer & Decision Engineering (CoDE)
Université libre de Bruxelles
esteban.zimanyi@ulb.be

Chapter 5: Logical Data Warehouse Design

1. Relational Data Warehouse Design
2. Relational Implementation of the Conceptual Model
3. Logical Representation of Hierarchies
4. Advanced Modeling Aspects
5. Slowly Changing Dimensions
6. SQL/OLAP Operations
7. The Northwind Cube in Analysis Services Multidimensional

Table of Contents

1. Relational Data Warehouse Design

2. Relational Implementation of the Conceptual Model

3. Logical Representation of Hierarchies

4. Advanced Modeling Aspects

5. Slowly Changing Dimensions

6. SQL/OLAP Operations

7. The Northwind Cube in Analysis Services Multidimensional

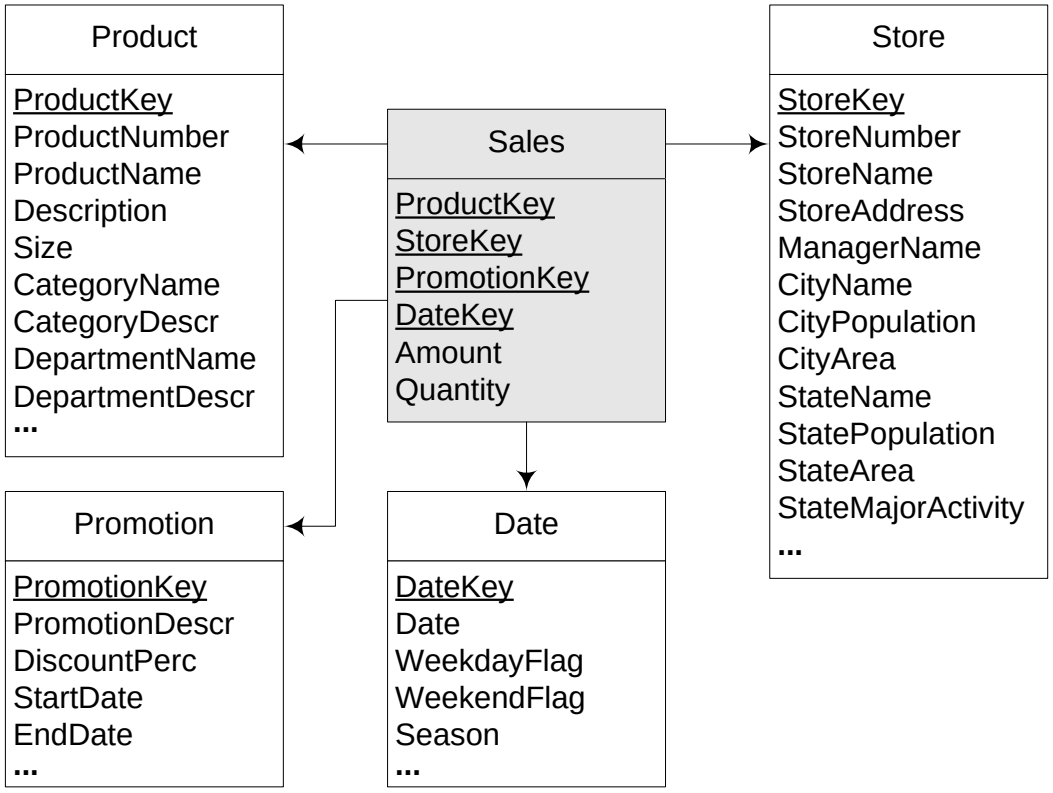
OLAP Technologies

- **Relational OLAP** (**ROLAP**)
  - Stores data in relational databases
  - Supports **SQL extensions** and special **special access methods** to efficiently implement the model and its operations
- **Multidimensional OLAP** (**MOLAP**)
  - Stores data in special data structures (e.g., arrays) and implement OLAP operations in these structures
  - **Better performance** than ROLAP for query and aggregation
  - **Less storage capacity** than ROLAP
- **Hybrid OLAP** (**HOLAP**)
  - Combines both technologies
  - E.g., detailed data stored in relational databases, aggregations kept in a separate MOLAP store

Relational Data Warehouse Design

- In ROLAP systems, tables organized in specialized structures
- **Star schema**: One **fact table** and a set of **dimension tables**
  - Referential integrity constraints between fact table and dimension tables
  - Dimension tables may contain redundancy in the presence of hierarchies
  - Dimension tables denormalized, fact tables normalized
- **Snowflake schema**: Avoids redundancy of star schemas by normalizing dimension tables
  - Normalized tables optimize storage space, but decrease performance
- **Starflake schema**: Combination of the star and snowflake schemas, some dimensions normalized, other not
- **Constellation schema**: Multiple fact tables that share dimension tables

