Chapter 3: Logical Data Warehouse Design

Data Warehouse

3.1 Logical Modeling of Data Warehouses

- Relational OLAP (ROLAP), which stores data in relational databases and supports extensions to SQL and special access methods to efficiently implement the multidimensional data model and the related operations.
- Multidimensional OLAP (MOLAP), which stores data in specialized multidimensional data structures and implements the OLAP operations over those data structures.
- Hybrid OLAP (HOLAP), which combines both approaches.

3.2 Relational Data Warehouse Design

- The star schema, where there is one central fact table, and a set of dimension tables, one for each dimension.
- In a star schema, the dimension tables are, in general, not normalized.
- Fact tables are usually normalized: their key is the union of the foreign keys since this union functionally determines all the measures, while there is no functional dependency between the foreign key attributes.

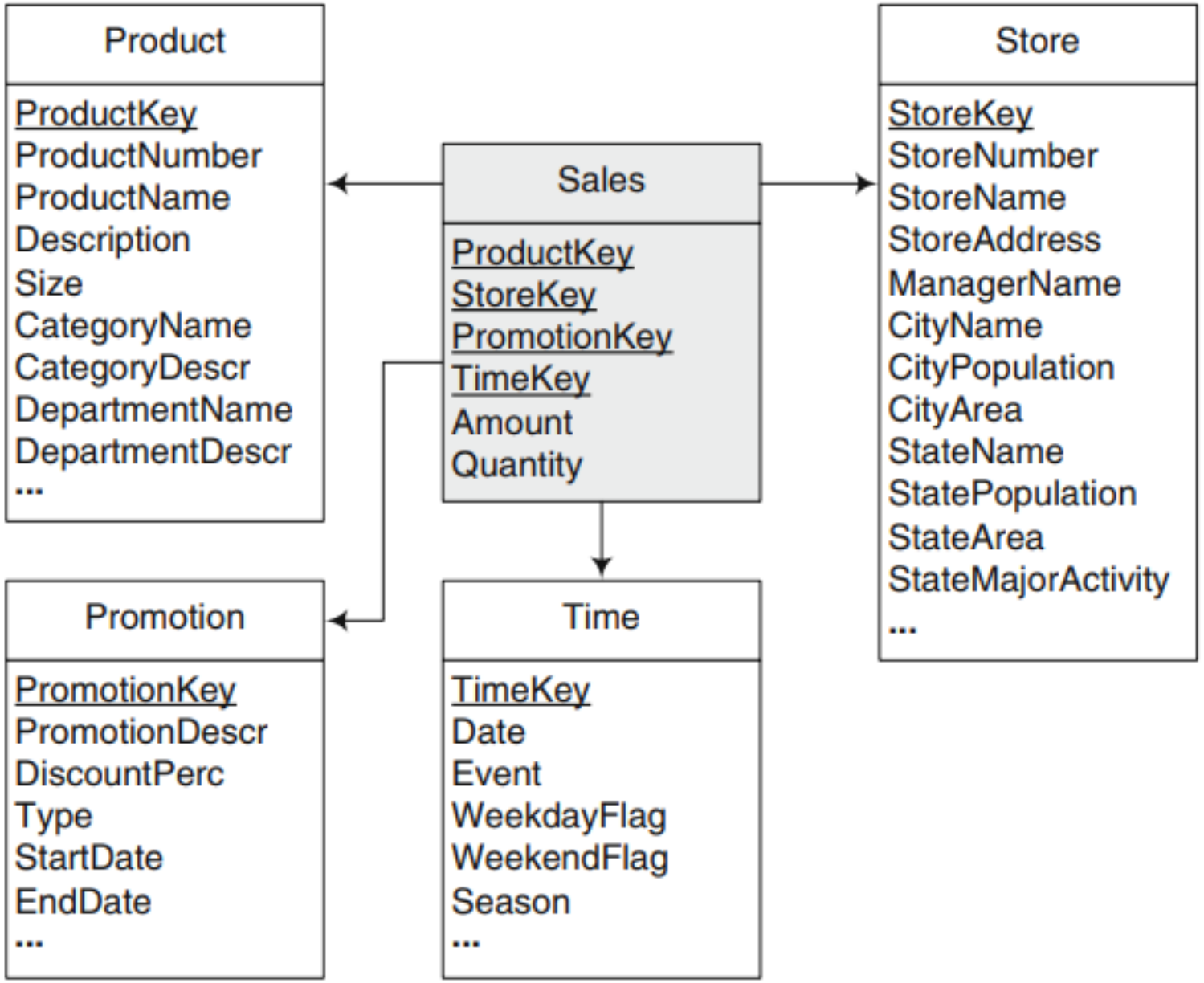


Fig. 5.1 An example of a star schema

- A snowflake schema avoids the redundancy of star schemas by normalizing the dimension tables.

Therefore, a dimension is represented by several tables related by referential integrity constraints.

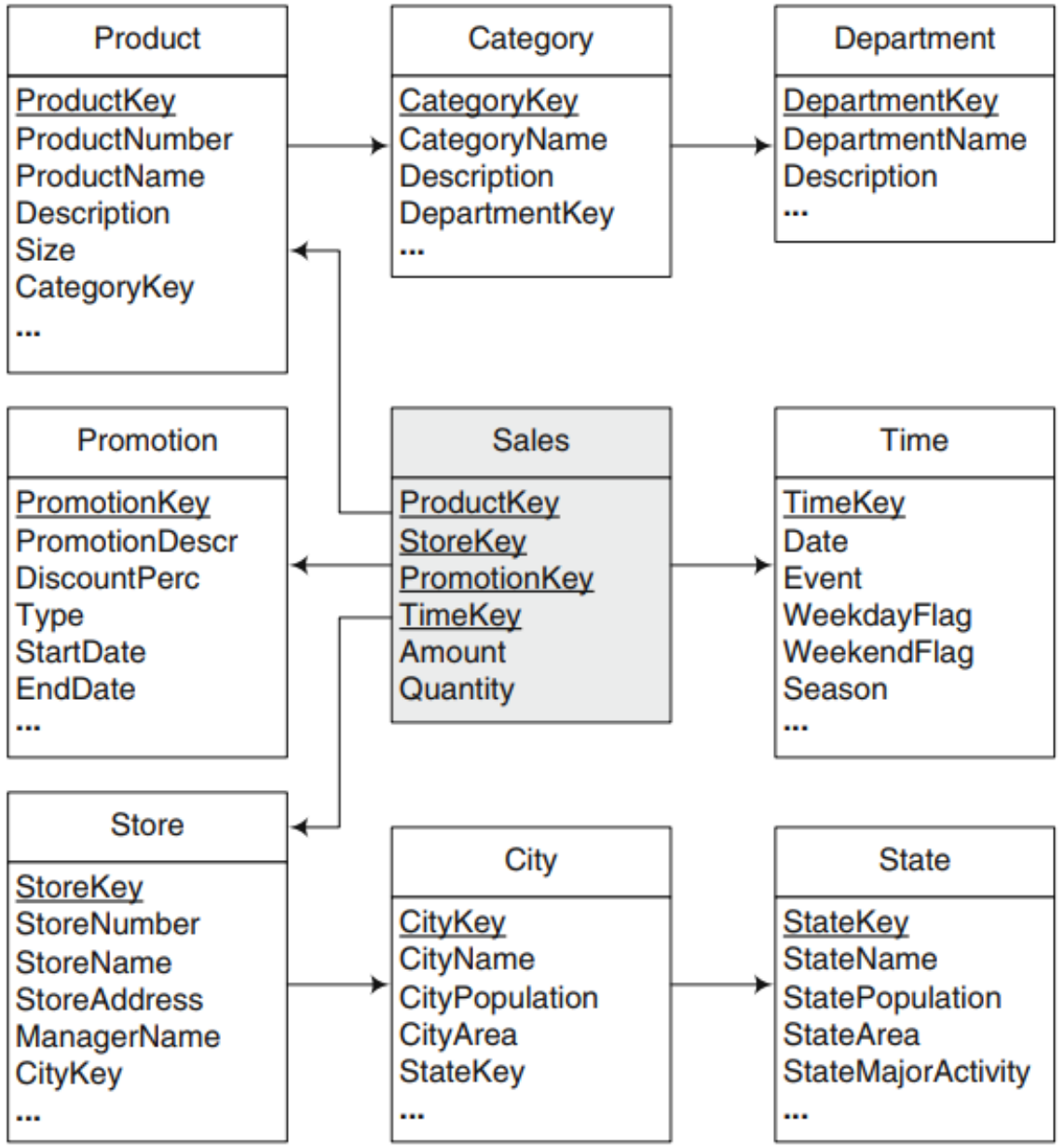


Fig. 5.2 An example of a snowflake schema

Fig. 5.1 An example of a star schema

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

Fig. 5.2 An example of a snowflake schema

```
SELECT CategoryName, SUM(Amount)
FROM Product P, Category C, Sales S
WHERE P.ProductKey = S.ProductKey AND P.CategoryKey = C.CategoryKey
GROUP BY CategoryName
```

- A starflake schema is a combination of the star and the snowflake schemas, where some dimensions are normalized while others are not.
- A constellation schema has multiple fact tables that share dimension tables.

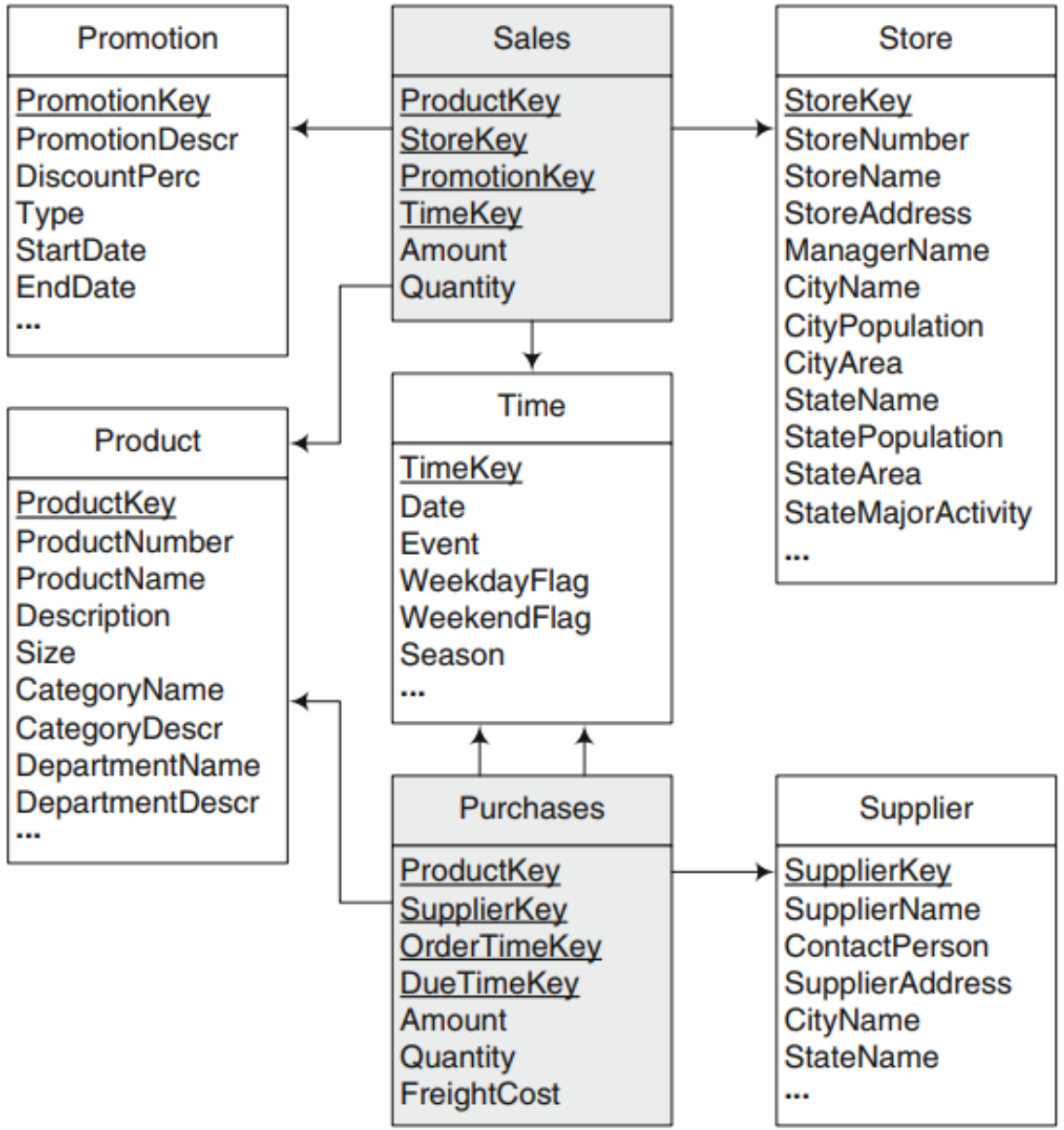


Fig. 5.3 An example of a constellation schema

3.3 Relational Implementation of the Conceptual Model

3.3 Relational Implementation of the Conceptual Model

- Surrogate keys are generated for each dimension level in a data warehouse.
- The main reason for this is to provide independence from the keys of the underlying source systems because such keys can change across time.
- Surrogate keys are usually represented as integers in order to increase efficiency, whereas keys from source systems may be represented in less efficient data types such as strings. Nevertheless, the keys coming from the source systems should also be kept in the dimensions to be able to match data from sources with data in the warehouse.