Example of a Star Schema



Example of a Snowflake Schema

Example of a Constellation Schema

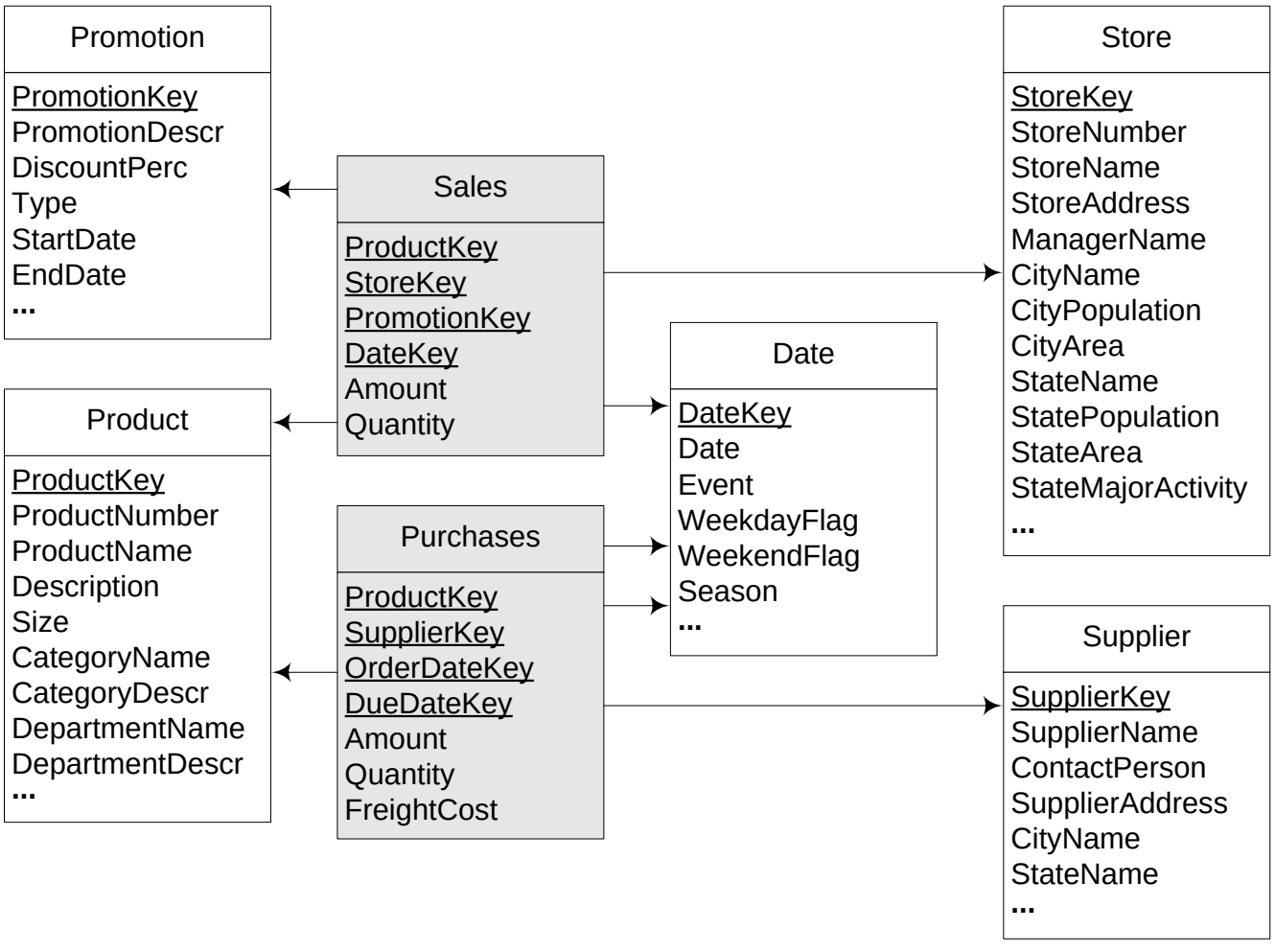


Table of Contents

1. Relational Data Warehouse Design

2. Relational Implementation of the Conceptual Model

3. Logical Representation of Hierarchies

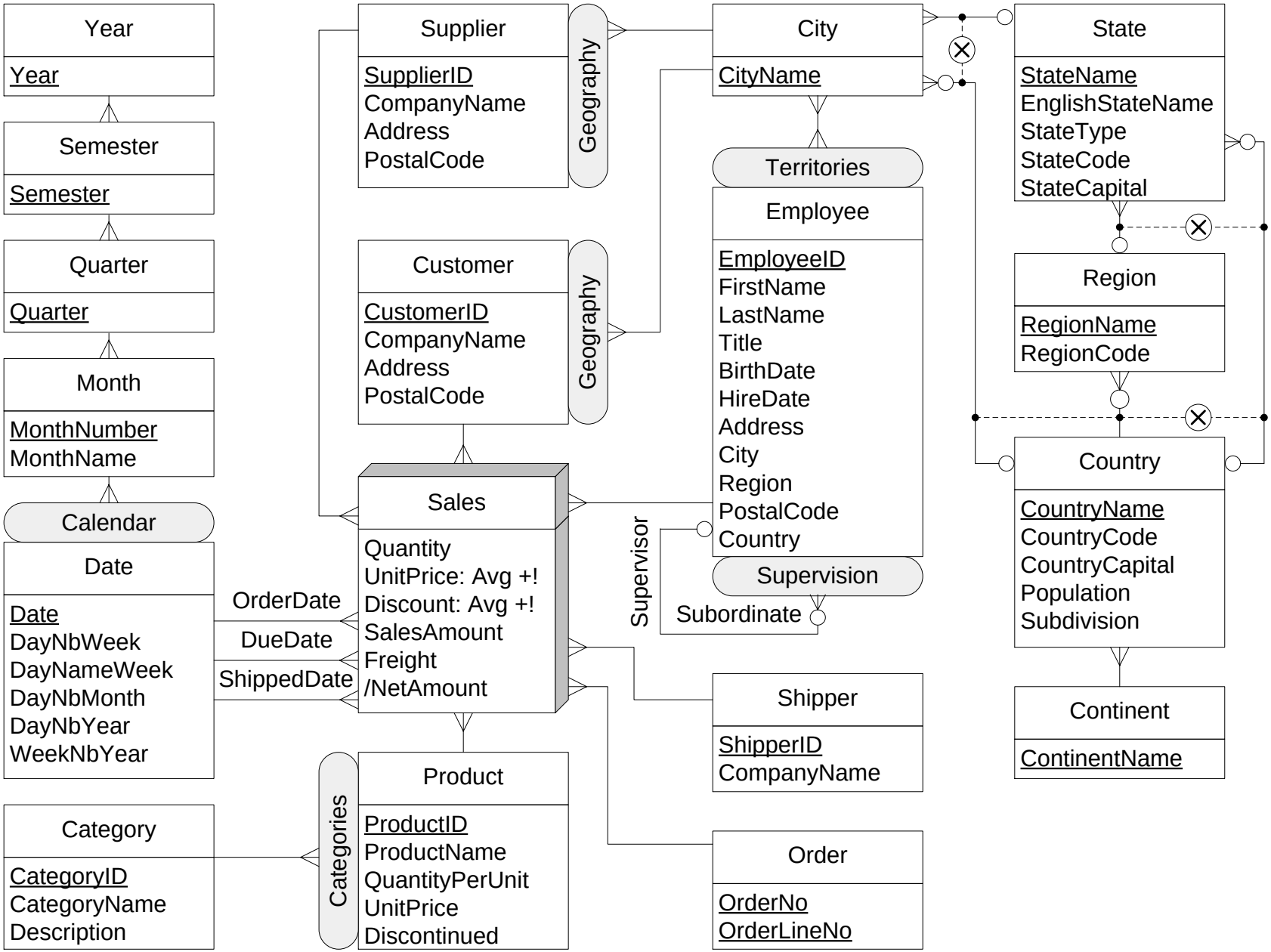
4. Advanced Modeling Aspects

5. Slowly Changing Dimensions

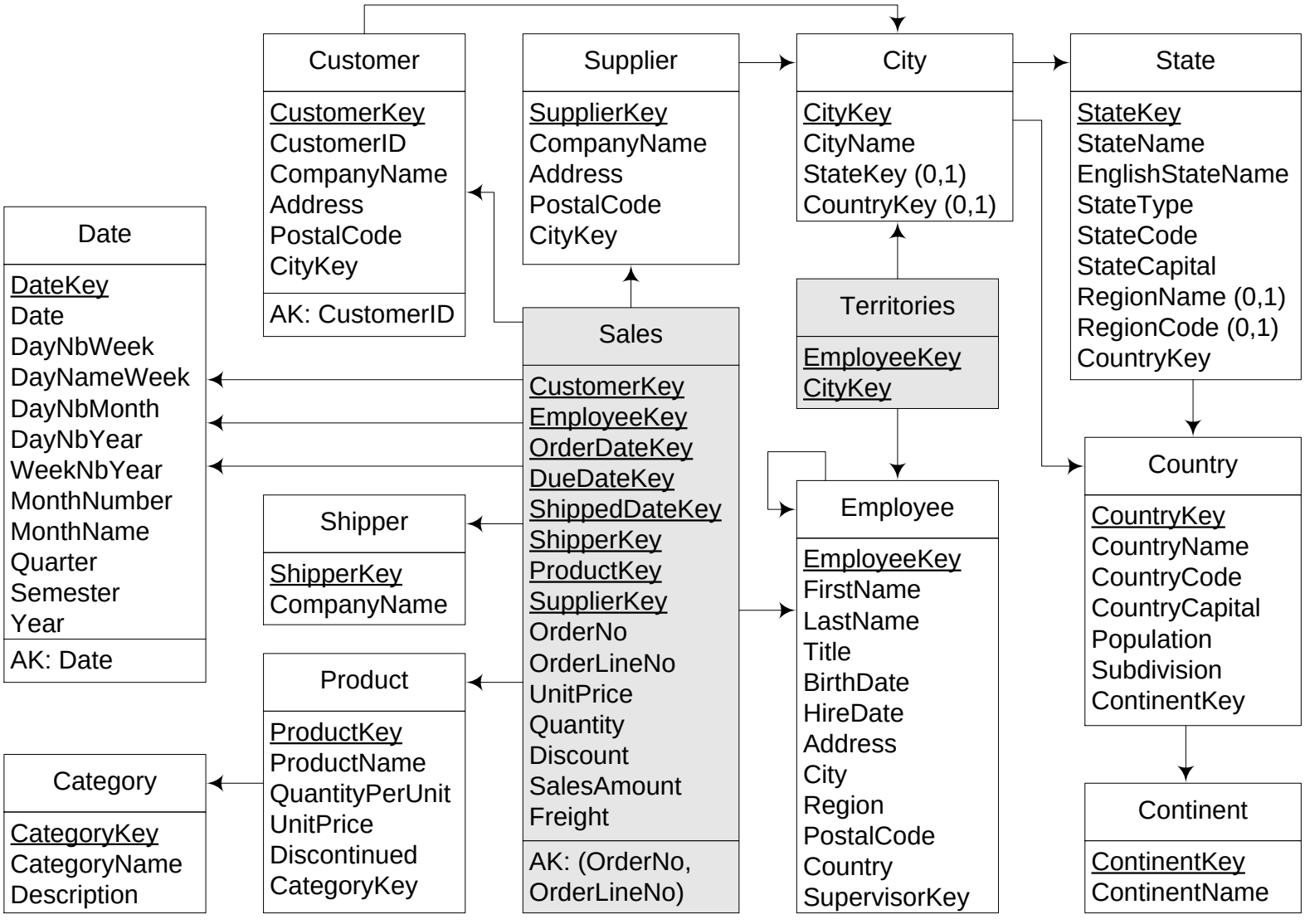
6. SQL/OLAP Operations

7. The Northwind Cube in Analysis Services Multidimensional

Conceptual Representation of the Northwind Data Warehouse



Relational Representation of the Northwind Data Warehouse



Relational Implementation of the Conceptual Model

- A set of rules to translate the conceptual model (the MultiDim model) into the relational model

Rule 1: A level L, provided it is not related to a fact with a one-to-one relationship, is mapped to a table T_L that contains all attributes of the level

- A surrogate key may be added to the table, otherwise the identifier of the level will be the key of the table
- Additional attributes will be added to this table when mapping relationships using Rule 3 below

Rule 2: A fact F is mapped to a table T_F that includes as attributes all measures of the fact

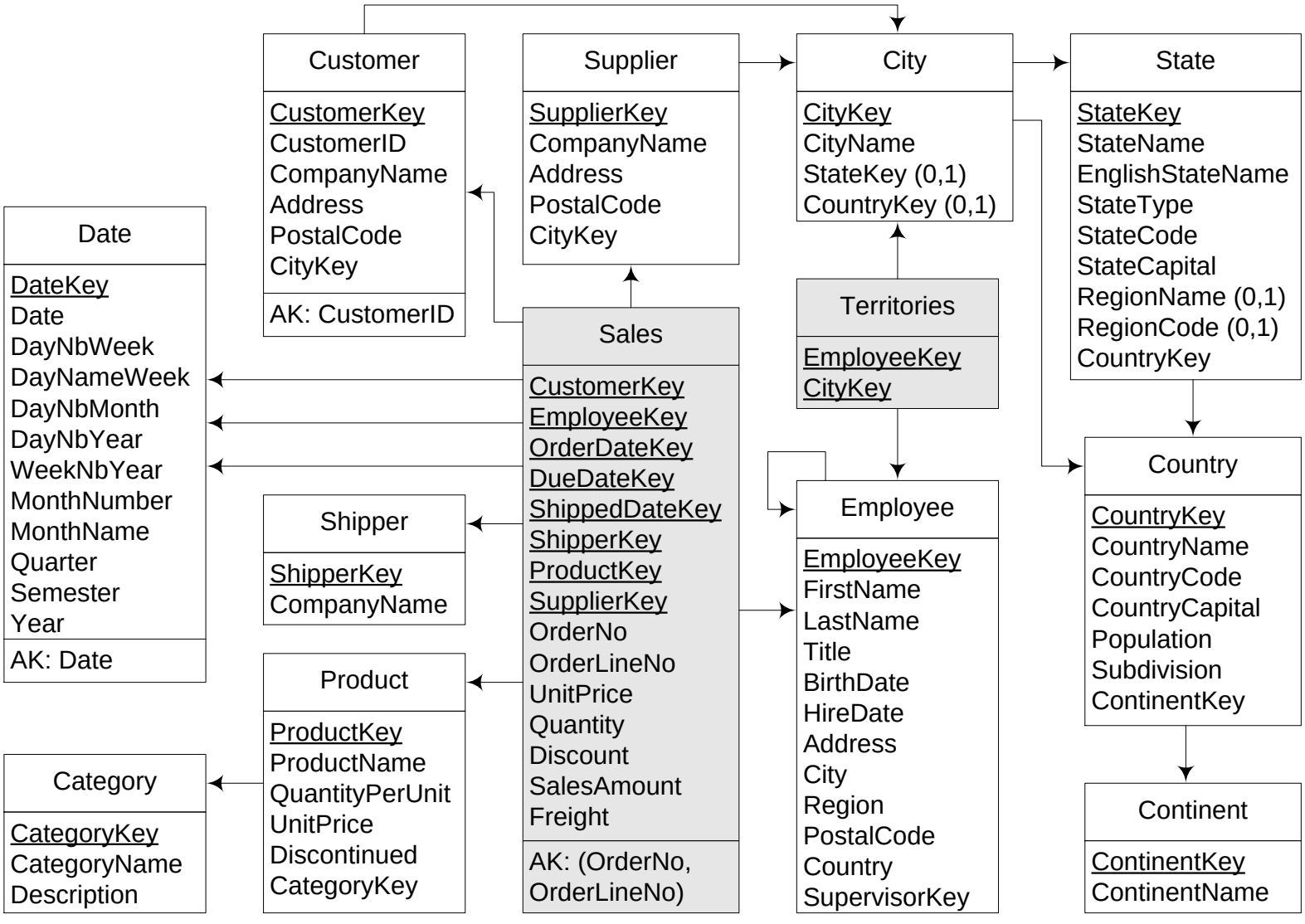
- A surrogate key may be added to the table
- Additional attributes will be added to this table when mapping relationships using Rule 3 below

Relational Implementation of the Conceptual Model

Rule 3: A relationship between either a fact F and a dimension level L, or between dimension levels L_P and L_C (standing for the parent and child levels, respectively), can be mapped in three different ways, depending on its cardinalities:

- **Rule 3a:** If the relationship is one-to-one, the table corresponding to the fact (T_F) or to the child level (T_C) is extended with all the attributes of the dimension level or the parent level, respectively
- **Rule 3b:** If the relationship is one-to-many, the table corresponding to the fact (T_F) or to the child level (T_C) is extended with the surrogate key of the table corresponding to the dimension level (T_L) or the parent level (T_P), respectively, that is, there is a foreign key in the fact or child table pointing to the other table
- **Rule 3c:** If the relationship is many-to-many, a new table T_B (standing for bridge table) is created that contains as attributes the surrogate keys of the tables corresponding to the fact (T_F) and the dimension level (T_L), or the parent (T_P) and child levels (T_C), respectively. If the relationship has a distributing attribute, an additional attribute is added to the table to store this information

Relational Representation of the Northwind Data Warehouse



Relational Representation of the Northwind Data Warehouse

- The Sales table includes one FK for each level related to the fact with a one-to-many relationship
- For Time, several roles: OrderDate, DueDate, and ShippedDate
- Order: related to the fact with a one-to-one relationship, called a **degenerate**, or a **fact dimension**
- Fact table contains five attributes representing the measures:
  - UnitPrice, Quantity, Discount, SalesAmount, and Freight.
- The many-to-many parent-child relationship between Employee and Territory is mapped to the table Territories, containing two foreign keys
- Customer has a surrogate key CustomerKey and a database key CustomerID
- SupplierKey in Supplier is a database key

The Time Dimension

- Data warehouse: a historical database
- Time dimension present in almost all data warehouses.
- In a star or snowflake schema, time is included both as foreign key(s) in a fact table and as a time dimension containing the aggregation levels
- OLTP databases: temporal information is usually derived from attributes of a DATE data type
  - Example: A weekend is computed on-the-fly using appropriate functions
- In a data warehouse time information is stored as explicit attributes in the time dimension
  - Easy to compute: Total sales during weekends

    ```
    SELECT SUM(SalesAmount)
    FROM   Date D, Sales S
    WHERE  D.DateKey = S.DateKey AND D.WeekendFlag
    ```
- The granularity of the time dimension varies depending on their use
- Time dimension with a granularity month spanning 5 years will have $5 \times 12 = 60$ tuples
- Time dimension may have more than one hierarchy

Table of Contents

1. Relational Data Warehouse Design

2. Relational Implementation of the Conceptual Model

3. Logical Representation of Hierarchies
   - Balanced Hierarchies
   - Unbalanced Hierarchies
   - Recursive Hierarchies
   - Generalized Hierarchies
   - Alternative Hierarchies
   - Parallel Hierarchies
   - Nonstrict Hierarchies