3.3 Relational Implementation of the Conceptual Model

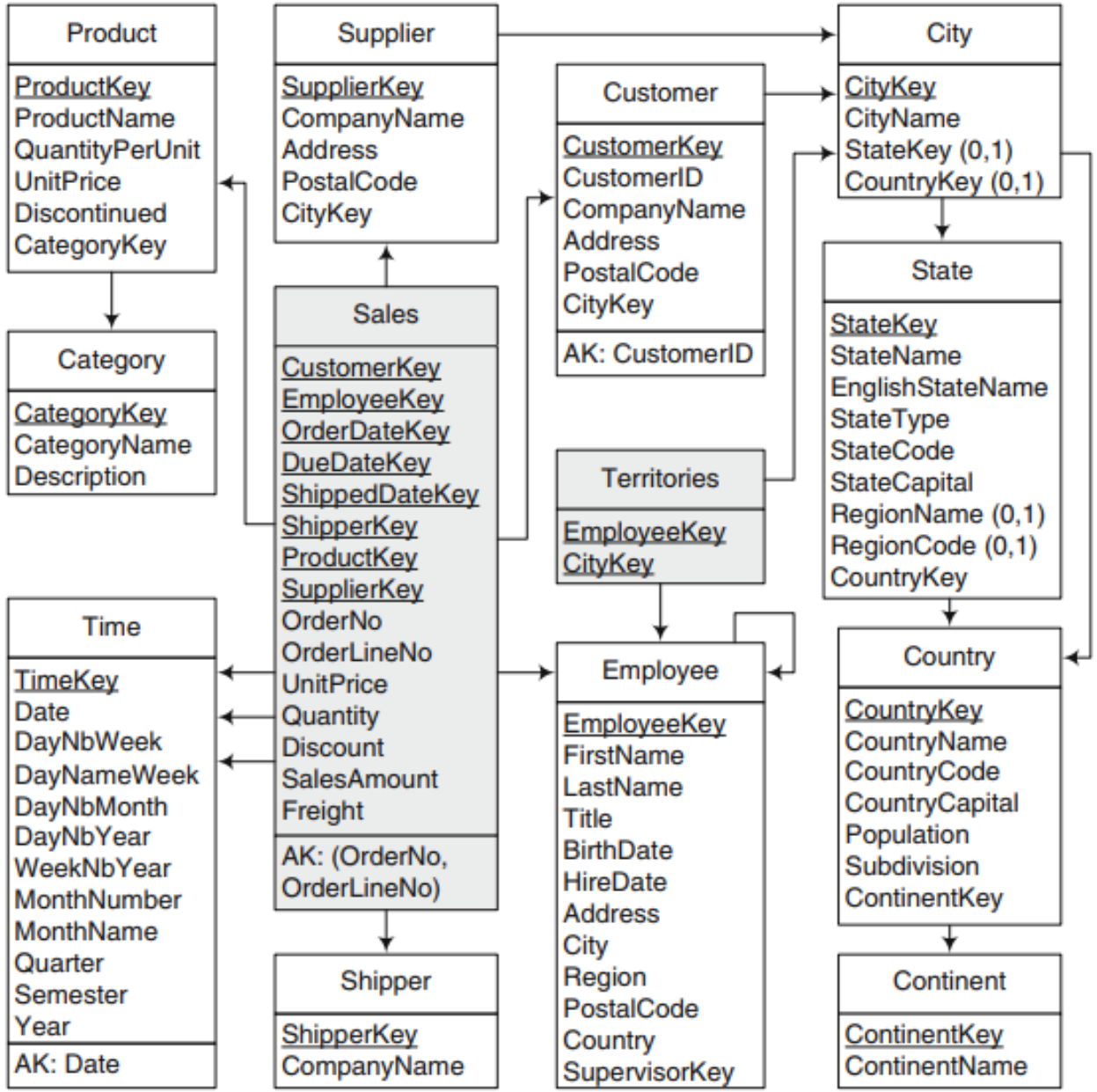


Fig. 5.4 Relational representation of the Northwind data warehouse

3.4 Time Dimension

- A data warehouse is, in the end, a historical database
- In OLTP database applications, temporal information is usually derived from attributes of type DATE using the functions provided by the database system

```
SELECT SUM(SalesAmount)
FROM Time T, Sales S
WHERE T.TimeKey = S.TimeKey AND T.WeekendFlag
```

- The granularity of the time dimension varies depending on their use.
- For example, if we are interested in monthly data, we would define the time dimension with a granularity that will correspond to a month.

5 years: 5 × 12 = 60 tuples

- If we are interested in more detailed data, we could define the time dimension with a granularity that corresponds to a second.

5 × 12 × 30 × 24 × 3,600 = 155,520,000 tuples

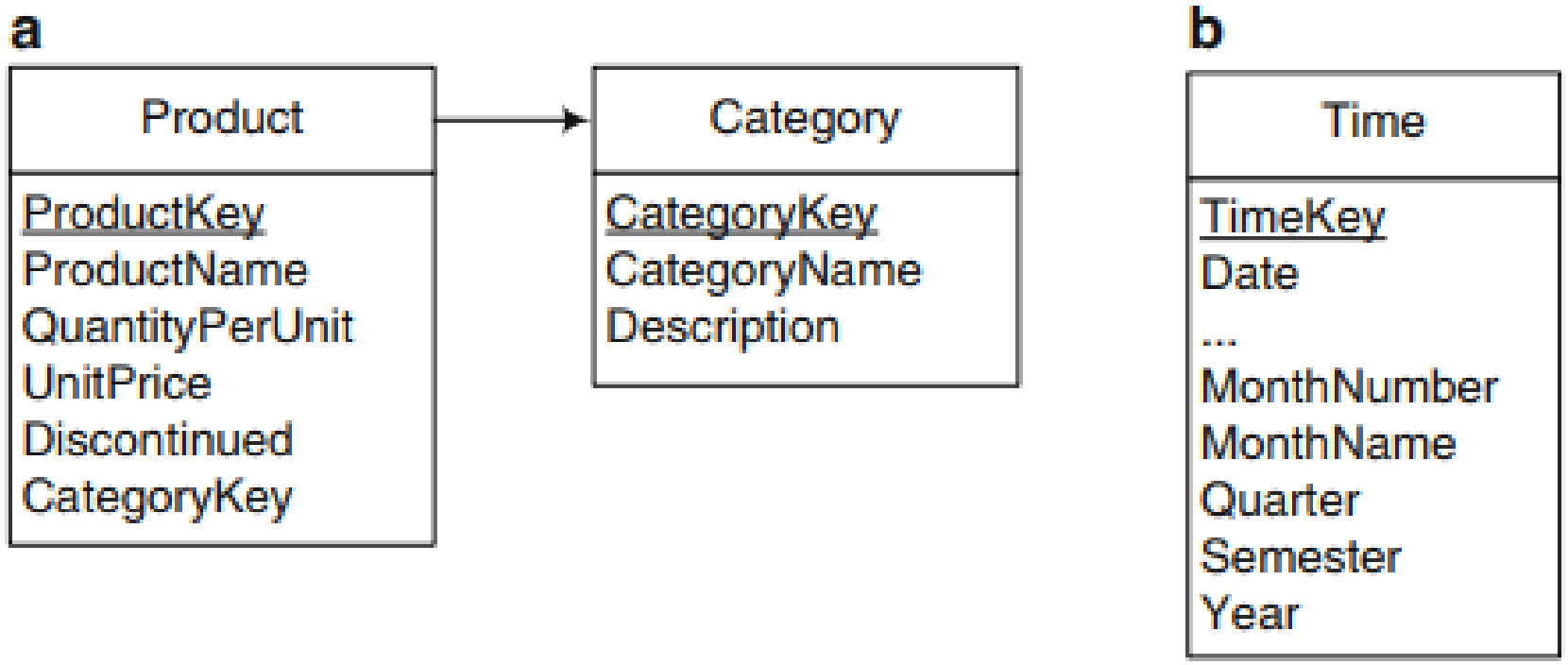
- Time dimension may have more than one hierarchy.
- If we use a single hierarchy, we must be careful to satisfy the summarizability conditions.
- For example, a day aggregates correctly over a month and a year level (a day belongs to exactly 1 month and 1 year), whereas a week may correspond to 2 different months, and thus the week level cannot be aggregated over the month level in a time dimension hierarchy.

3.5 Logical Representation of Hierarchies

3.5.1 Balanced Hierarchies

- In a conceptual multidimensional schema, the levels of dimension hierarchies are represented independently, and these levels are linked by parent-child relationships

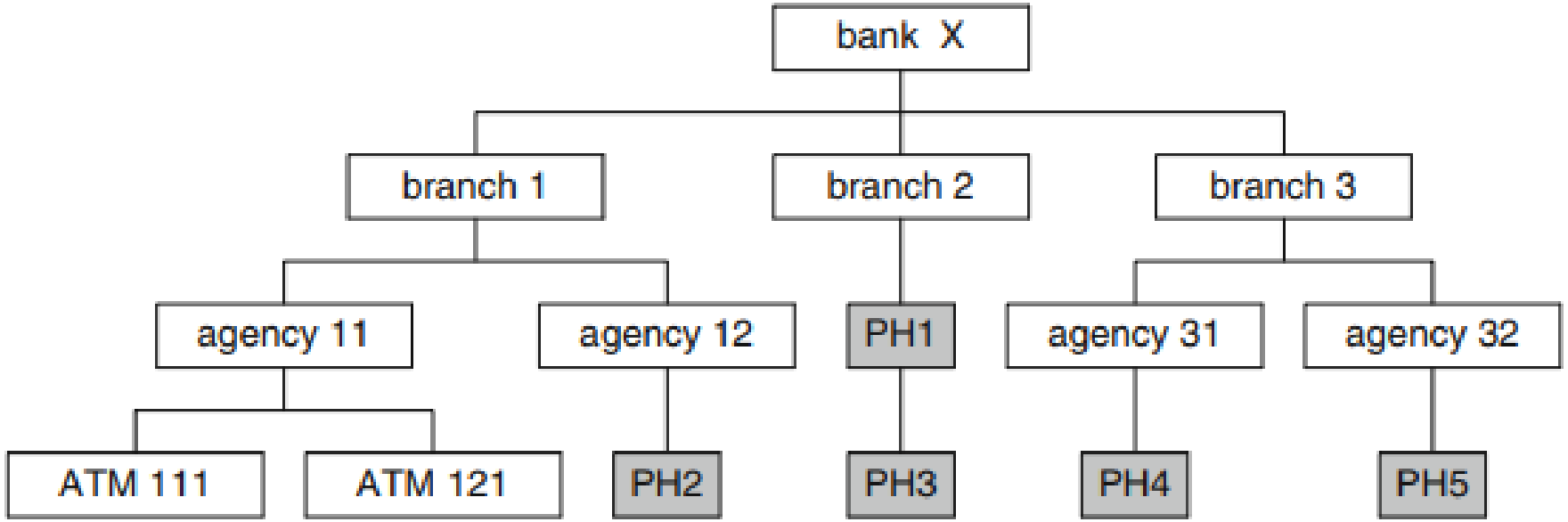
Fig. 5.5 Relations for a balanced hierarchy. (a) Snowflake structure. (b) Flat table



- As we have seen in Sect. 5.2, snowflake schemas better represent hierarchical structures than star schemas, since every level can be easily distinguished and, further, levels can be reused between different hierarchies.
- On the other hand, star schemas facilitate query formulation since fewer joins are needed for expressing queries, owing to denormalization. Additionally, much research has been done to improve system performance for processing star queries. However, star schemas have some drawbacks. For example, they do not model hierarchies adequately since the hierarchy structure is not clear.

3.5.2 Unbalanced Hierarchies

Fig. 5.6 Transformation of the unbalanced hierarchy into a balanced one using placeholders



3.5.2 Unbalanced Hierarchies

- The above transformation has the following consequences. First, the fact table contains measures belonging to all levels whose members can be a leaf at the instance level.
- In particular, recursive queries are necessary for traversing a parent-child hierarchy. Recursive queries are allowed both in SQL and in MDX.

3.5.3 Generalized Hierarchies

3.5.2 Unbalanced Hierarchies

- Generalized hierarchies account for the case where dimension members are of different kinds, and each kind has a specific aggregation path.
- As was the case for balanced hierarchies, two approaches can be used for representing generalized hierarchies at the logical level: create a table for each level, leading to snowflake schemas, or create a single flat table for all the levels, where null values are used for attributes that do not pertain to specific members.
- Alternatively, a mix of these two approaches can be followed: create one table for the common levels and another table for the specific ones.
- Finally, we could also use separate fact and dimension tables for each path.