4. Advanced Modeling Aspects

5. Slowly Changing Dimensions

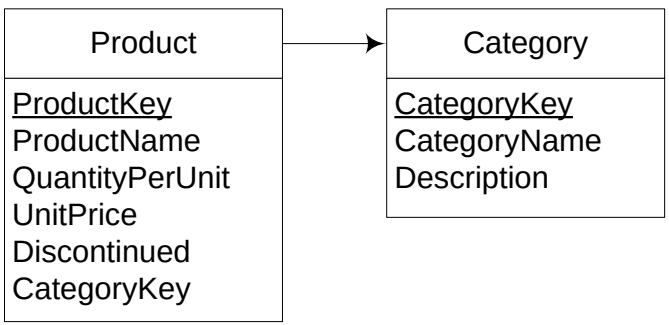
6. SQL/OLAP Operations

7. The Northwind Cube in Analysis Services Multidimensional

Balanced Hierarchies

- Applying the mapping rules to balanced hierarchies yields snowflake schemas
  - **Normalized tables** or **snowflake structure:** each level is represented as a separate table that includes the key and the descriptive attributes of the level
  - Example: applying Rules 1 and 3b to the Categories hierarchy yields a snowflake structure with tables Product and Category
- If star schemas are required we represent hierarchies using **Denormalized** or **flat tables**
  - The key and the descriptive attributes of all levels forming a hierarchy are included in one table

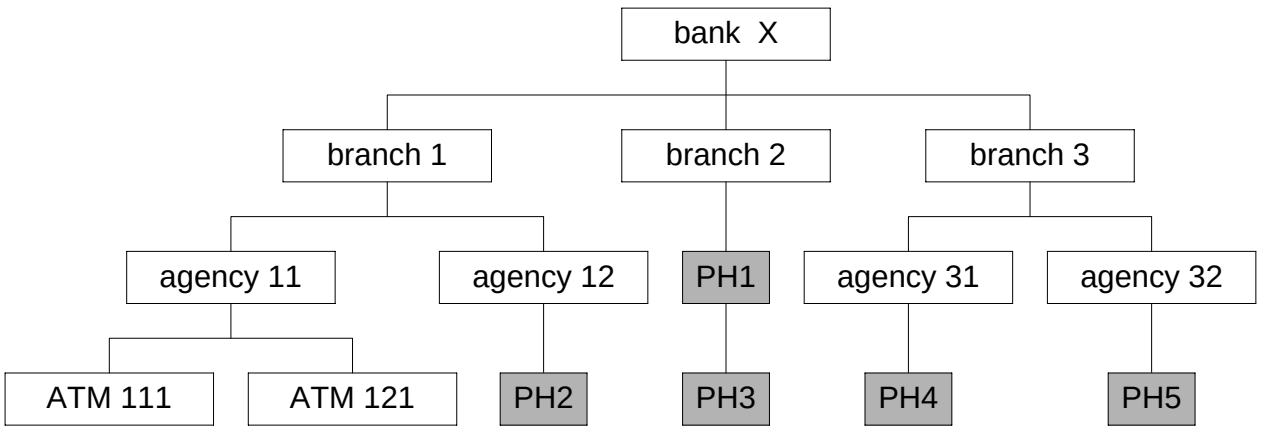
Snowflake structure



Flat table

Unbalanced Hierarchies

- Do not satisfy the summarizability conditions → mapping may exclude members without children
  - In the branches example, measures will be aggregated into higher levels only for agencies that have ATMs and only for branches that have agencies
  - To avoid this problem, an unbalanced hierarchy can be transformed into a balanced one using placeholders (marked PH1,PH2,. . .,PHn), or null values in missing levels



Unbalanced Hierarchies

- Shortcomings:
  - A fact table must include common measures belonging to different hierarchy levels, since members of any of these levels can be a leaf at the instance level
  - Common measures have different granularities
    - Example: Measures for the ATM level and for the Agency level)
  - Placeholders must be created and managed for aggregation
    - Example: The same measure value must be repeated for branch 2, while using two placeholders for the two consecutive missing levels
  - The introduction of meaningless values requires more storage space
  - A special interface must be developed to hide placeholders from users

Recursive Hierarchies

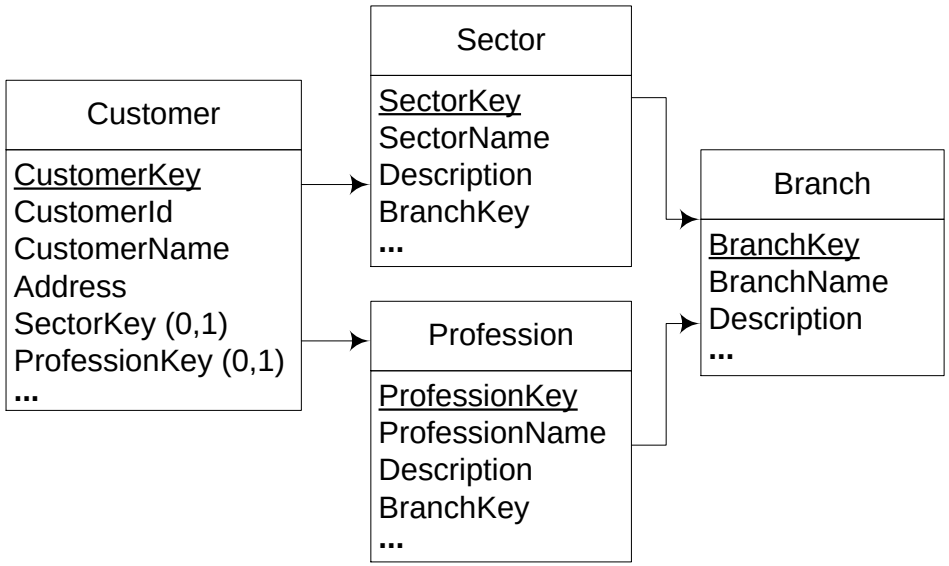
- Mapping recursive hierarchies to the relational model yields **parent-child tables** containing all attributes of a level, and an additional foreign key relating child members to their corresponding parent
- Table Employee represents a recursive hierarchy
- Operations over parent-child tables are complex, recursive queries are necessary for traversing a recursive hierarchy

Generalized Hierarchies

- Several approaches
  - Create a table for each level of the hierarchy, leading to snowflake schema
  - A flat representation with null values for attributes that do not pertain to specific members
  - Create separate separate fact and dimension tables for each path
  - Create one table for the common levels and another table for the specific ones
- Disadvantage of the first three approaches: common levels of the hierarchy cannot be easily distinguished and managed; null values require specification of additional constraints
- In the fourth solution, an additional attribute must be created in the table representing the common levels of the hierarchy

Generalized Hierarchies: Relational Representation

- Traditional mapping of generalization from the ER model to relational tables (e.g., Rule 7) presents problems due to the inclusion of null values and the loss of the hierarchical structure
- Applying the mapping to the generalized hierarchy yields

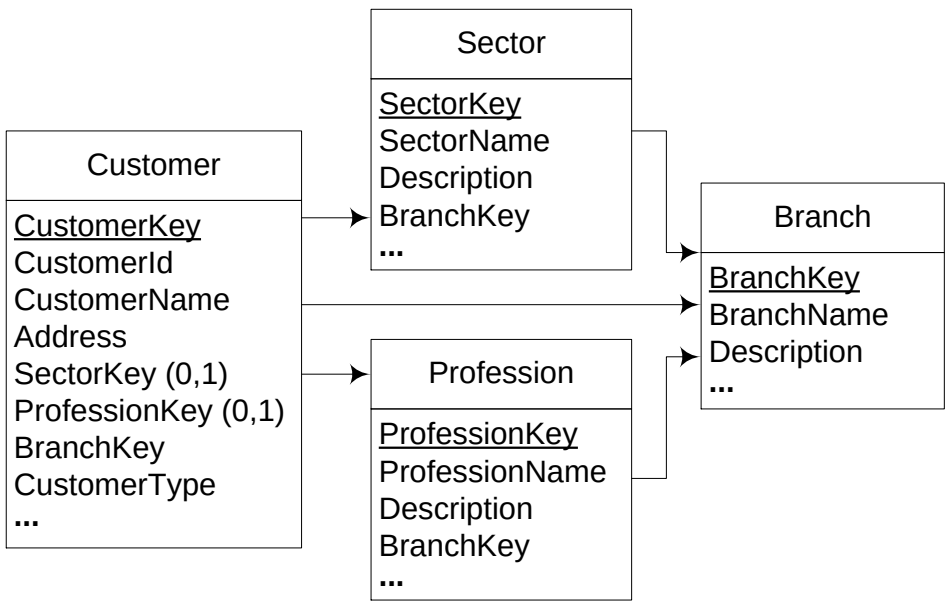


- Mapping represents the hierarchical structure, but does not allow to traverse just the common levels

Generalized Hierarchies: Improved Relational Representation

- We must add the following mapping rule:

Rule 4 : A table corresponding to a splitting level in a generalized hierarchy must have an additional attribute, which is a foreign key of the next joining level, provided this level exists. The table may also include a discriminating attribute that indicates the specific aggregation path of each member.



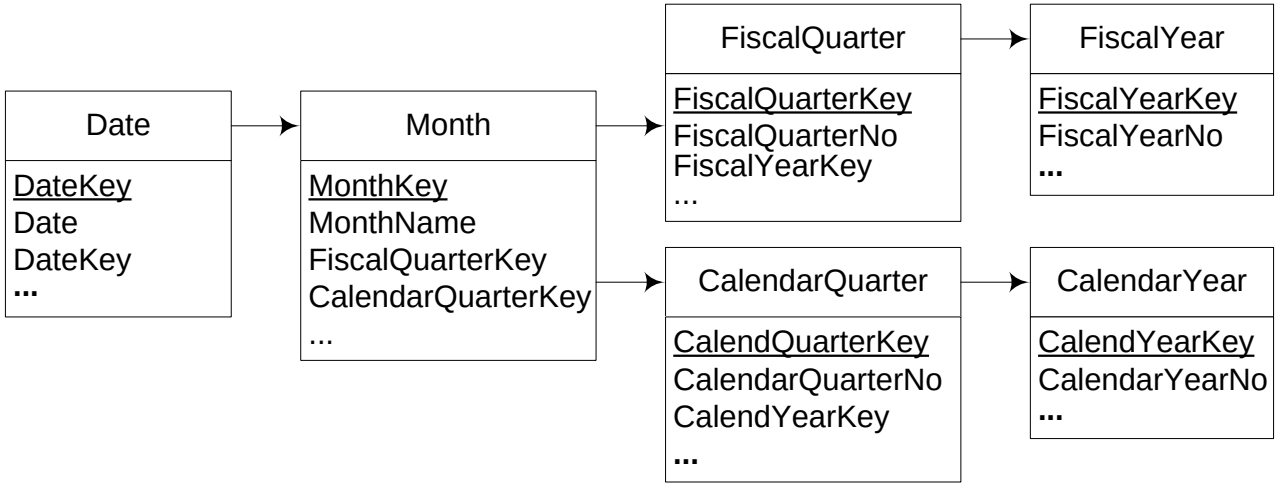
Generalized Hierarchies: Improved Relational Representation

- With this schema we can:
  - Use paths including the specific levels, for example Profession or Sector
  - Access the levels common to all members, i.e., ignore the levels between the splitting and joining ones (e.g., use the hierarchy Customer → Branch
- Integrity constraints must be specified to ensure that only one of the foreign keys for the specialized levels may have a value

```
ALTER TABLE Customer ADD CONSTRAINT CustomerTypeCK
        CHECK ( CustomerType IN ('Person', 'Company') )
ALTER TABLE Customer ADD CONSTRAINT CustomerPersonFK
        CHECK ( (CustomerType != 'Person') OR
        ( ProfessionKey IS NOT NULL AND SectorKey IS NULL ) )
ALTER TABLE Customer ADD CONSTRAINT CustomerCompanyFK
        CHECK ( (CustomerType != 'Company') OR
        ( ProfessionKey IS NULL AND SectorKey IS NOT NULL ) )
```