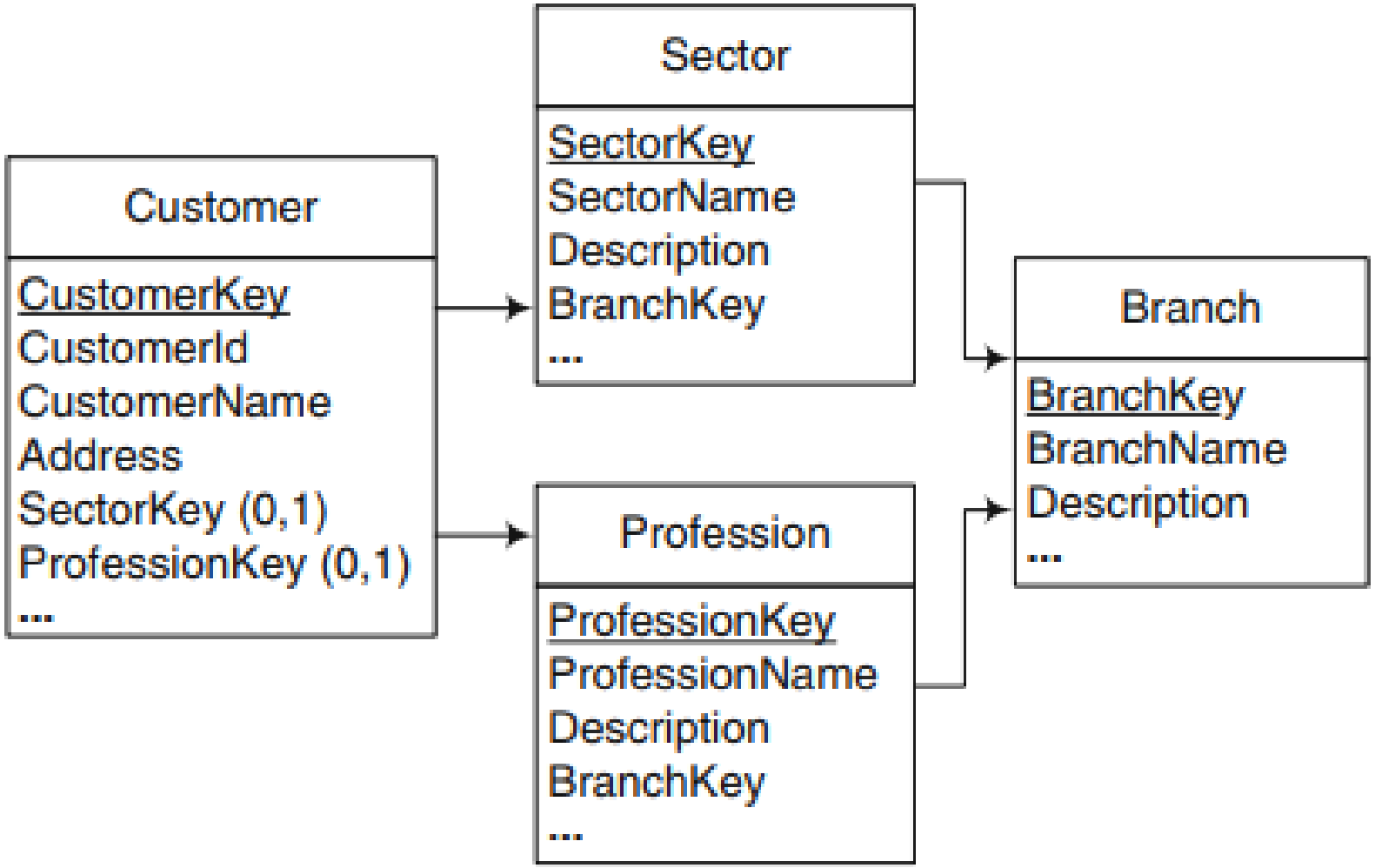
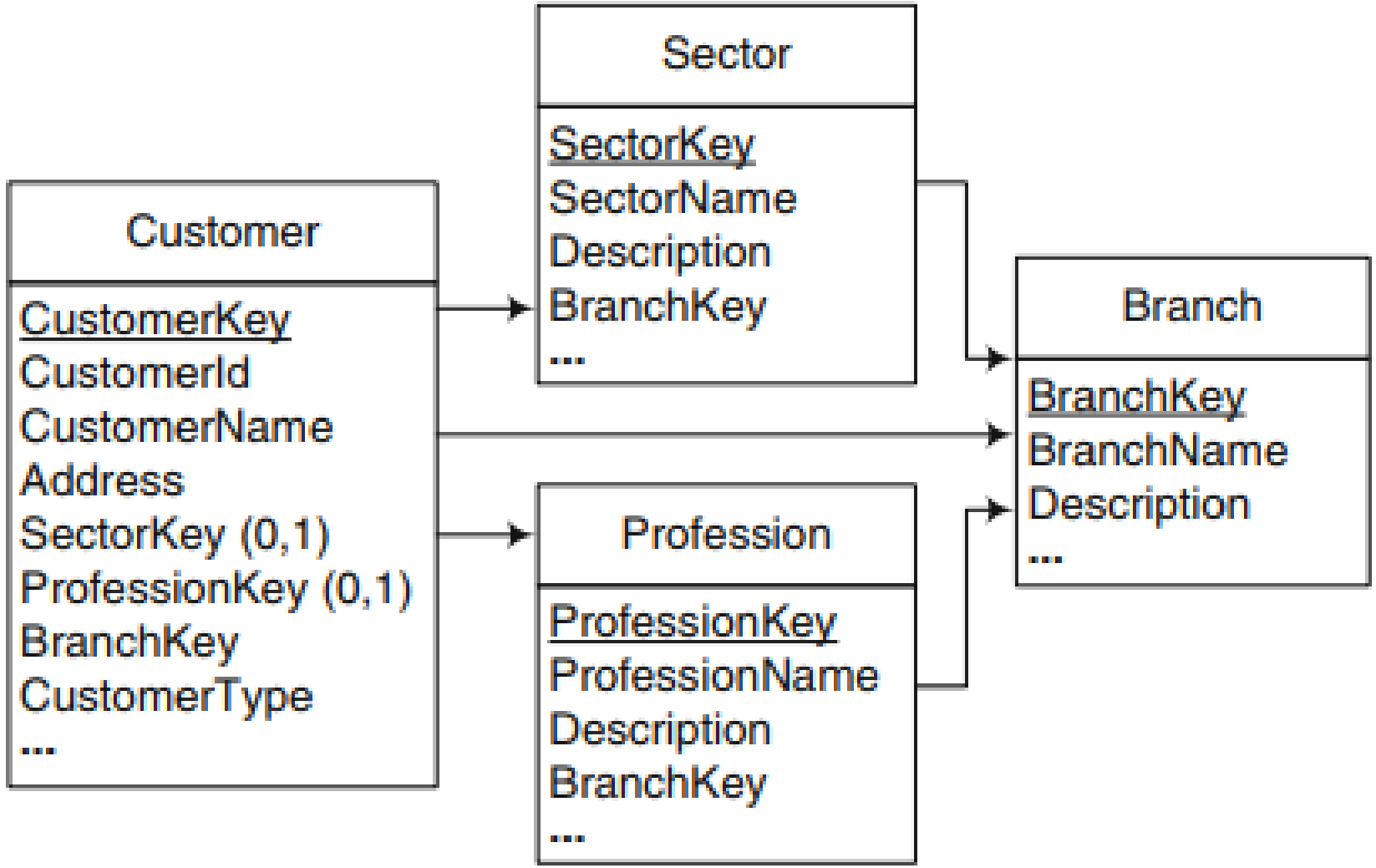


Fig. 5.7 Relations for the generalized hierarchy

Fig. 5.8 Improved relational representation of the generalized hierarchy



```
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 ) )
```

- The mapping above can also be applied to ragged hierarchies since these hierarchies are a special case of generalized hierarchies.
- Finally, another solution would be to transform the hierarchy at the instance level by including placeholders in the missing intermediate levels.

3.5.4 Alternative Hierarchies

3.5.4 Alternative Hierarchies

- For alternative hierarchies, the traditional mapping to relational tables can be applied.
- Note that even though generalized and alternative hierarchies can be easily distinguished at the conceptual level (see Figs. 4.6a and 4.9), this distinction cannot be made at the logical level (compare Figs. 5.7 and 5.9).

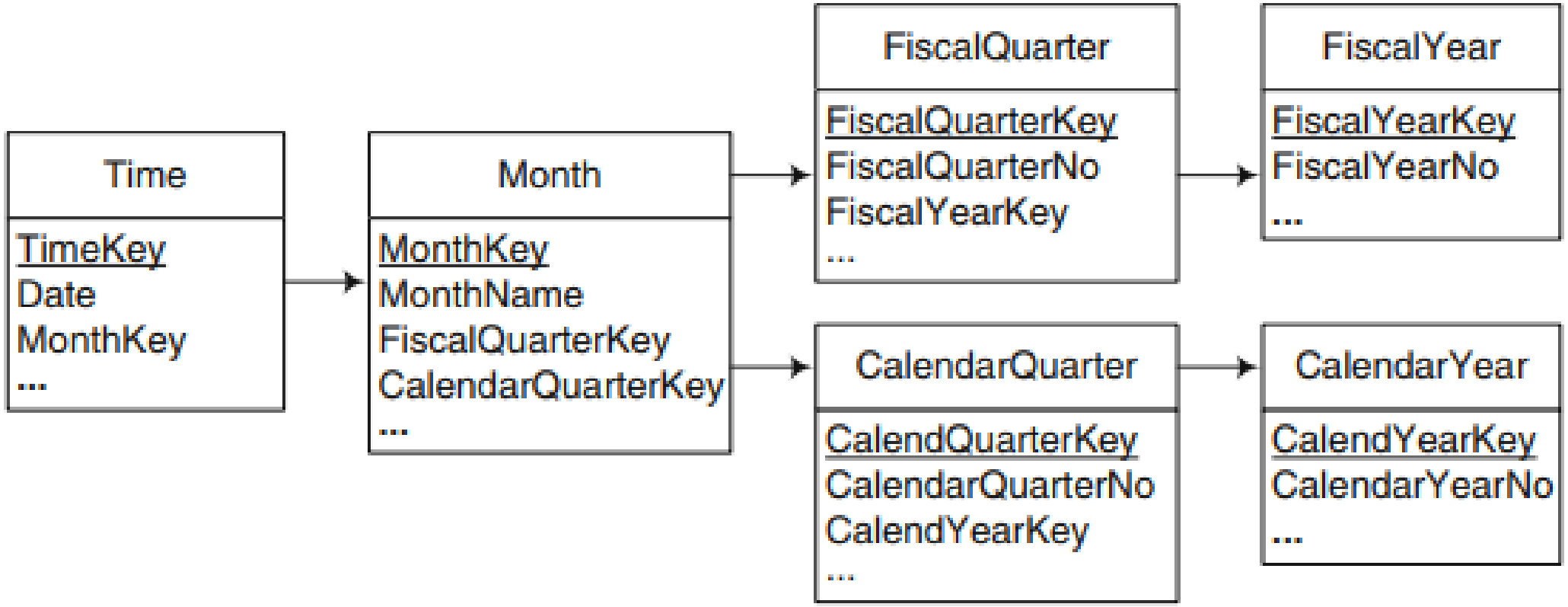


Fig. 5.9 Relations for the alternative hierarchy

3.5.5 Parallel Hierarchies

- As parallel hierarchies are composed of several hierarchies, their logical mapping consists in combining the mappings for the specific types of hierarchies.
- Note that shared levels in parallel dependent hierarchies are represented in one table (State, in this example). Since these levels play different roles in each hierarchy, we can create views in order to facilitate queries and visualization.
- Finally, note also that both alternative and parallel dependent hierarchies can be easily distinguished at the conceptual level.