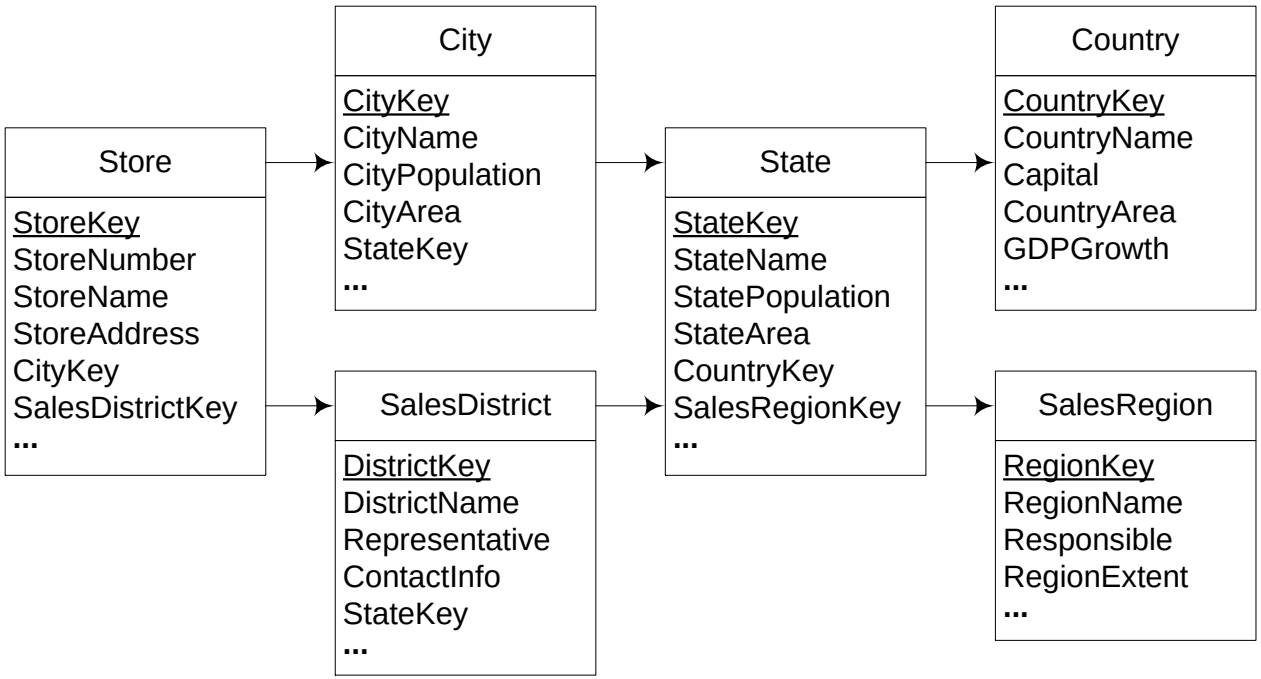
Alternative Hierarchies

- Traditional mapping to relational tables can be applied
- Generalized and alternative hierarchies distinguished at the conceptual level, not at logical level



Parallel Hierarchies

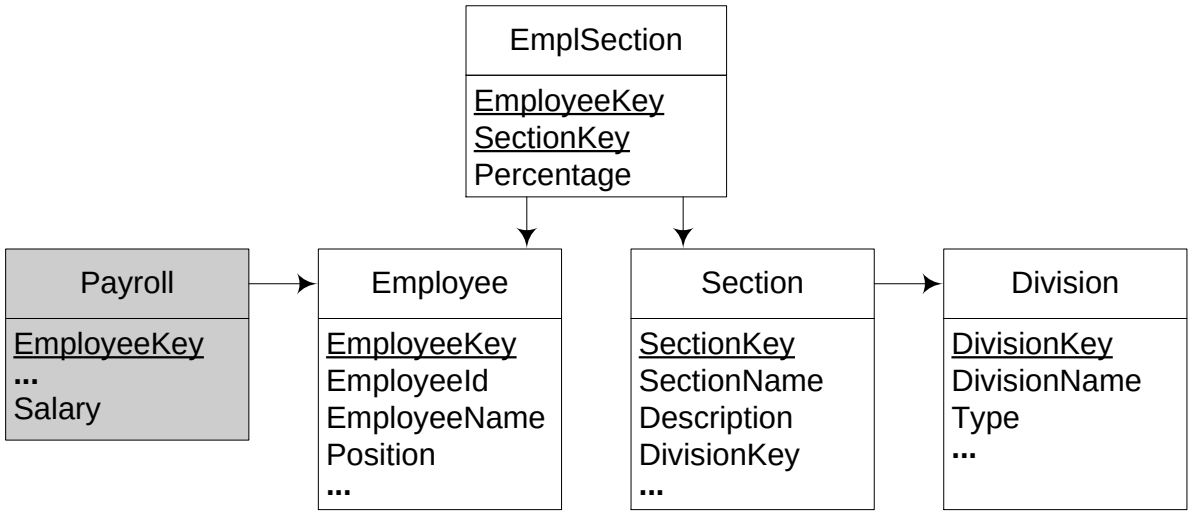
- Composed of several hierarchies → logical mapping combines the mappings for each type



- Shared levels represented in one table (e.g., State).

Nonstrict Hierarchies

- The mapping creates relations for representing the levels, and an additional relation (a **bridge table**) for representing the many-to-many relationship between them



- Bridge tables (e.g., EmplSection) represent many-to-many relationships
- If the parent-child relationship has a distributing attribute the bridge table will have an attribute to store its values

Nonstrict Hierarchies: Alternative Solution

- Transform a nonstrict hierarchy into a strict one, including an additional dimension in the fact
- Then, the mapping for a strict hierarchy can be applied
- The choice between the two solutions depends on:
  - **Data structure and size**: Bridge tables require less space than additional dimensions
  - **Performance and applications**: For bridge tables, join operations, calculations, and programming effort are needed to aggregate measures correctly; in additional dimensions, measures in the fact table ready for aggregation along the hierarchy

Table of Contents

1. Relational Data Warehouse Design

2. Relational Implementation of the Conceptual Model

3. Logical Representation of Hierarchies

4. Advanced Modeling Aspects

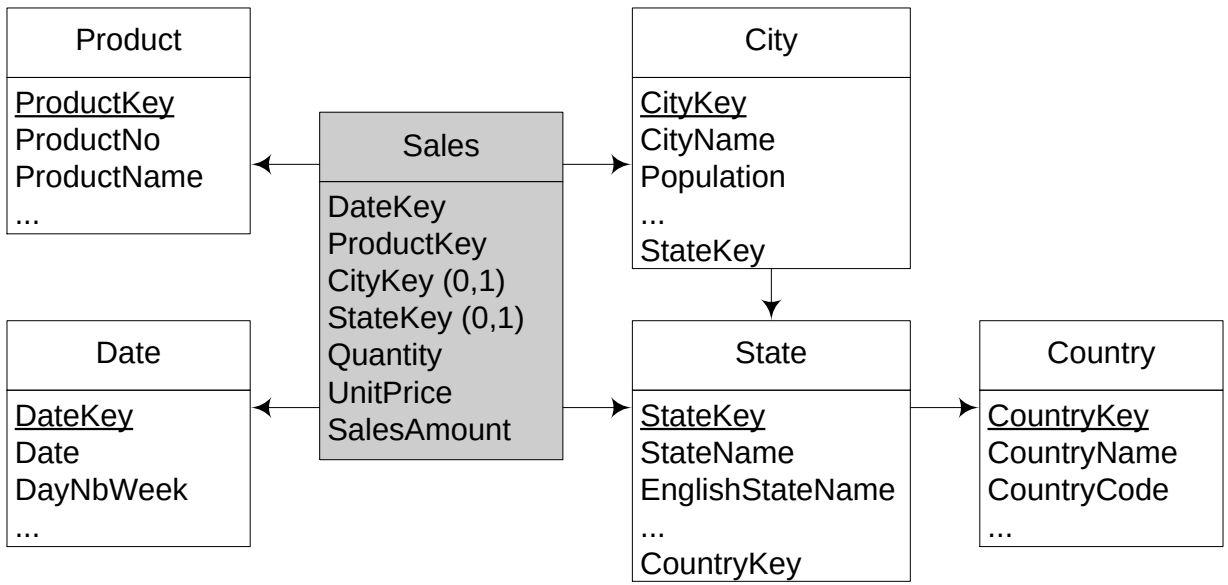
5. Slowly Changing Dimensions

6. SQL/OLAP Operations

7. The Northwind Cube in Analysis Services Multidimensional

Facts with Multiple Granularities

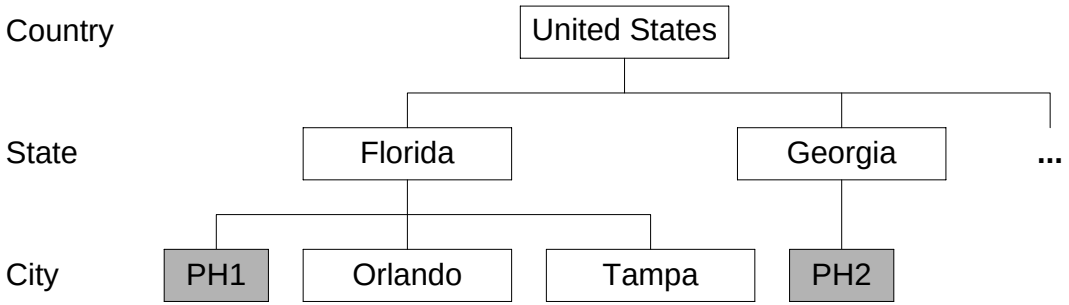
- **First approach**: Use multiple foreign keys, one for each alternative granularity, in a similar way as it done for generalized hierarchies



- Both attributes CityKey and StateKey are optional, triggers must be specified to ensure that only one of the foreign keys has a value

Facts with Multiple Granularities

- **Second approach**: Remove granularity variation at the instance level using placeholders, similarly as in unbalanced hierarchies



- Placeholders are used for facts that refer to nonleaf levels
- Two possible cases:
  - A fact member points to a nonleaf member that has children
    ⇒ PH1 represents all cities other than the existing children
  - A fact member points to a nonleaf member withouth children
    ⇒ PH2 represents all (unknown) cities of the state

Many-to-Many Dimensions

- Mapping rules create relations representing the fact, the dimension levels, and a bridge table representing the many-to-many relationship between **fact table and dimension**
- A bridge table BalanceClient relates the fact table Balance with the dimension table Client
- A surrogate key added to the Balance fact table to relate facts with clients.

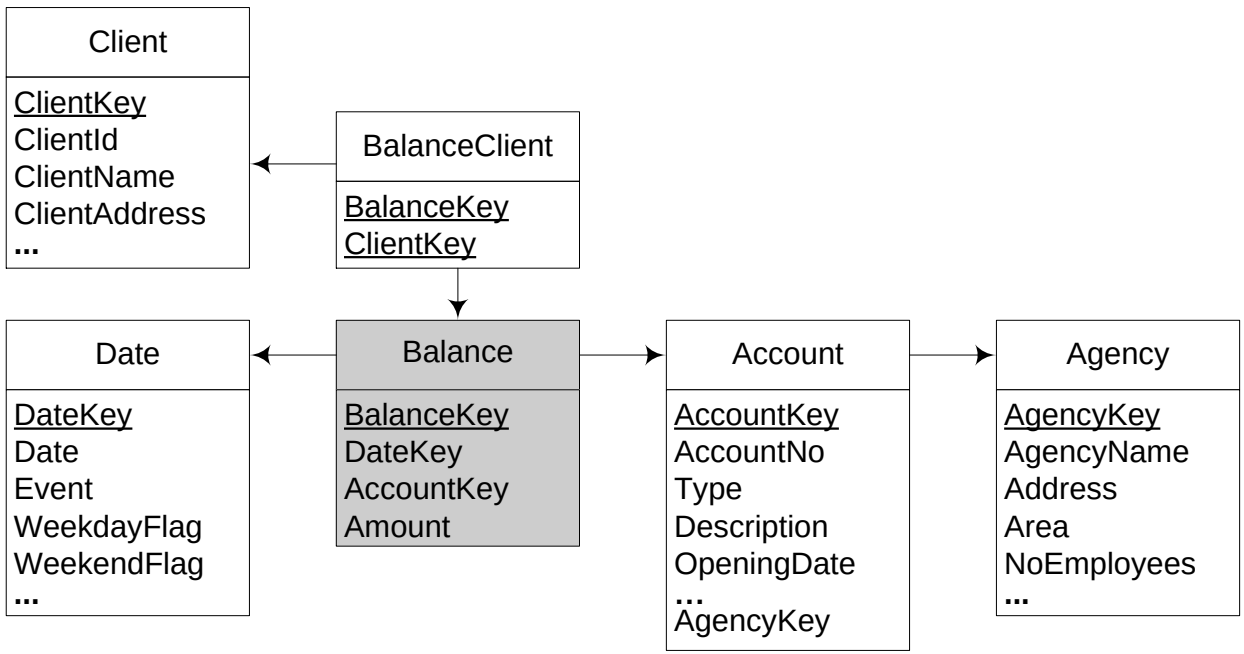


Table of Contents

1. Relational Data Warehouse Design

2. Relational Implementation of the Conceptual Model

3. Logical Representation of Hierarchies

4. Advanced Modeling Aspects

5. Slowly Changing Dimensions

6. SQL/OLAP Operations

7. The Northwind Cube in Analysis Services Multidimensional

Slowly Changing Dimensions

- In real-world situations, dimensions can change at the **schema** and **instance** level
- Example: at the schema level, when an attribute is deleted from the data sources and is no longer available it should also be deleted from the dimension table
- At the instance level two kinds of changes
  - A **correction** must be made to the dimension tables due to an error, the new data should replace the old one
  - When the **contextual conditions** of an analysis scenario change, the contents of dimension tables must change accordingly