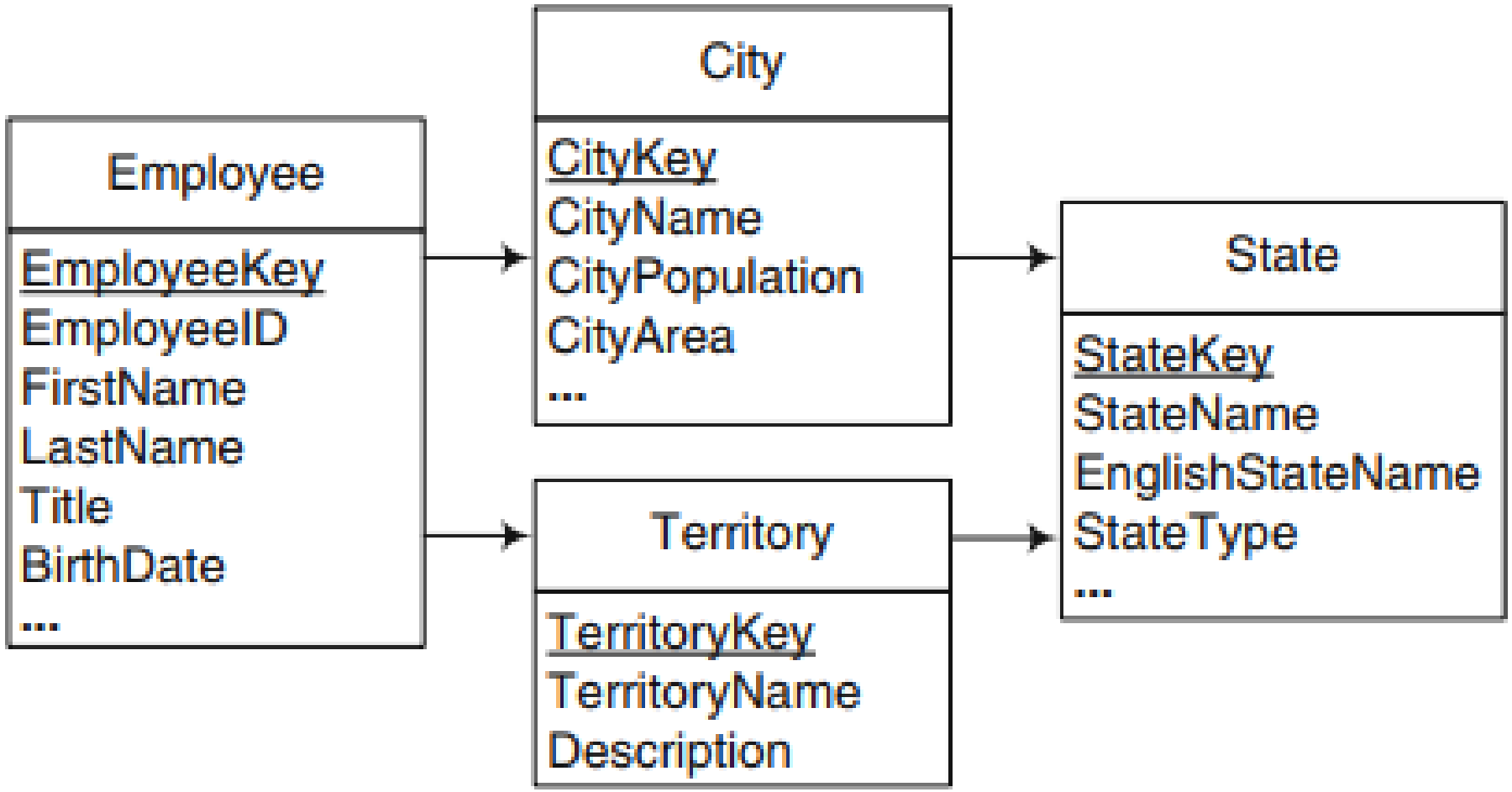
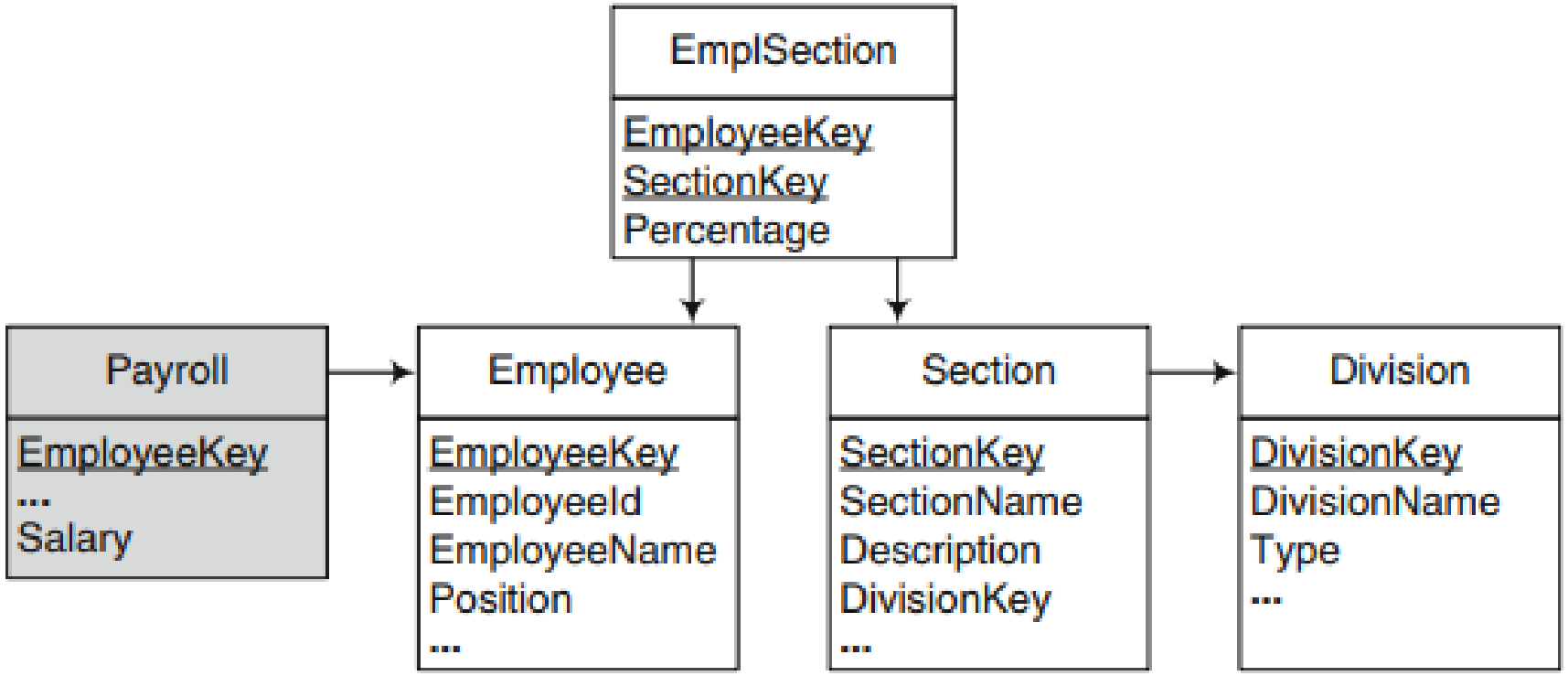


Fig. 5.10 Relations for the parallel dependent hierarchies

3.5.6 Nonstrict Hierarchies

Fig. 5.11 Relations for the nonstrict hierarchy



- Data structure and size: Bridge tables require less space than creating additional dimensions
- Performance and applications: For bridge tables, join operations, calculations, and programming effort are needed to aggregate measures correctly, while in the case of additional dimensions, measures in the fact table are ready for aggregation along the hierarchy
- Finally, still another option consists in transforming the many-to-manyrelationship into a one-to-many relationship by defining a “primary” relationship

3.6 Advanced Modeling Aspects

3.6.1 Facts with Multiple Granularities

3.6.1 Facts with Multiple Granularities

- Two approaches can be used for the logical representation of facts with multiple granularities.
- The first one consists in using multiple foreign keys, one for each alternative granularit.
- The second approach consists in removing granularity variation at the instance level with the help of Placeholders.

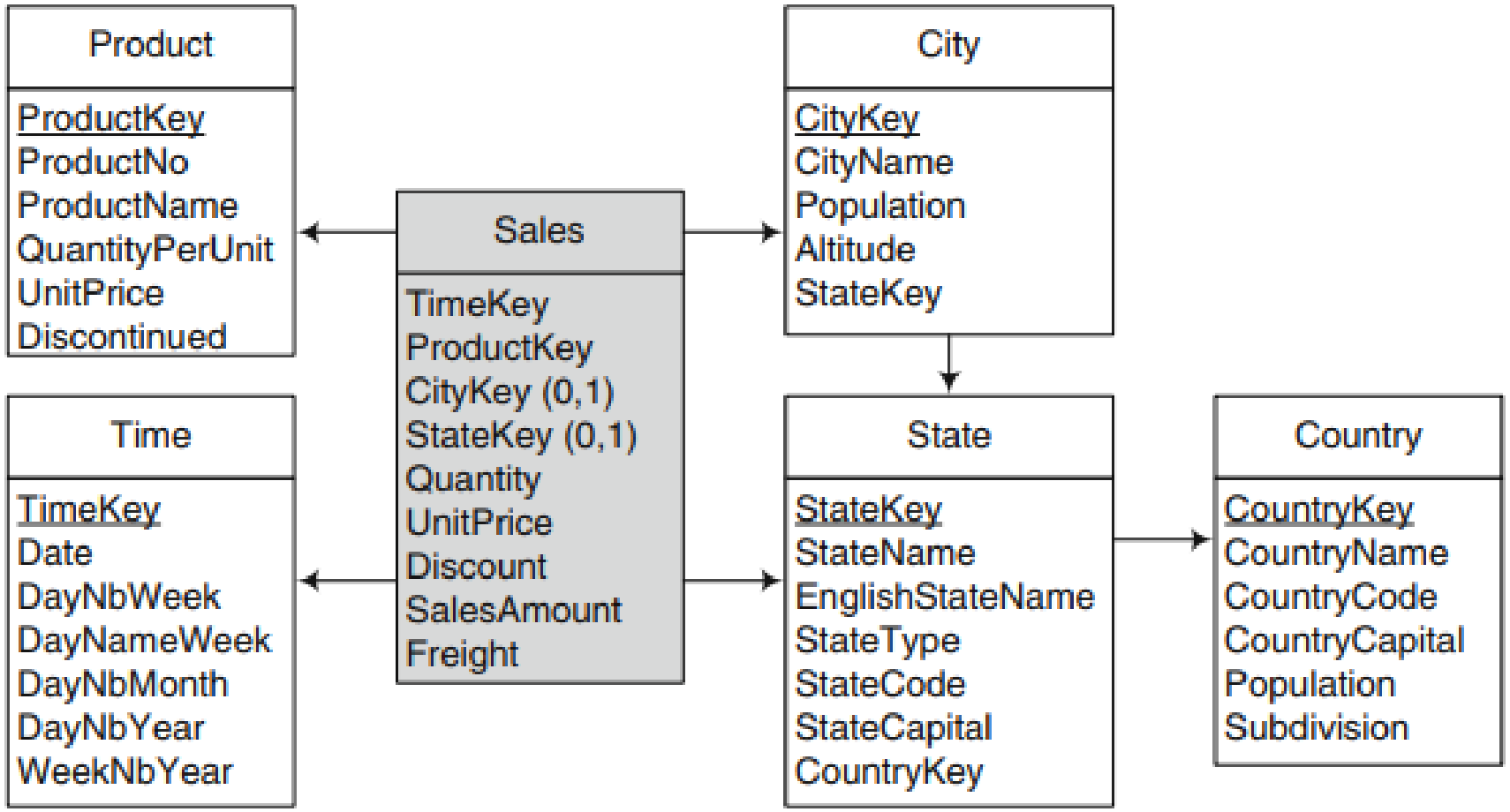


Fig. 5.12 Relations for the fact with multiple granularities

- Obviously, in both solutions, the issue is to guarantee the correct summarization of measures.
- In the first solution, when aggregating at the state level, we need to perform a union of two subqueries, one for each alternative path.
- In the second solution, when aggregating at the city level, we obtain the placeholders in the result.

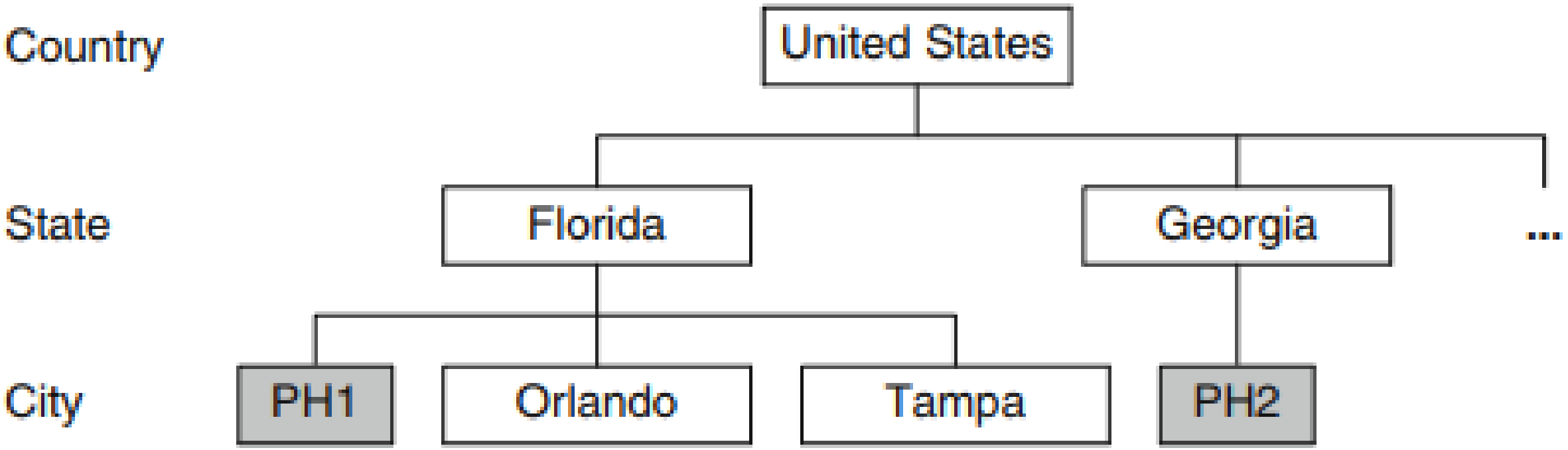


Fig. 5.13 Using placeholders for the fact with multiple granularities

3.6.2 Many-to-Many Dimensions

3.6.2 Many-to-Many Dimensions

- The mapping to the relational model given in Sect. 5.3, applied to many-tomany dimensions, creates relations representing the fact, the dimension levels, and an additional bridge table representing the many-to-many relationship between the fact table and the dimension.

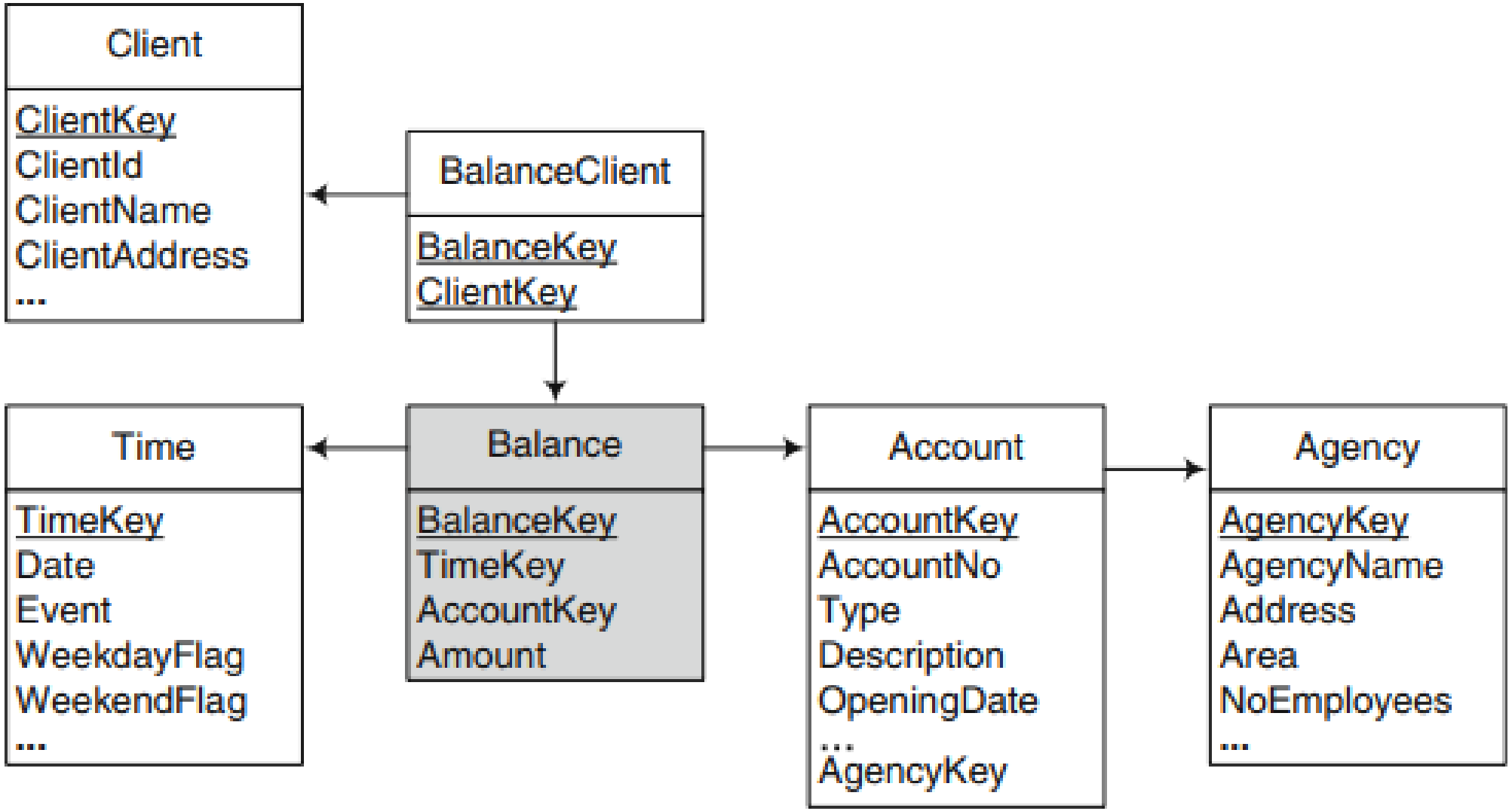


Fig. 5.14 Relations for the many-to-many dimension

3.7 Slowly Changing Dimensions

3.7 Slowly Changing Dimensions

- So far, we have assumed that new data that arrives to the warehouse only corresponds to facts, which means dimensions are stable, and their data do not change. However, in many real-world situations, dimensions can change both at the structure and the instance level.
- First, when a correction must be made to the dimension tables due to an error, the new data should replace the old one.
- Second, when the contextual conditions of an analysis scenario change, the contents of dimension tables must change accordingly. We cover these two latter cases in this section.

TimeKey	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	Discontinued	CategoryName	Description
p1	prod1	No	cat1	desc1
p2	prod2	No	cat1	desc1
p3	prod3	No	cat2	desc2
p4	prod4	No	cat2	desc2

```
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

- Type 1, consists in overwriting the old value of the attribute with the new one.
- Type 2, the tuples in the dimension table are versioned, and a new tuple is inserted each time a change takes place.
- Type 3, consists in introducing an additional column for each attribute subject to change, which will hold the new value of the attribute.

- Type 4, a new dimension, called a minidimension, is created to store the most frequently changing attributes.
- Type 5 approach is an extension of type 4, where the primary dimension table is extended with a foreign key to the minidimension table.
- Type 6 approach extends a type 2 dimension with an additional column containing the current value of an attribute.
- Type 7, in the case that there are many attributes in the dimension table for which we need to support both current and historical perspectives.

3.8 SQL/OLAP Operations

3.8.1 Data Cube

- A relational database is not the best data structure to hold data that is, in nature, multidimensional.
- This explains why MOLAP systems, which store data in special arrays, deliver good performance.

Fig. 5.15 (a) A data cube with two dimensions, Product and Customer. (b) A fact table representing the same data

a

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

b

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

```
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
```

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

Fig. 5.16 Data cube corresponding to the fact table

3.8.2 ROLLUP, CUBE, and GROUPING SETS

3.8.2 ROLLUP, CUBE, and GROUPING SETS

- Computing a cube with n dimensions in the way described above would require 2n GROUP BY statements, which is not very efficient.
- For this reason, SQL/OLAP extends the GROUP BY clause with the ROLLUP and CUBE operators. The former computes group subtotals in the order given by a list of attributes. The latter computes all totals of such a list.
- Over the grouped tuples, the HAVING clause can be applied, as in the case of a typical GROUP BY.

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

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

a

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

b

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

Fig. 5.17 Operators. GROUP BY ROLLUP (a) and GROUP BY CUBE (b)

3.8.2 ROLLUP, CUBE, and GROUPING SETS

- The ROLLUP and CUBE operators are simply shorthands for a more powerful operator, called GROUPING SETS, which is used to precisely specify the aggregations to be computed.

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

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

3.8.3 Window Functions

- A very common OLAP need is to compare detailed data with aggregate values. SQL/OLAP provides the means to perform this through a feature called window partitioning.

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

Fig. 5.18 Sales of products to customers compared with the maximum amount sold for that product

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

- A second SQL/OLAP feature, called window ordering, is used to order the rows within a partition. This feature is useful, in particular, to compute rankings. Two common aggregate functions applied in this respect are ROW NUMBER and RANK.

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

a

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

b

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

Fig. 5.19 (a) Ranking of products in the sales of customers. (b) Ranking of customers in the sales of products

- A third kind of feature of SQL/OLAP is window framing, which defines the size of the partition. This is used to compute statistical functions over time series, like moving averages.

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

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

a

Product Key	Year	Month	Sales Amount	MovAvg
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

b

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

Fig. 5.20 (a) Three-month moving average of the sales per product. (b) Year-to-date sum of the sales per product

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