Slowly Changing Dimensions

- Example: a Sales fact table related to the dimensions Time, Employee, Customer, and Product, and a SalesAmount measure; A star representation of table Product

DateKey	EmployeeKey	CustomerKey	ProductKey	SalesAmount
t1	e1	c1	p1	100
t2	e2	c2	p1	100
t3	e1	c3	p3	100
t4	e2	c4	p4	100

ProductKey	ProductName	Disc.	CategoryName	Description
p1	prod1	No	cat1	desc1
p2	prod2	No	cat1	desc1
p3	prod3	No	cat2	desc2
p4	prod4	No	cat2	desc2

- New tuples entered into the Sales fact table as new sales occur
- Other updates likely to occur:
  - A product starts to be commercialized → a new tuple in Product must be inserted
  - Data about a product may also be wrong, and must be corrected
  - The category of a product may need to be changed
- These dimensions are called **slowly changing dimensions**

Slowly Changing Dimensions

- **Query**: Total sales per employee and product category

```
SELECT    E.EmployeeKey, P.CategoryName, SUM(SalesAmount)
FROM      Sales S, Product P
WHERE     S.ProductKey = P.ProductKey
GROUP BY E.EmployeeKey, P.CategoryName
```

EmployeeKey	CategoryName	SalesAmount
e1	cat1	100
e2	cat1	100
e1	cat2	100
e2	cat2	100

- At instant *t* after t4 category of product p1 changes to cat2
- If we just overwrite the category the same query would return:

EmployeeKey	CategoryKey	SalesAmount
e1	cat2	200
e2	cat2	200

- **Incorrect result**: products affected by the category change were already associated with sales data
- If the new category is the result of an error correction (that is, the actual category of p1 is cat2), this result would be correct
- **Seven kinds** of slowly changing dimensions

Slowly Changing Dimensions: Type 1

- The simplest case
- **Overwrite** the old value of the attribute with the new one
- Assumes that the modification is due to an **error** in the dimension data
- In SQL

```
UPDATE Product
SET    CategoryName = cat2
WHERE  ProductName = p1
```

Slowly Changing Dimensions: Type 2

- Tuples in the dimension table are **versioned**: a new tuple is inserted each time a change occurs
- Tuples in the fact table match the tuple in the dimension table corresponding to the right version
- Example: Product extended with attributes From and To specifying the validity interval of the tuple
  - A row for p1 is inserted in Product, with its new category cat2
  - Sales prior to t will contribute to the aggregation of cat1, the ones occurred after t will contribute to cat2

Product Key	Product Name	Disc.	Category Name	Description	From	To
p1	prod1	No	cat1	desc1	2010-01-01	2011-12-31
p11	prod1	No	**cat2**	desc2	2012-01-01	Now
p2	prod2	No	cat1	desc1	2012-01-01	Now
p3	prod3	No	cat2	desc2	2012-01-01	Now
p4	prod4	No	cat2	desc2	2012-01-01	Now

- Now indicates that the tuple is still valid
- A product participates in the fact table with as many surrogates as there are attribute changes

Slowly Changing Dimensions: Type 3

- A column is added for each attribute subject to change, it holds the new value of the attribute
- Example: Both CategoryName and Description changed
  - product p1 changes category from c1 to c2
  - the associated description also changes from desc1 todesc2

Product Key	Product Name	Disc.	Category Name	NewCateg	Description	NewDesc
p1	prod1	No	**cat1**	**cat2**	**desc1**	**desc2**
p2	prod2	No	cat1	Null	desc1	Null
p3	prod3	No	cat2	Null	desc2	Null
p4	prod4	No	cat2	Null	desc2	Null

- Only the two more recent versions of the attributes can be represented
- The validity interval of the tuples is not stored

Slowly Changing Dimensions: Type 4

- Aims at handling very large dimension tables and attributes that change frequently
- A **minidimension** is created to store the most frequently changing attributes
  - Example: In the Product dimension attributes SalesRanking and PriceRange change frequently
  - A new dimension ProductFeatures is created, with key ProductFeaturesKey, and attributes SalesRanking and PriceRange

Product FeaturesKey	Sales Ranking	Price Range
pf1	1	1-100
pf2	2	1-100
...	...	...
pf200	7	500-600

- There is a row in the minidimension for each unique combination of SalesRanking and PriceRange in the data

Slowly Changing Dimensions: Type 5

- An extension of Type 4, where the primary dimension table is extended with a foreign key to the minidimension table
- The Product dimension:

Product Key	Product Name	Disc.	CurrentProduct FeaturesKey
p1	prod1	No	pf1
...	...	...	...

- Foreign key is a Type 1 attribute: when any feature of the product changes, the current ProductFeaturesKey value is stored in the Product table
- CurrentProductFeaturesKey in the Product dimension allows rolling up historical facts based on the current product profile

Slowly Changing Dimensions: Type 6

- Extends a Type 2 dimension with an additional column containing the current value of an attribute
  - Example: Product dimension extended with attributes From and To
  - CurrentCategoryKey contains the current value of the Category attribute

Product Key	Product Name	Disc.	Category Key	From	To	Current CategoryKey
p1	prod1	No	c1	2010-01-01	2011-12-31	**c11**
p11	prod1	No	**c11**	2012-01-01	9999-12-31	**c11**
p2	prod2	No	c1	2010-01-01	9999-12-31	c1
p3	prod3	No	c2	2010-01-01	9999-12-31	c2
p4	prod4	No	c2	2011-01-01	9999-12-31	c2

- CategoryKey attribute used to group facts based on the product category effective when facts occurred
- CurrentCategoryKey attribute groups facts based on the current category

Slowly Changing Dimensions: Type 7

- Similar to the Type 6, when there are many attributes in the dimension table
- Adds an foreign key of the dimension table with the natural (not surrogate) key (ProductName in our example) if it is **durable**
- Example: Product dimension the same as in Type 2, but the fact table looks:

DateKey	Employee Key	Customer Key	Product Key	Product Name	SalesAmount
d1	e1	c1	p1	prod1	100
d2	e2	c2	p11	prod1	100
d3	e1	c3	p3	prod3	100
d4	e2	c4	p4	prod4	100

- ProductKey can be used for historical analysis based on the product values effective when the fact occurred
- An additional view CurrentProduct to support current analysis: it keeps only current values of the Product dimension:

Product Name	Disc.	Category Key
prod1	No	c2
prod2	No	c1
prod3	No	c2
prod4	No	c2

Slowly Changing Dimensions in a Snowflake Representation

- Handled in similar way as above
- Consider a snowflake representation for the Product dimension

Product Key	Product Name	Disc.	Category Key
p1	prod1	No	c1
p2	prod2	No	c1
p3	prod3	No	c2
p4	prod4	No	c2

Category Key	Category Name	Description
c1	cat1	desc1
c2	cat2	desc2
c3	cat3	desc3
c4	cat4	desc4

- When product p1 changes its category to c2, in a Type-2 solution, we add two temporal attributes to the Product table

Product Key	Product Name	Disc.	Category Key	From	To
p1	prod1	No	c1	2010-01-01	2011-12-31
p11	prod11	No	**c2**	2012-01-01	Now
p2	prod2	No	c1	2010-01-01	Now
p3	prod3	No	c2	2010-01-01	Now
p4	prod4	No	c2	2011-01-01	Now

- The Category table remains unchanged

Slowly Changing Dimensions in a Snowflake Representation

- If change occurs at an upper level in the hierarchy, for example, a description is changed, it must be propagated downward
- Example: the description of category cat1 changes:

Category Key	Category Name	Description	From	To
c1	cat1	desc1	2010-01-01	2011-12-31
c11	cat1	**desc11**	2012-01-01	Now
c2	cat2	desc2	2012-01-01	Now
c3	cat3	desc3	2010-01-01	Now
c4	cat4	desc4	2010-01-01	Now

- This change must be propagated to the Product table:

Product Key	Product Name	Disc.	Category Key	From	To
p1	prod1	No	c1	2010-01-01	2011-12-31
p11	prod1	No	**c11**	2012-01-01	Now
p2	prod2	No	c1	2010-01-01	Now
p3	prod3	No	c2	2010-01-01	Now
p4	prod4	No	c2	2011-01-01	Now

Table of Contents

1. Relational Data Warehouse Design

2. Relational Implementation of the Conceptual Model

3. Logical Representation of Hierarchies

4. Advanced Modeling Aspects

5. Slowly Changing Dimensions

6. SQL/OLAP Operations

7. The Northwind Cube in Analysis Services Multidimensional

The Data Cube in the Relational Model

- A relational table is not the best structure for multidimensional data
- Consider a cube Sales, with dimensions Product and Customer, and a measure SalesAmount
- The data cube contains all possible (2^2) aggregations of the cube cells: SalesAmount by Product, by Customer, and by both Product and Customer, plus the base nonaggregated data

	c1	c2	c3	TotalBy Product
p1	100	105	100	305
p2	70	60	40	170
p3	30	40	50	120
TotalBy Customer	200	205	190	595

- A relational fact table representing the same data

ProductKey	CustomerKey	SalesAmount
p1	c1	100
p1	c2	105
p1	c3	100
p2	c1	70
p2	c2	60
p2	c3	40
p3	c1	30
p3	c2	40
p3	c3	50

The Data Cube in the Relational Model

- Consider the Sales fact table
- To compute all possible aggregations along Product and Customer we must scan the whole relation
- Computed in SQL using NULL value:

```
SELECT   ProductKey, CustomerKey, SalesAmount
FROM     Sales
    UNION
SELECT   ProductKey, NULL, SUM(SalesAmount)
FROM     Sales
GROUP BY ProductKey
    UNION
SELECT   NULL, CustomerKey, SUM(SalesAmount)
FROM     Sales
GROUP BY CustomerKey
    UNION
SELECT   NULL, NULL, SUM(SalesAmount)
FROM     Sales
```

Data cube

ProductKey	CustomerKey	SalesAmount
p1	c1	100
p2	c1	70
p3	c1	30
NULL	c1	200
p1	c2	105
p2	c2	60
p3	c2	40
NULL	c2	205
p1	c3	100
p2	c3	40
p3	c3	50
NULL	c3	190
p1	NULL	305
p2	NULL	170
p3	NULL	120
NULL	NULL	595

SQL/OLAP Operations

- Computing a cube with n dimensions requires 2^n GROUP BY
- SQL/OLAP extends the GROUP BY clause with the ROLLUP and CUBE operators
- ROLLUP computes group subtotals in the order given by a list of attributes
- CUBE computes all totals of such a list
- Shorthands for a more powerful operator, GROUPING SETS
- Equivalent queries

```
SELECT    ProductKey, CustomerKey, SUM(SalesAmount)
FROM      Sales
GROUP BY ROLLUP(ProductKey, CustomerKey)
```

```
SELECT    ProductKey, CustomerKey, SUM(SalesAmount)
FROM      Sales
GROUP BY GROUPING SETS((ProductKey,CustomerKey),(ProductKey),())
```