```
SELECT ProductKey, Year, Month, SalesAmount, AVG(SalesAmount) AS
YTD
FROM Sales S1, Sales S2
WHERE S1.ProductKey = S2.ProductKey AND
S1.Year = S2.Year AND S1.Month >= S2.Month
```

3.9 Definition of the Northwind Cube in Analysis Services

3.9.1 Data Sources

3.9.1 Data Sources

- A data warehouse retrieves its data from one or several data stores.
- A data source contains connection information to a data store, which includes the location of the server, a login and password, a method to retrieve the data, and security permissions.

3.9.2 Data Source Views

3.9.2 Data Source Views

- A data source view (DSV) defines the relational schema that is used for populating an Analysis Services database. This schema is derived from the schemas of the various data sources. Indeed, some transformations are often needed in order to load data from sources into the warehouse.
- Analysis Services allows the user to specify friendly names for tables and columns. In order to facilitate visibility and navigation for large data warehouses, it also offers the possibility to define customizable views within a DSV, called diagrams, that show only certain tables.

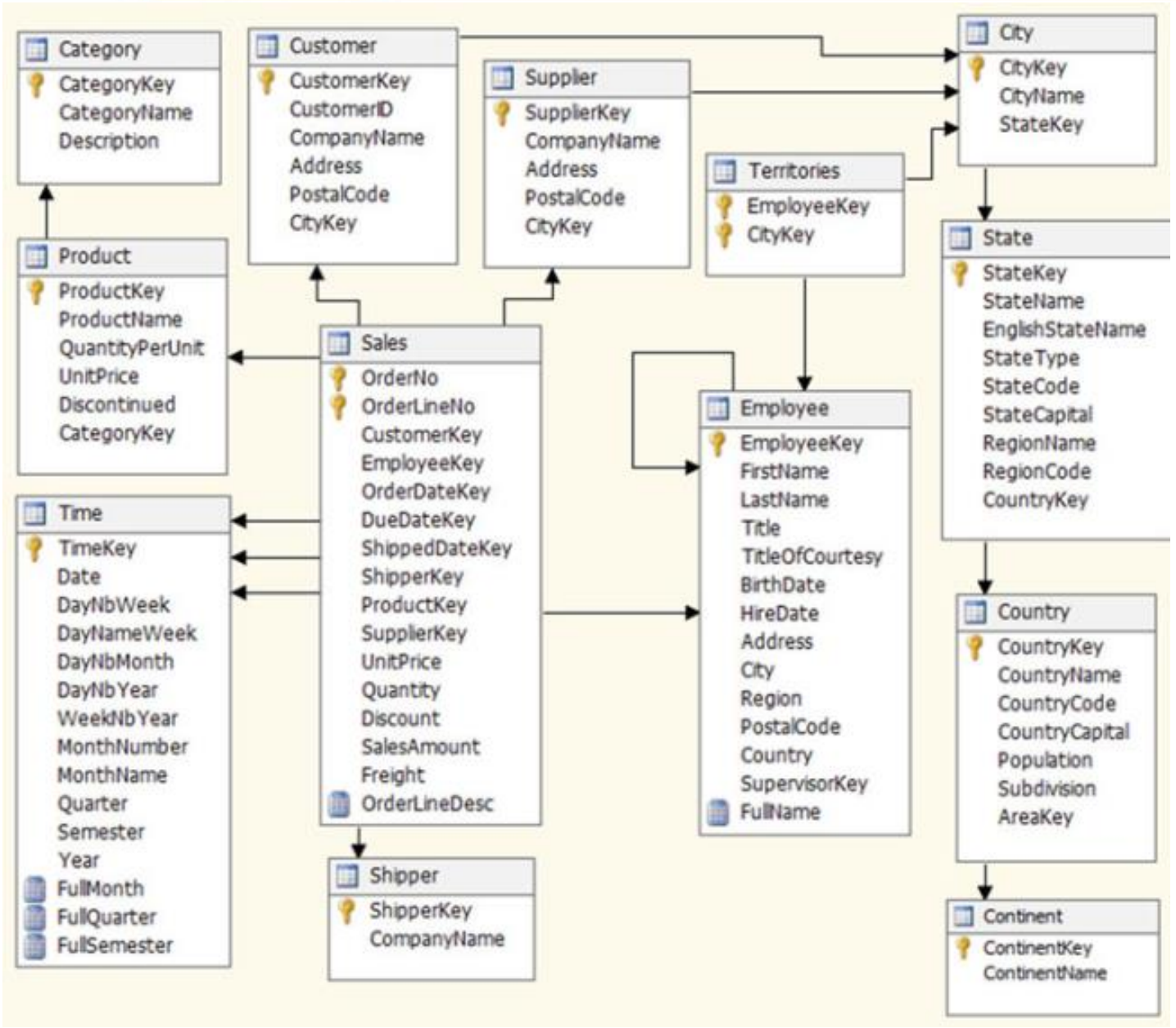


Fig. 5.21 The data source view for the Northwind cube

- In the Employee dimension table:

FirstName + ' ' + LastName

- In the Time dimension table:

MonthName + ' ' + CONVERT(CHAR(4),Year)

'Q' + CONVERT(CHAR(1), Quarter) + ' ' + CONVERT(CHAR(4), Year)

'S' + CONVERT(CHAR(1), Semester) + ' ' + CONVERT(CHAR(4), Year)

- In the Sales fact table:

CONVERT(CHAR(5),OrderNo) + ' - ' + CONVERT(CHAR(1),OrderLineNo)

3.9.3 Dimensions

- A regular dimension has a direct one-to-many link between a fact table and a dimension table.
- A reference dimension is indirectly related to the fact table through another dimension.
- In a role-playing dimension, a single fact table is related to a dimension table more than once.
- A fact dimension, also referred to as degenerate dimension, is similar to a regular dimension, but the dimension data are stored in the fact table.
- In a many-to-many dimension, a fact is related to multiple dimension members and a member is related to multiple facts.

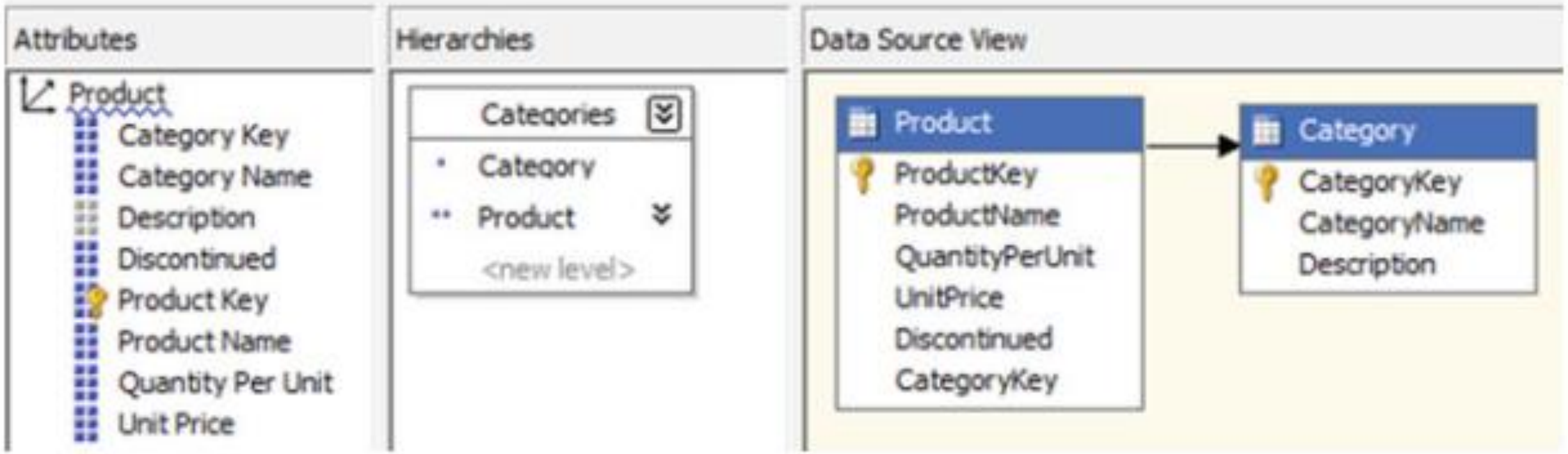


Fig. 5.22 Definition of the Product dimension

Fig. 5.23 Browsing the hierarchy of the Product dimension

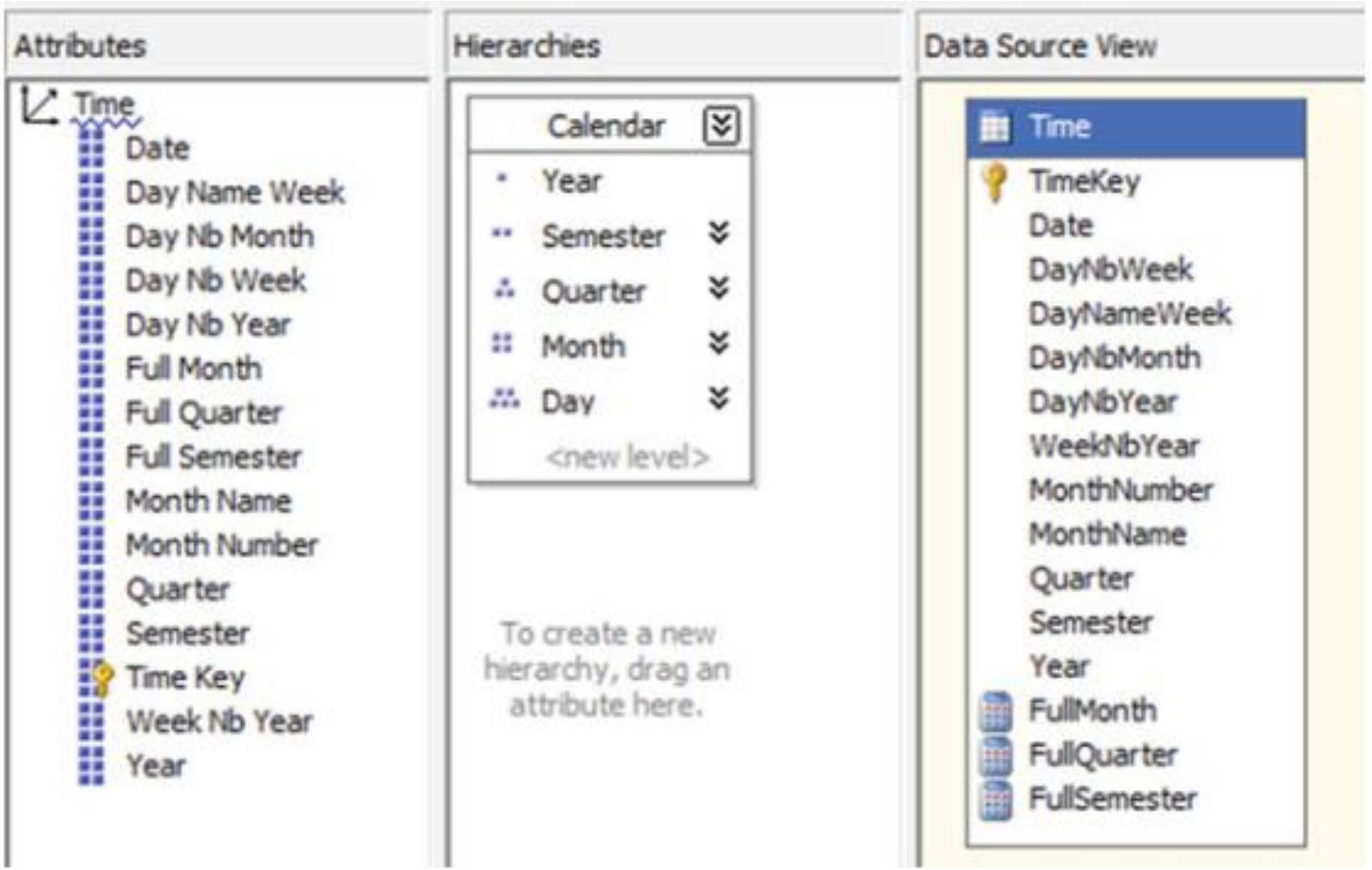


Fig. 5.24
Definition of the Time dimension

- Attributes in hierarchies must have a one-to-many relationship to their parents in order to ensure correct roll-up operations.
- For example, a quarter must roll up to its semester. In Analysis Services, this is stated by defining a key for each attribute composing a hierarchy.
- By default, this key is set to the attribute itself, which implies that, for example, years are unique.

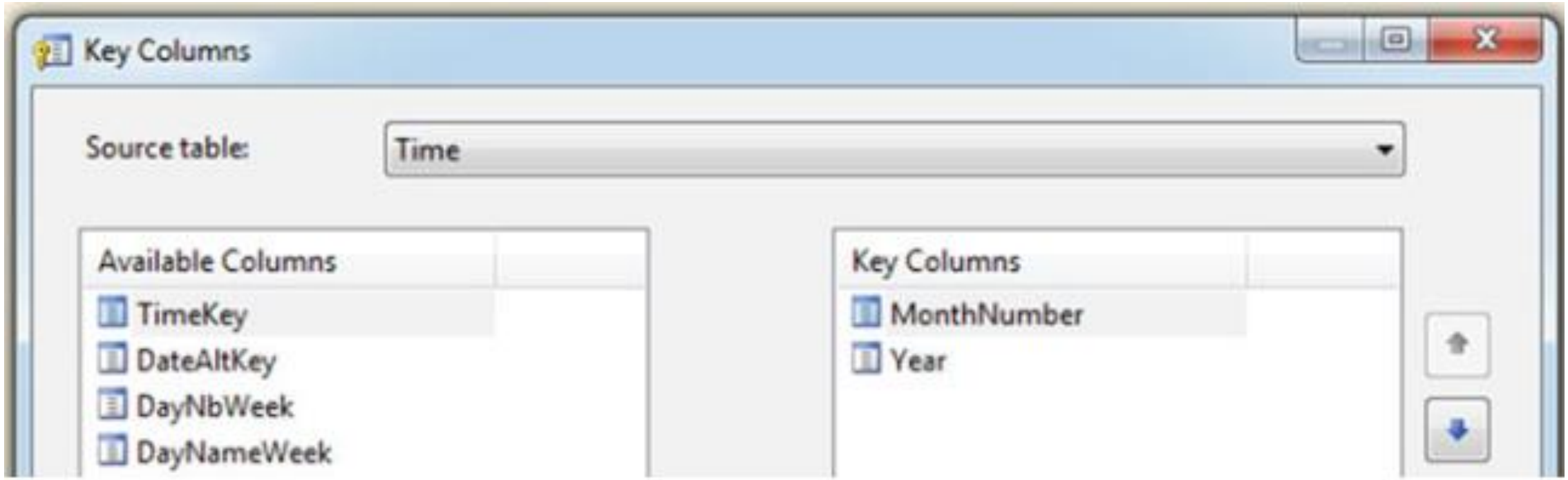


Fig. 5.25 Definition of the key for attribute MonthNumber in the Calendar hierarchy

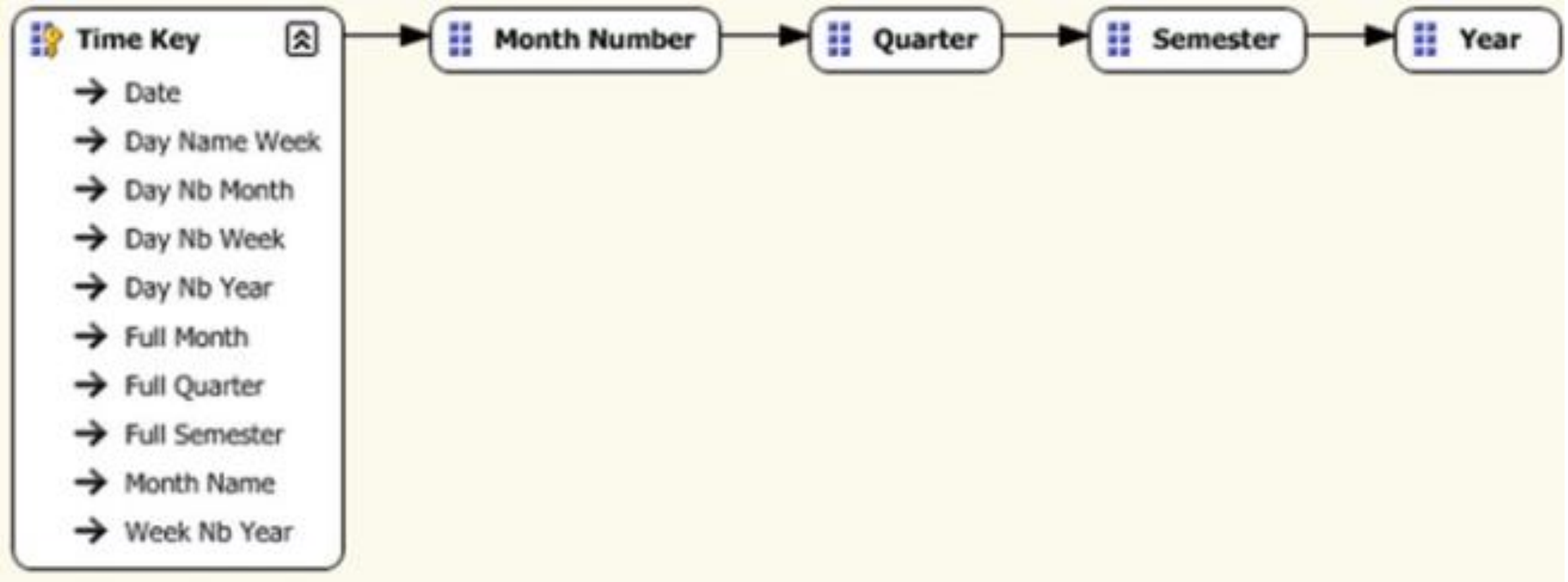


Fig. 5.26 Definition of the relationships in the Calendar hierarchy

- When creating a user-defined hierarchy, it is necessary to establish the relationships between the attributes composing such hierarchy. These relationships correspond to functional dependencies.
- The relationships for the Time dimension are given in Fig. 5.26.
- In Analysis Services, there are two types of relationships, flexible and rigid. Flexible relationships can evolve across time (e.g., a product can be assigned to a new category), while rigid ones cannot (e.g., a month is always related to its year). The relationships shown in Fig. 5.26 are rigid, as indicated by the solid arrowhead.

Fig. 5.27 Browsing the hierarchy in the Time dimension

- The definition of the fact dimension Order follows similar steps than for the other dimensions, except that the source table for the dimension is the fact table. The key of the dimension will be composed of the combination of the order number and the line number.
- Finally, in many-to-many dimensions, like in the case of City and Employee, we also need to indicate that the bridge table Territories is actually defined as a fact table, so Analysis Services can take care of the doublecounting problem. This is also done when defining the cube.

3.9.4 Hierarchies

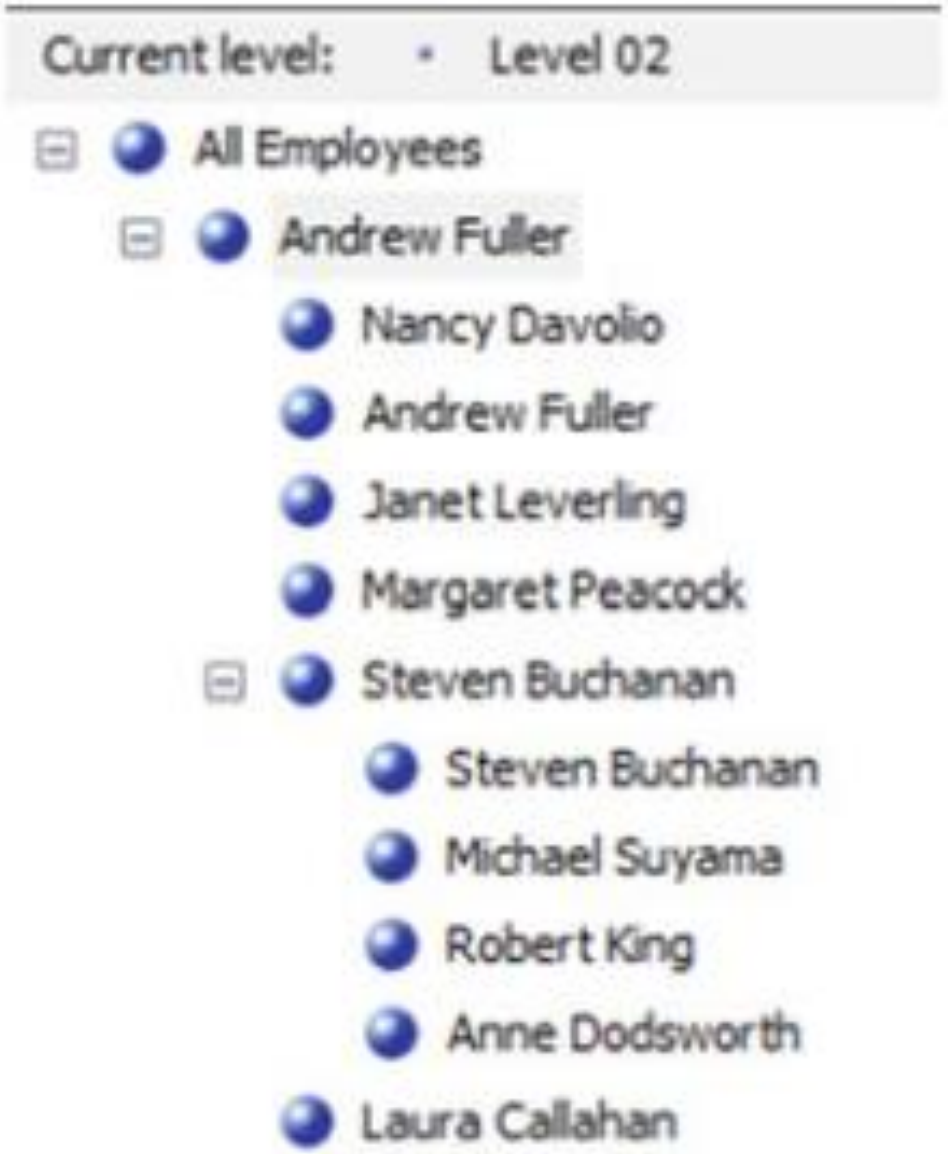


Fig. 5.28 Browsing the Supervision hierarchy in the Employee dimension

- Balanced hierarchies are supported by Analysis Services. Examples of these hierarchies in the Northwind cube are the Time and the Product dimensions studied above.
- Analysis Services does not support unbalanced hierarchies. We have seen in Sect. 5.5.2 several solutions to cope with them. On the other hand, Analysis Services supports parent-child hierarchies, which are a special case of unbalanced hierarchies. We have seen that such hierarchies define a hierarchical relationship between the members of a dimension.
- In parent-child hierarchies, the hierarchical structure between members is taken into account when measures are aggregated.
- Analysis Services does not support generalized hierarchies.

- **Never:** Level members are never hidden.
- **OnlyChildWithNoName:** A level member is hidden when it is the only child of its parent and its name is null or an empty string.
- **OnlyChildWithParentName:** A level member is hidden when it is the only child of its parent and its name is the same as the name of its parent.
- **NoName:** A level member is hidden when its name is empty.
- **ParentName:** A level member is hidden when its name is identical to that of its parent.

3.9.5 Cubes

- In Analysis Services, a cube is built from one or several DSVs. A cube consists of one or more dimensions from dimension tables and one or more measure groups from fact tables. A measure group is composed by a set of measures.
- The facts in a fact table are mapped as measures in a cube. Analysis Services allows multiple fact tables in a single cube.
- In this case, the cube typically contains multiple measure groups, one from each fact table.

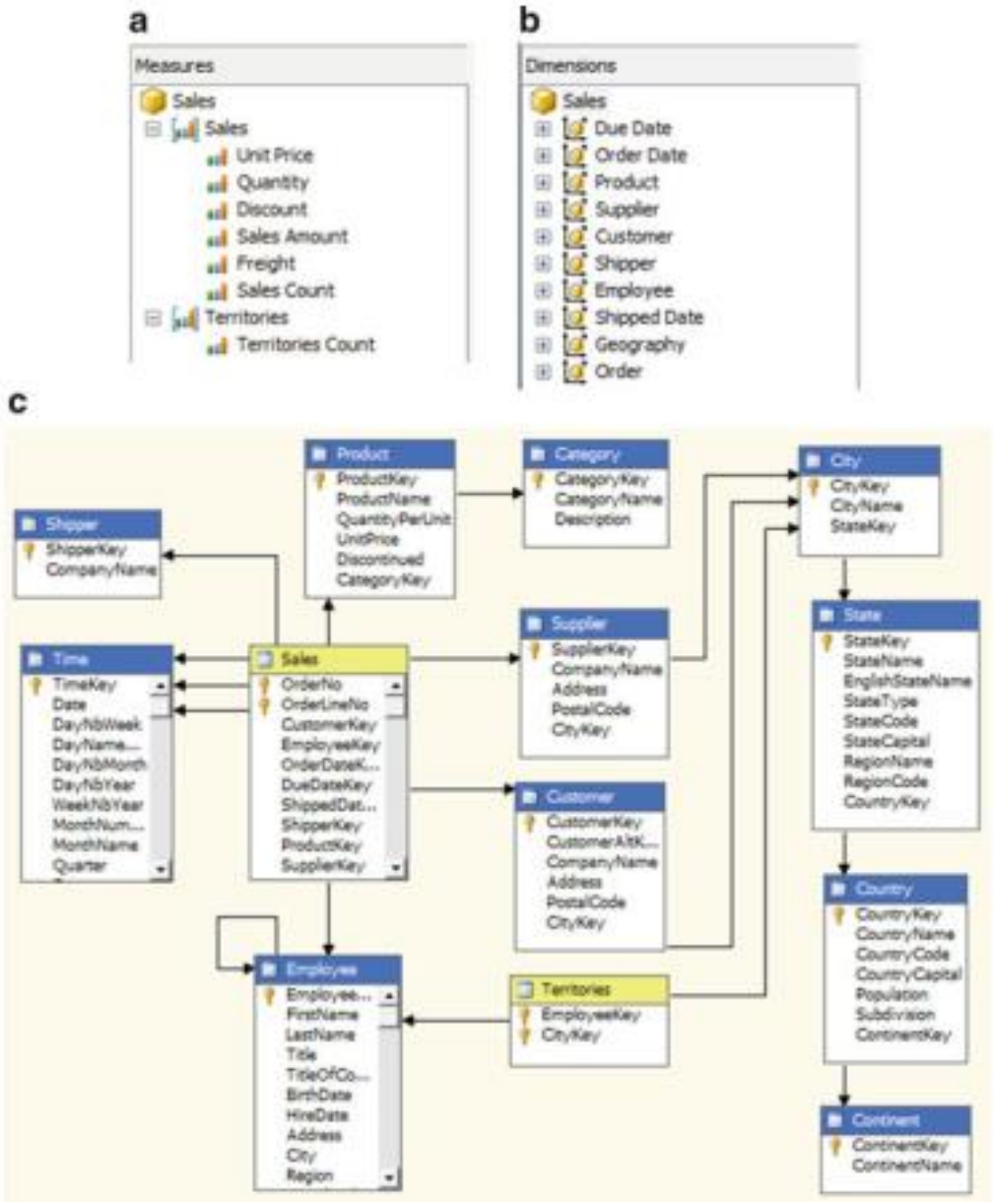
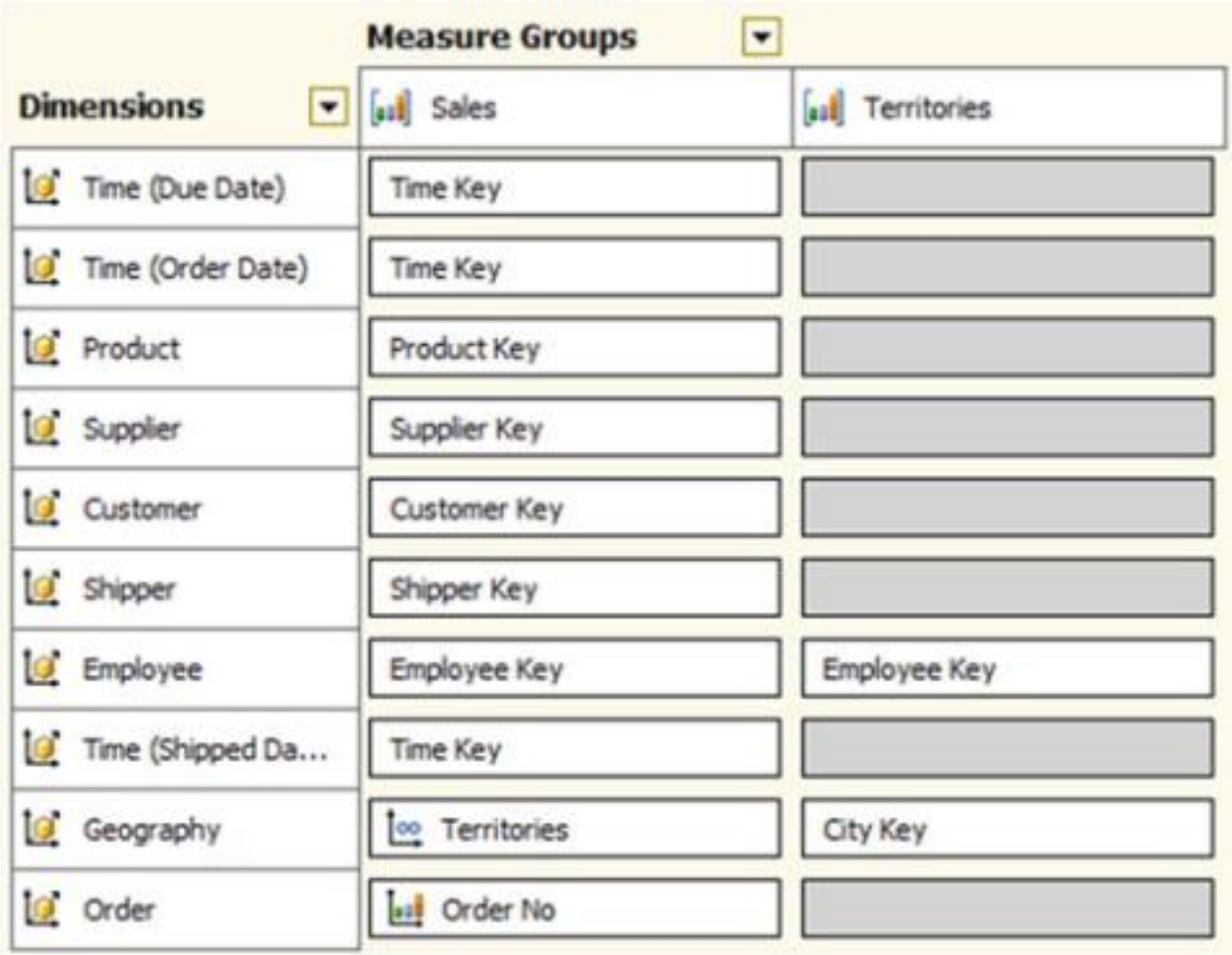


Fig. 5.29 Definition of the Northwind cube in Analysis Services.

(a) Measure groups.

(b) Dimensions.

(c) Schema of the cube.



Dimensions	Measure Groups: Sales	Measure Groups: Territories
Time (Due Date)	Time Key	
Time (Order Date)	Time Key	
Product	Product Key	
Supplier	Supplier Key	
Customer	Customer Key	
Shipper	Shipper Key	
Employee	Employee Key	Employee Key
Time (Shipped Da...	Time Key	
Geography	Territories	City Key
Order	Order No	

Fig. 5.30 Definition of the dimensions of the Northwind cube

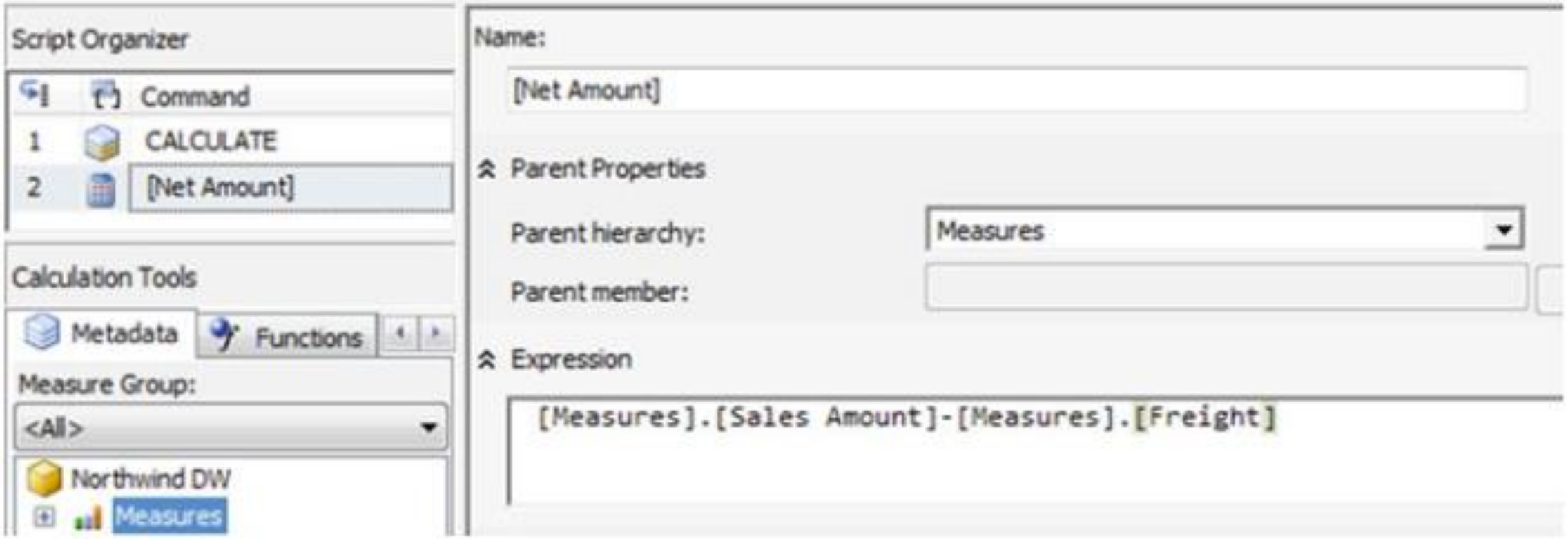


Fig. 5.31 Definition of the Net Amount derived measure

Fig. 5.32 Browsing the Northwind cube in Excel

	A	B	C	D	E
1	Sales Amount	Column Labels			
2	Row Labels	1996	1997	1998	Grand Total
3	Europe	113290,8904	351142,6916	219090,1785	683523,7605
4	North America	49524,655	154427,8604	103073,3274	307025,8428
5	Canada	7283,0801	30589,2604	9539,475	47411,8155
6	Mexico	4687,9	12700,8275	3734,9	21123,6275
7	United States	37553,6749	111137,7725	89798,9524	238490,3998
8	Alaska	4675,8	3951,3749	4792,8876	13420,0625
9	California		1698,4025	1378,07	3076,4725
10	San Francisco		1698,4025	1378,07	3076,4725
11	Let's Stop N Shop		1698,4025	1378,07	3076,4725
12	Idaho	10338,2649	56241,075	35674,5099	102253,8498
13	Montana		1426,74	326	1752,74
14	New Mexico	9923,78	19383,75	19982,5499	49290,0799
15	Oregon	1828	15150,975	11011,135	27990,11
16	Washington	2938,2	10810,4551	15516,8	29265,4551
17	Wyoming	7849,63	2475	1117	11441,63
18	South America	29034,32	64629,0564	60942,879	154606,2554
19	Grand Total	191849,8654	570199,6084	383106,3849	1145155,859

By Customer and Time

3.10 Definition of the Northwind Cube in Mondrian

3.10 Definition of the Northwind Cube in Mondrian

- Mondrian is an open-source relational online analytical processing (ROLAP) server. It is also known as Pentaho Analysis Services and is a component of the Pentaho Business Analytics suite.
- In Mondrian, a cube schema written in an XML syntax defines a mapping between the physical structure of the relational data warehouse and the multidimensional cube.
- A cube schema contains the declaration of cubes, dimensions, hierarchies, levels, measures, and calculated members.
- A cube schema does not define the data source; this is done using a JDBC connection string.

3.10 Definition of the Northwind Cube in Mondrian