- Equivalent queries

```
SELECT    ProductKey, CustomerKey, SUM(SalesAmount)
FROM      Sales
GROUP BY CUBE(ProductKey, CustomerKey)
```

```
SELECT    ProductKey, CustomerKey, SUM(SalesAmount)
FROM      Sales
GROUP BY GROUPING SETS((ProductKey, CustomerKey),(ProductKey),(CustomerKey),())
```

SQL/OLAP Operations

GROUP BY ROLLUP

ProductKey	CustomerKey	SalesAmount
p1	c1	100
p1	c2	105
p1	c3	100
p1	NULL	305
p2	c1	70
p2	c2	60
p2	c3	40
p2	NULL	170
p3	c1	30
p3	c2	40
p3	c3	50
p3	NULL	120
NULL	NULL	595

GROUP BY CUBE

ProductKey	CustomerKey	SalesAmount
p1	c1	100
p2	c1	70
p3	c1	30
NULL	c1	200
p1	c2	105
p2	c2	60
p3	c2	40
NULL	c2	205
p1	c3	100
p2	c3	40
p3	c3	50
NULL	c3	190
NULL	NULL	595
p1	NULL	305
p2	NULL	170
p3	NULL	120

SQL/OLAP Operations: Window Partitioning

- Allows detailed data to be compared with aggregate values
- Example: relevance of each customer with respect to the product sales

```
SELECT ProductKey, CustomerKey, SalesAmount,
       MAX(SalesAmount) OVER (PARTITION BY ProductKey) AS MaxAmount
FROM   Sales
```

- First three columns are obtained from the Sales table
- The fourth column:
  - For each tuple define a **window** having all tuples of the same product
  - SalesAmount is aggregated over this window using the MAX function

ProductKey	CustomerKey	SalesAmount	MaxAmount
p1	c1	100	105
p1	c2	105	105
p1	c3	100	105
p2	c1	70	70
p2	c2	60	70
p2	c3	40	70
p3	c1	30	50
p3	c2	40	50
p3	c3	50	50

SQL/OLAP Operations: Window Ordering

- Allows the rows within a partition to be ordered
- Useful to compute rankings, with functions ROW_NUMBER and RANK
- Example: How does each product rank in the sales of each customer

```
SELECT ProductKey, CustomerKey, SalesAmount, ROW_NUMBER() OVER
       (PARTITION BY CustomerKey ORDER BY SalesAmount DESC) AS RowNo
FROM   Sales
```

- First tuple evaluated by opening a window with all tuples of customer c1, ordered by the sales amount
- Product p1 is the one most demanded by customer c1

Product Key	Customer Key	Sales Amount	RowNo
p1	c1	100	1
p2	c1	70	2
p3	c1	30	3
p1	c2	105	1
p2	c2	60	2
p3	c2	40	3
p1	c3	100	1
p3	c3	50	2
p2	c3	40	3

SQL/OLAP Operations: Window Framing

- Defines the size of the partition
- Used to compute statistical functions over time series, like moving average
- Example: Three-month moving average of sales by product

```
SELECT ProductKey, Year, Month, SalesAmount, AVG(SalesAmount) OVER
       (PARTITION BY ProductKey ORDER BY Year, Month ROWS 2 PRECEDING)
       AS MovAvg3M
FROM   Sales
```

- For each tuple, the window contains the tuples having the same product ordered by year and month
- The average over the current tuple and the previous two ones (if they exist) is computed

Product Key	Year	Month	Sales Amount	MovAvg3M
p1	2011	10	100	100
p1	2011	11	105	102.5
p1	2011	12	100	101.67
p2	2011	12	60	60
p2	2012	1	40	50
p2	2012	2	70	56.67
p3	2012	1	30	30
p3	2012	2	50	40
p3	2012	3	40	40

SQL/OLAP Operations: Window Framing

- Defines the size of the partition
- Used to compute statistical functions over time series, like moving average
- Example: Year-to-date sum of sales by product

```
SELECT ProductKey, Year, Month, SalesAmount, AVG(SalesAmount) OVER (PARTITION BY
       ProductKey, Year ORDER BY Month ROWS UNBOUNDED PRECEDING) AS YTD
FROM   Sales
```

- For each tuple, the window contains the tuples of the current product and year ordered by month
- SUM is applied to all the tuples before the current tuple (ROWS UNBOUNDED PRECEDING)

Product Key	Year	Month	Sales Amount	YTD
p1	2011	10	100	100
p1	2011	11	105	205
p1	2011	12	100	305
p2	2011	12	60	60
p2	2012	1	40	40
p2	2012	2	70	110
p3	2012	1	30	30
p3	2012	2	50	80
p3	2012	3	40	120

Table of Contents

1. Relational Data Warehouse Design

2. Relational Implementation of the Conceptual Model

3. Logical Representation of Hierarchies

4. Advanced Modeling Aspects

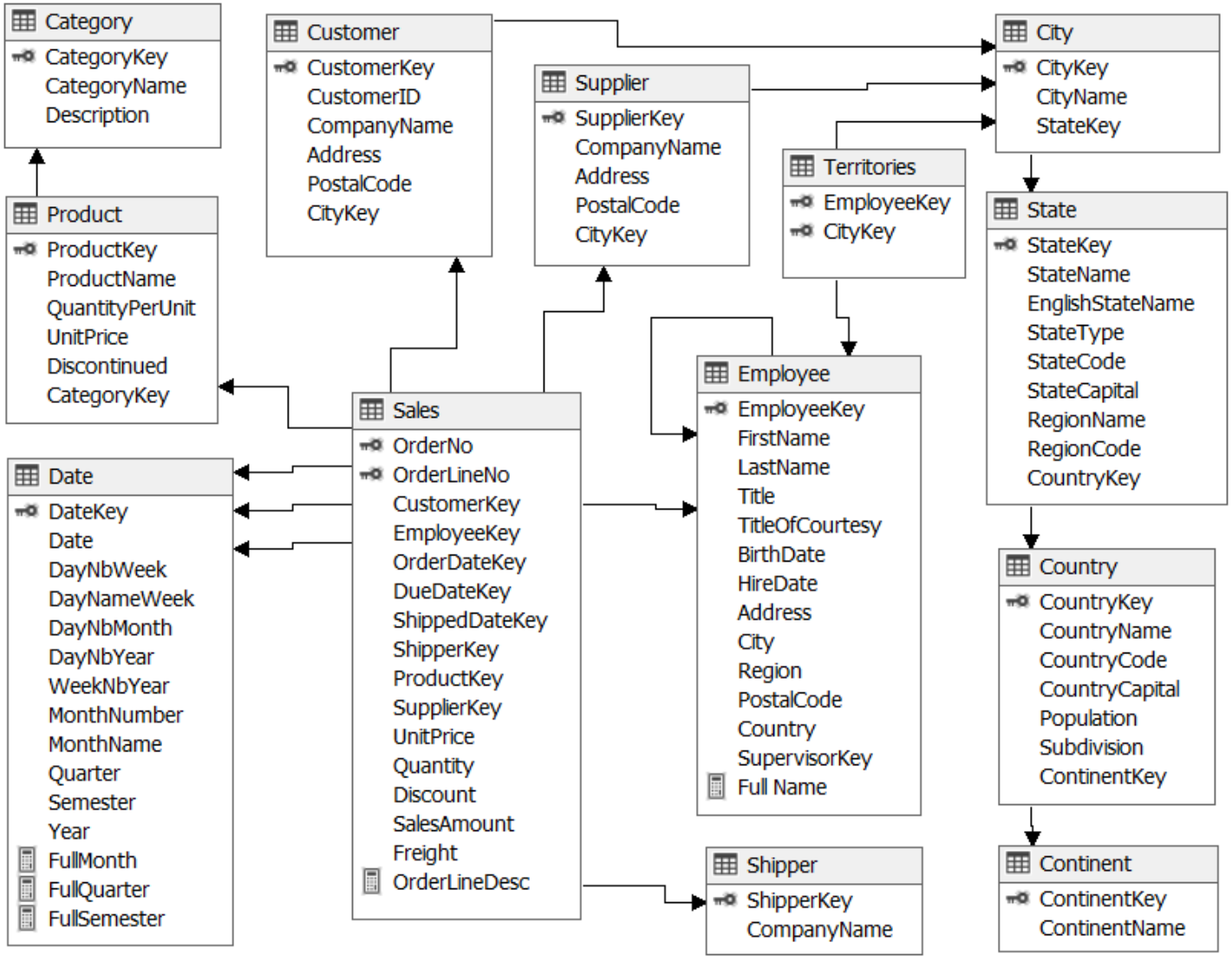
5. Slowly Changing Dimensions

6. SQL/OLAP Operations

7. The Northwind Cube in Analysis Services Multidimensional

The Northwind Cube in Analysis Services: Data Source View

- Defines the schema used for populating an Analysis Services database
- Schema is derived from the schemas of the various data sources

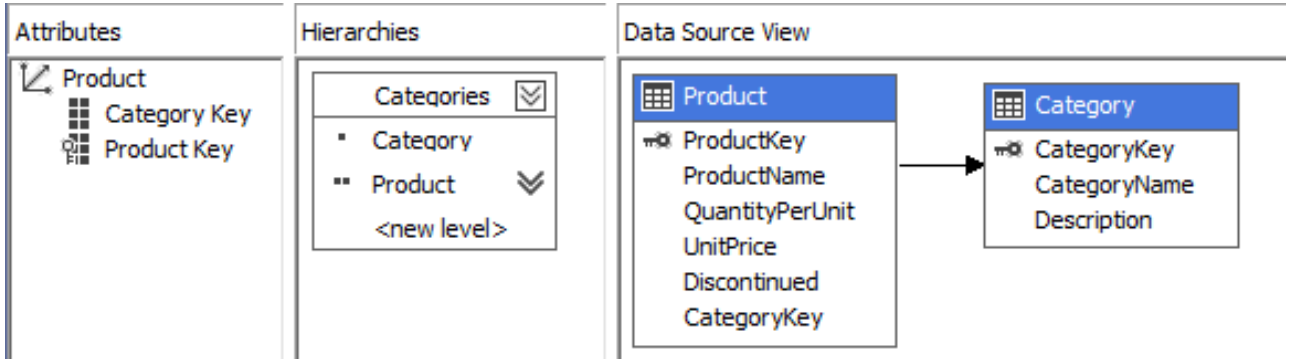


The Northwind Cube in Analysis Services: Dimensions

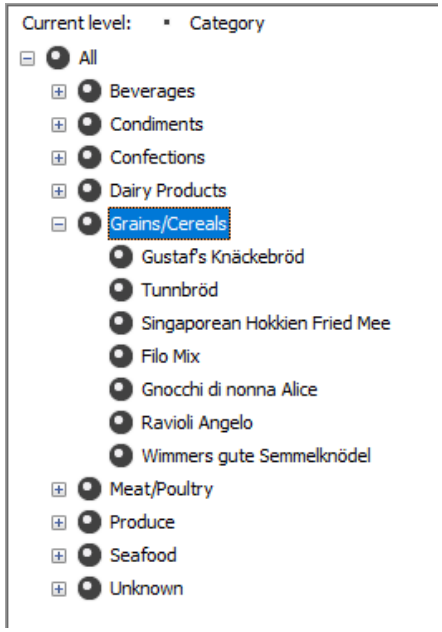
- **Regular dimension**: Has a direct one-to-many link between a fact table and a dimension table
- **Reference dimension**: Indirectly related to the fact table through another dimension
  - Example: Geography dimension, related to the Sales fact table through the Customer and Supplier dimensions
- **Role-playing dimension**: A single fact table is related to a dimension table more than once
  - Example: Dimensions OrderDate, DueDate, and ShippedDate, which all refer to the Time dimension
- **Fact dimension**: Also called **degenerate dimension**, similar to a regular dimension but data are stored in the fact table (e.g., dimension Order)
- **Many-to-many dimension**: A fact is related to multiple dimension members and a member is related to multiple facts beginitemize
- Example: Relationship between Employees and Cities, which is represented in the bridge table Territories. This table must be defined as a fact table in Analysis Services

The Northwind Cube in Analysis Services: Regular Dimensions

- Definition of dimension Product

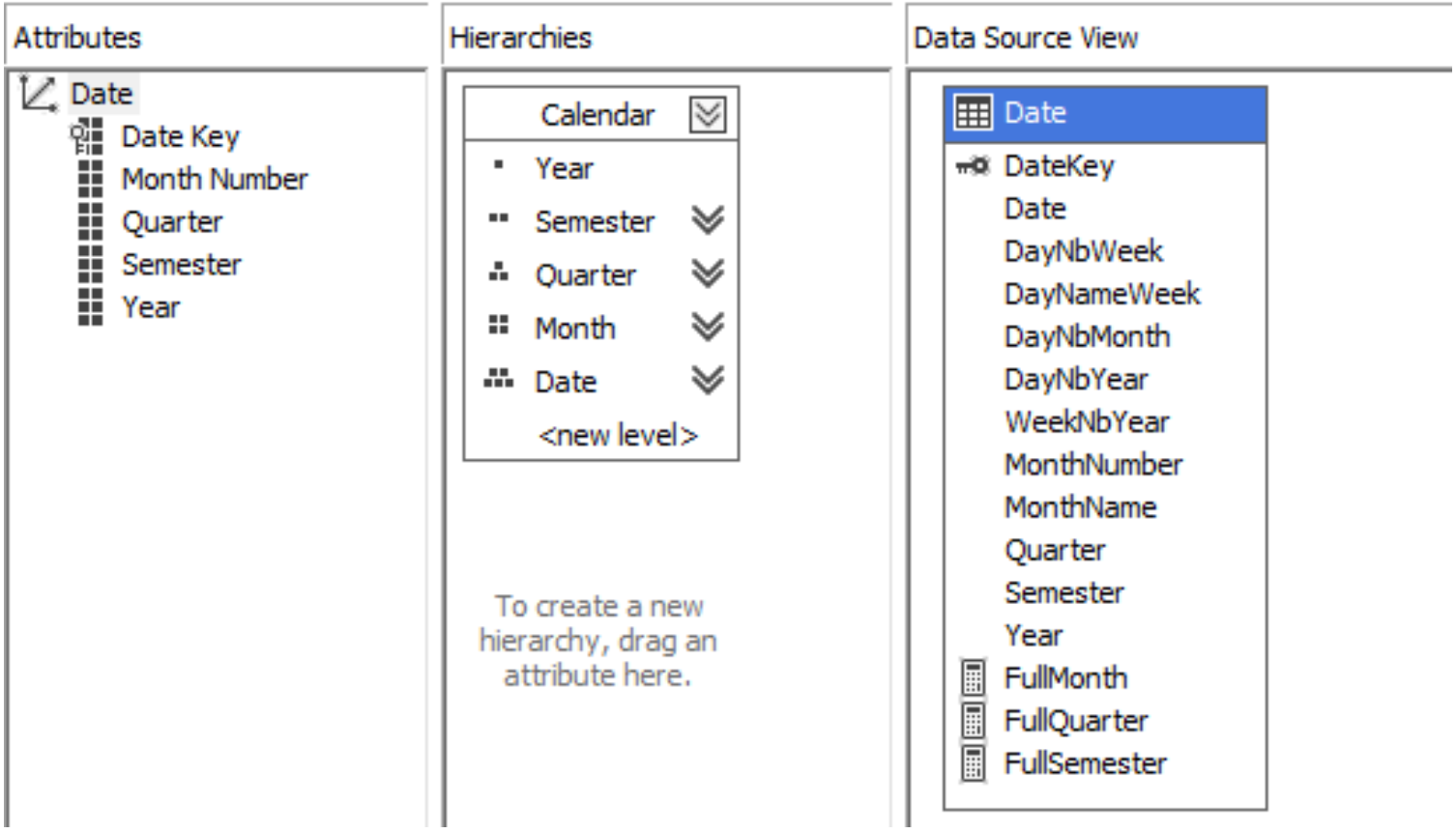


- Browsing the hierarchy of the Product dimension



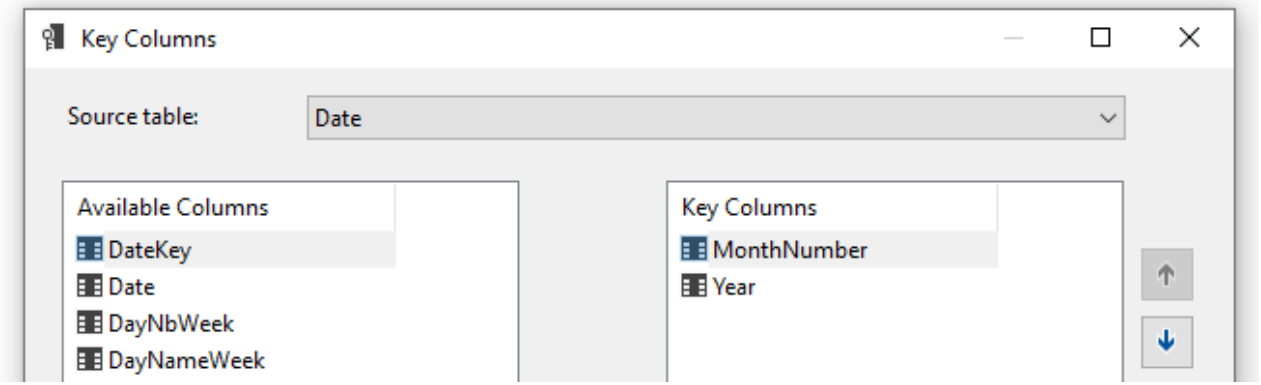
The Northwind Cube in Analysis Services: Time Dimensions

- Type property of the dimension must be set to Time
- Used to identify attributes that correspond to the typical subdivision of time
- DayNbMonth of type DayOfMonth, MonthNumber type MonthOfYear, etc.



The Northwind Cube in Analysis Services: Key Columns

- Attributes in hierarchies must have a one-to-many relationship to their parents
  - Example: A quarter must roll-up to its semester
- In Analysis Services this is stated defining a key for each attribute in a hierarchy
- In Northwind, MonthNumber has values such as 1, 2, etc. → same value in several quarters
- Key of the attribute: a combination of MonthNumber and Year
- Done by defining KeyColumns property of the attribute

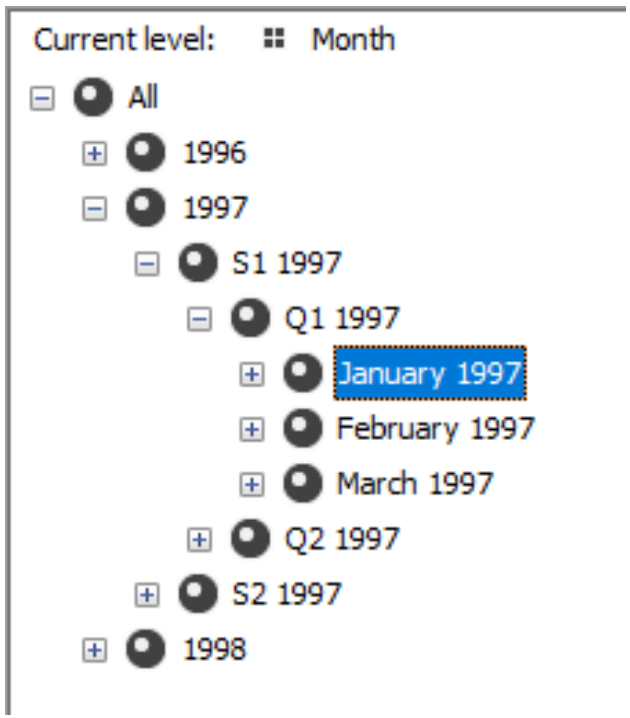


The Northwind Cube in Analysis Services: Relationships

- When creating a user-defined hierarchy: need to define relationships between the attributes
- Two types of relationships
  - **Flexible relationships** can evolve in time (e.g., a product can be assigned to a new category)
  - **Rigid relationships** cannot (e.g., a month always related to its year)
- Relationships for the Time dimension

The Northwind Cube in Analysis Services: Browsing Hierarchies

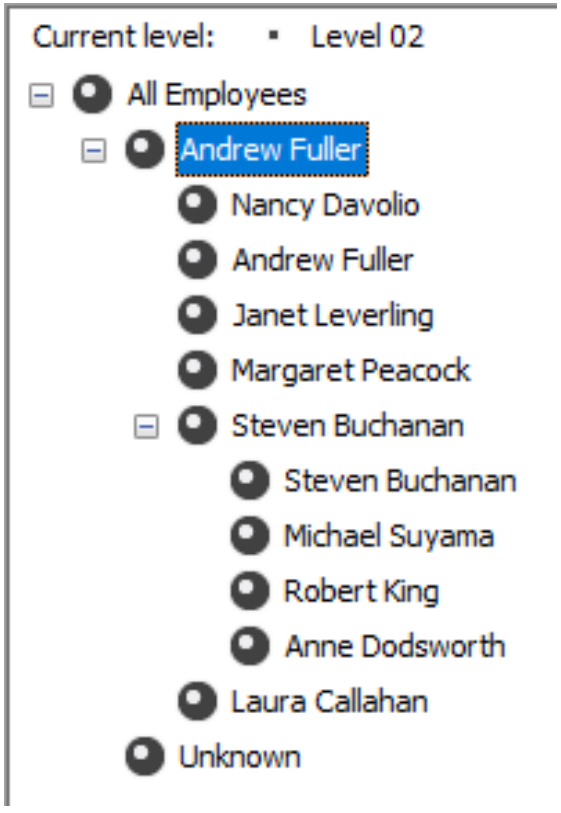
- Members of the Calendar hierarchy



- Named calculations FullSemester, FullQuarter, and FullMonth displayed when browsing

The Northwind Cube in Analysis Services: Browsing Parent-Child Hierarchies

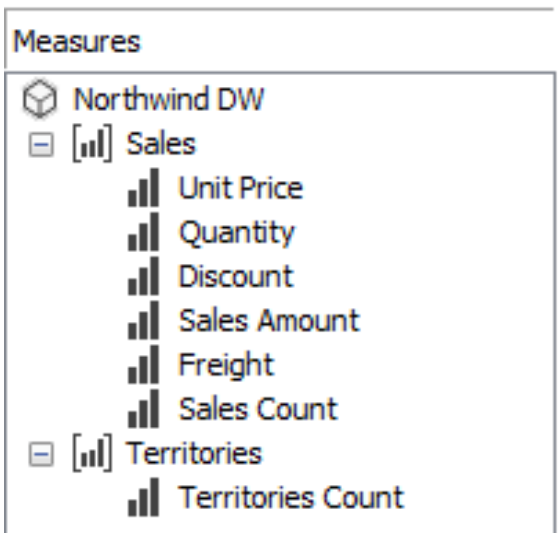
- Example: the Supervision hierarchy in the Employee dimension
- Column SupervisorKey: foreign key referencing EmployeeKey
- Usage property how attributes will be used
- Value of Usage: Parent for the SupervisorKey attribute, Regular for all other ones except EmployeeKey, Key for EmployeeKey



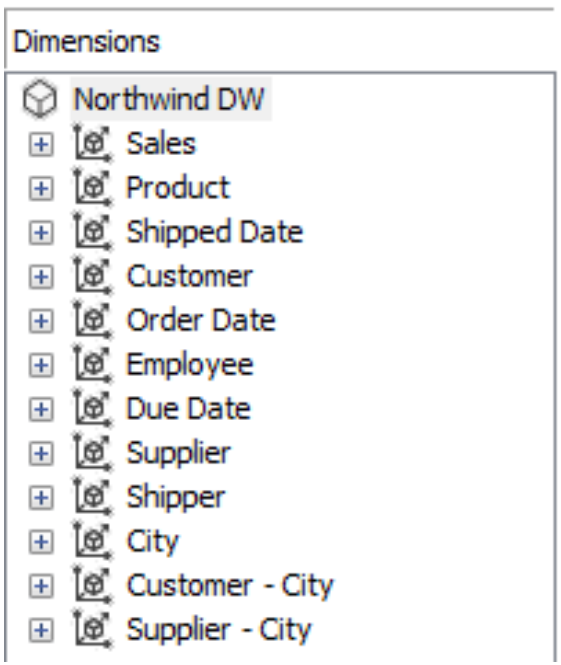
The Northwind Cube in Analysis Services: Cubes

- Cube built from one or several data source views
- Cube consists of one or more dimensions from dimension tables and one or more measure groups
- Facts in a fact table are mapped as measures in a cube

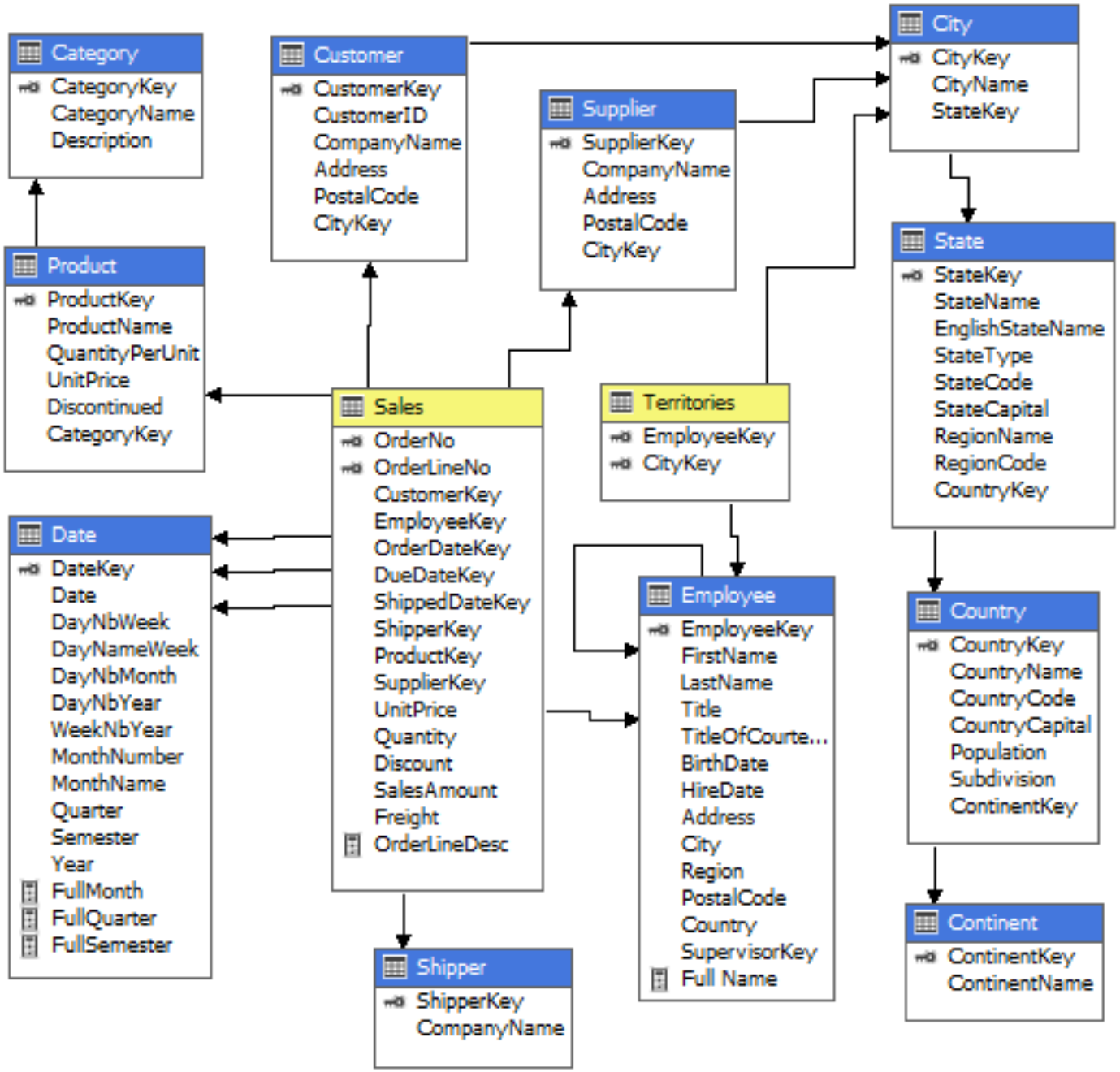
Measure groups



Dimensions

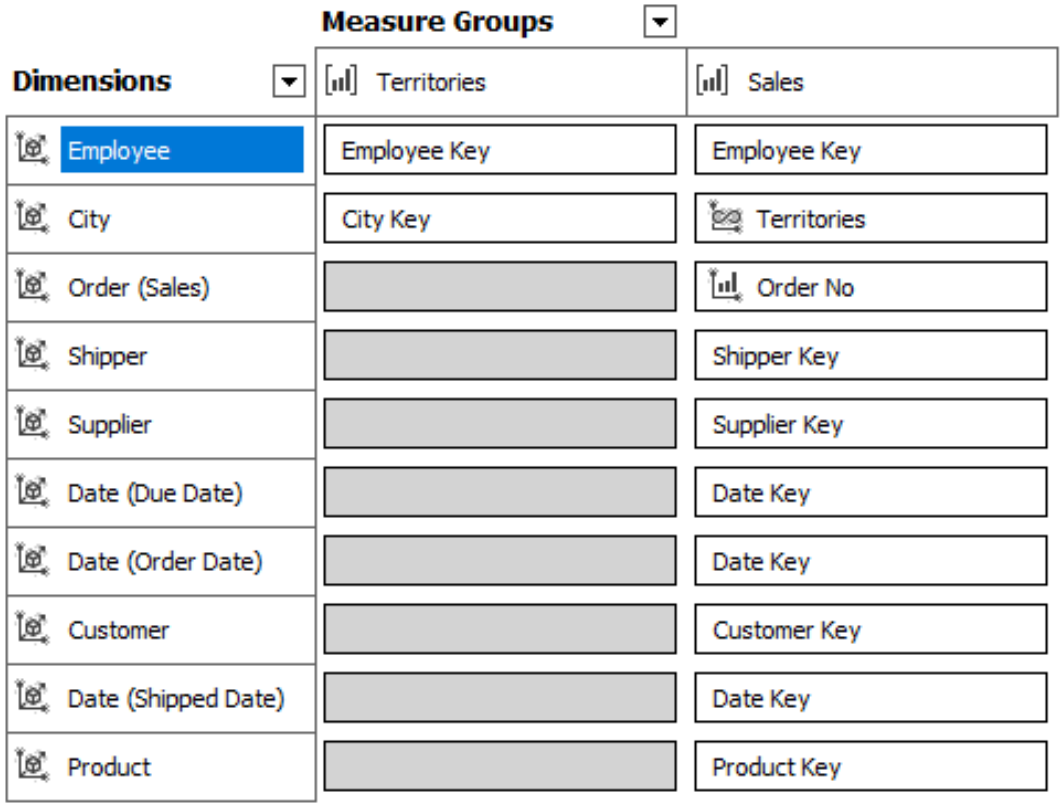


The Northwind Cube in Analysis Services: Cube Definition



The Northwind Cube in Analysis Services: Cube Definition

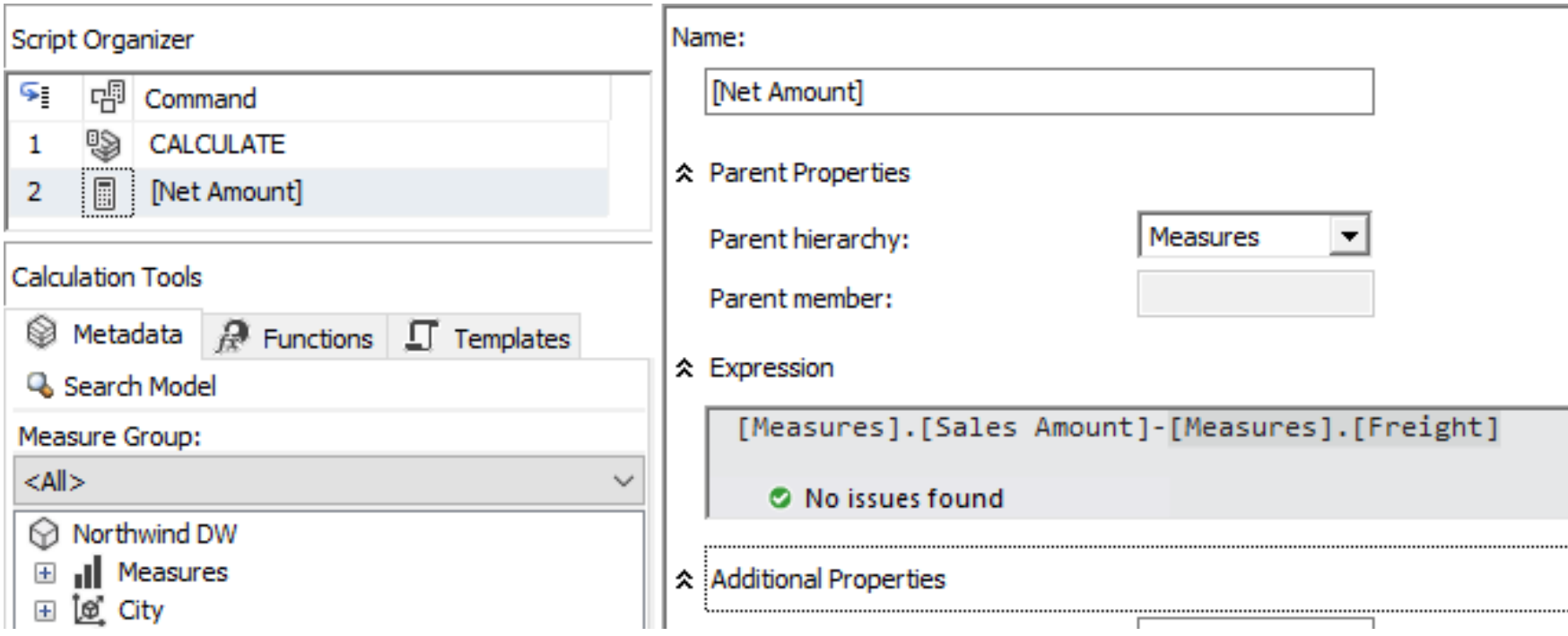
- Relationships between dimensions and measure groups in the cube



- With respect to the Sales measure group, all dimensions but the last two are regular
- Geography: many-to-many dimension linked to the measure group through the Territories fact table
- Order is a fact dimension

The Northwind Cube in Analysis Services: Cube Definition

- We can define the default measure of the cube, Sales Amount, used by default by MDX
- Derived measure Net Amount defined
- Measure will be a calculated member in the Measures dimension
- Expression is the difference between the Sales Amount and the Freight measures



Browsing the Northwind Cube in Excel

Total Sales	Column Labels			
Row Labels	2016	2017	2018	Grand Total
Europe	$113.290,89	$351.142,69	$219.090,18	$683.523,76
North America				
Canada	$7.283,08	$30.589,26	$9.539,48	$47.411,82
Mexico	$4.687,90	$12.700,83	$3.734,90	$21.123,63
United States				
Alaska	$4.675,80	$3.951,37	$4.792,89	$13.420,06
California				
San Francisco				
Let's Stop N Shop		$1.698,40	$1.378,07	$3.076,47
Idaho	$10.338,26	$56.241,08	$35.674,51	$102.253,85
Montana		$1.426,74	$326,00	$1.752,74
New Mexico	$9.923,78	$19.383,75	$19.982,55	$49.290,08
Oregon	$1.828,00	$15.150,98	$11.011,14	$27.990,11
Washington	$2.938,20	$10.810,46	$15.516,80	$29.265,46
Wyoming	$7.849,63	$2.475,00	$1.117,00	$11.441,63
South America	$29.034,32	$64.629,06	$60.942,88	$154.606,26
Grand Total	$191.849,87	$570.199,61	$383.106,38	$1.145.155,86