```
1   <Schema name='NorthwindDW' metamodelVersion='4.0'
2           description='Sales cube of the Northwind company'>
3       <PhysicalSchema>
4           ...
5       </PhysicalSchema>
6       <Dimension name='Time' table='Time' ...>
7           ...
8       </Dimension>
9       <Cube name='Sales'>
10          <Dimensions>
11              ...
12          </Dimensions>
```

3.10 Definition of the Northwind Cube in Mondrian

```
13          <MeasureGroups>
14              <MeasureGroup name='Sales' table='Sales'>
15                  <Measures>
16                      ...
17                  </Measures>
18                  <DimensionLinks>
19                      ...
20                  </DimensionLinks>
21              </MeasureGroup>
22          </MeasureGroups>
23      </Cube>
24  </Schema>
```

3.10.1 Schemas and Physical Schemas

3.10.1 Schemas and Physical Schemas

```
1  <PhysicalSchema>
2      <Table name='Employee' keyColumn='EmployeeKey'>
3          <ColumnDefs>
4              <ColumnDef name='EmployeeKey' type='Integer' />
5              <ColumnDef name='FirstName' type='String' />
6              <ColumnDef name='LastName' type='String' />
7              ...
8              <CalculatedColumnDef name='FullName' type='String'>
9                  <ExpressionView>
10                     <SQL dialect='generic'>
11                         <Column name='FirstName' /> ||' '||
12                         <Column name='LastName' />
13                     </SQL>
```

```
14                    <SQL dialect='SQL Server'>
15                        <Column name='FirstName' /> + ' ' +
16                        <Column name='LastName' />
17                    </SQL>
18                </ExpressionView>
19            </CalculatedColumnDef>
20        </ColumnDefs>
21    </Table>
22    ...
23    <Link source='City' target='Employee' foreignKeyColumn='CityKey' />
24    ...
25 </PhysicalSchema>
```

3.10.2 Cubes, Dimensions, Attributes, and Hierarchies

3.10.2 Cubes, Dimensions, Attributes, and Hierarchies

```
1  <Dimension name='Time' table='Time' type='TIME'>
2      <Attributes>
3          <Attribute name='Year' keyColumn='Year' levelType='TimeYears' />
4          ...
5          <Attribute name='Month' levelType='TimeMonths'
6              nameColumn='FullMonth' orderByColumn='MonthNumber' />
7              <Key>
8                  <Column name='Year' />
9                  <Column name='MonthNumber' />
10             </Key>
11         </Attribute>
12         ...
13     </Attributes>
14     <Hierarchies>
15         <Hierarchy name='Calendar' hasAll='true'>
16             <Level attribute='Year' />
17             ...
18             <Level attribute='Month' />
19             ...
20         </Hierarchy>
21     </Hierarchies>
22 </Dimension>
```

3.10.2 Cubes, Dimensions, Attributes, and Hierarchies

```
1  <Dimension name='Product' table='Product' key='Product Key'>
2      <Attributes>
3          <Attribute name='Category Name' keyColumn='CategoryName'
4              table='Category' />
5          ...
6          <Attribute name='Product Name' keyColumn='ProductName' />
7          ...
8      </Attributes>
9      <Hierarchies>
10         <Hierarchy name='Categories' hasAll='true'>
11             <Level name='Category' attribute='Category Name' />
12             <Level name='Product' attribute='Product Name' />
13         </Hierarchy>
14     </Hierarchies>
15 </Dimension>
```

3.10.2 Cubes, Dimensions, Attributes, and Hierarchies

- Mondrian implicitly creates attribute hierarchies, even if a hierarchy is not defined explicitly for the attribute
- When a schema has more than one cube, these cubes may have several dimensions in common. If these dimensions have the same definitions, they are declared once and can be used in as many cubes as needed. In Mondrian, these are called shared dimensions.

3.10.2 Cubes, Dimensions, Attributes, and Hierarchies

```
1  <Dimension name='Employee' table='Employee' key='Employee Key'>
2      <Attributes>
3          <Attribute name='Employee Key' keyColumn='EmployeeKey' />
4          ...
5          <Attribute name='Supervisor Key' keyColumn='SupervisorKey' />
6      </Attributes>
7      <Hierarchies>
8          <Hierarchy name='Supervision' hasAll='true'>
9              <Level name='Employee' attribute='Employee Key'
10                 parentAttribute='Supervisor Key' nullParentValue='NULL' />
11         </Hierarchy>
12     </Hierarchies>
13 </Dimension>
```

3.10.2 Cubes, Dimensions, Attributes, and Hierarchies

```
1  <Dimension name='Customer' table='Customer' />
2      <Attributes>
3          <Attribute name='Continent' table='Continent'
4              keyColumn='ContinentKey' />
5          <Attribute name='Country' table='State' keyColumn='CountryKey' />
6          <Attribute name='Region' table='State' keyColumn='RegionName' />
7          <Attribute name='State' table='State' keyColumn='StateKey' />
8          <Attribute name='City' table='City' keyColumn='CityKey' />
9          <Attribute name='Customer' keyColumn='CustomerKey' />
10         ...
11     </Attributes>
12     <Hierarchies>
13         <Hierarchy name='Geography' />
14             <Level attribute name='Continent' />
15             <Level attribute name='Country' />
16             <Level attribute name='Region' hideMemberIf='IfBlankName' />
17             <Level attribute name='State' hideMemberIf='IfBlankName' />
18             <Level attribute name='City' />
19             <Level attribute name='Customer' />
20         </Hierarchy>
21     </Hierarchies>
22 </Dimension>
```

3.10.3 Measures

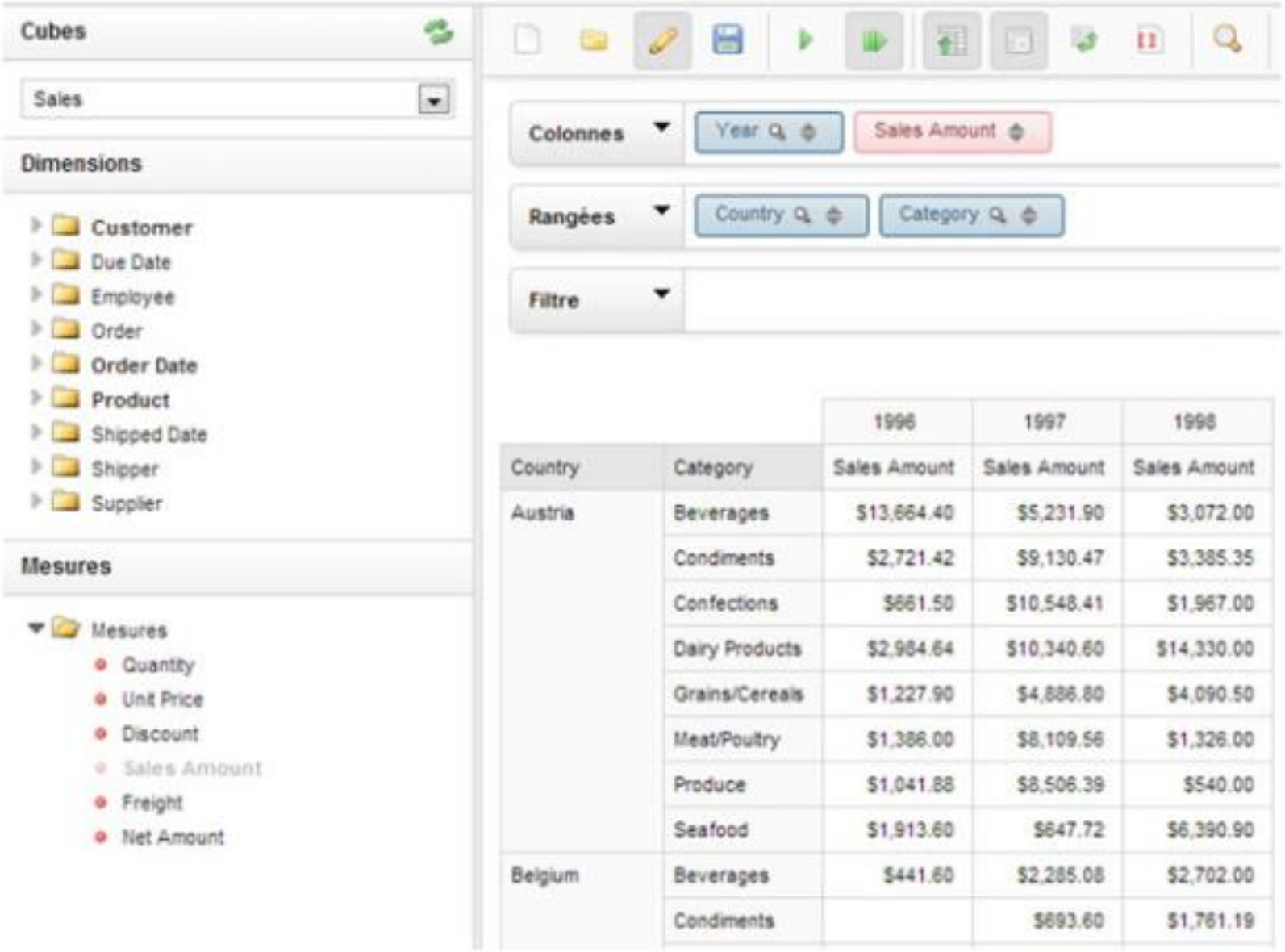
3.10.3 Measures

```
1  <Cube name='Sales'>
2      <Dimensions ... />
3      <MeasureGroups>
4          <MeasureGroup name='Sales' table='Sales'>
5              <Measures>
6                  <Measure name='Unit Price' column='UnitPrice'
7                      aggregator='avg' formatString='$#,##0.00' />
8                  <Measure name='Sales Count' aggregator='count' />
9                  ...
10             </Measures>
11             <DimensionLinks>
12                 <ForeignKeyLink dimension='Customer'
13                     foreignKeyColumn='CustomerKey' />
```

```
14                  <ForeignKeyLink dimension='OrderDate'
15                      foreignKeyColumn='OrderDateKey' />
16                  ...
17                  <FactLink dimension='Order' />
18              </DimensionLinks>
19          </MeasureGroup>
20      </MeasureGroups>
21      <CalculatedMember name='Net Amount' dimension='Measures'
22              formula='[Measures].[Sales Amount]-[Measures].[Freight]'>
23          <CalculatedMemberProperty name='FORMAT_STRING'
24              value='$#,##0.00' />
25      </CalculatedMember>
26  </Cube>
```

3.10.3 Measures

- A measure is defined within a measure group using a Measure element
- Mondrian supports calculated measures, which are calculated from other measures using an MDX formula.



		1996	1997	1998
Country	Category	Sales Amount	Sales Amount	Sales Amount
Austria	Beverages	$13,664.40	$5,231.90	$3,072.00
	Condiments	$2,721.42	$9,130.47	$3,385.35
	Confections	$661.50	$10,548.41	$1,967.00
	Dairy Products	$2,984.64	$10,340.60	$14,330.00
	Grains/Cereals	$1,227.90	$4,886.80	$4,090.50
	Meat/Poultry	$1,386.00	$8,109.56	$1,326.00
	Produce	$1,041.88	$8,506.39	$540.00
	Seafood	$1,913.60	$647.72	$6,390.90
Belgium	Beverages	$441.60	$2,285.08	$2,702.00
	Condiments		$693.60	$1,761.19

Fig. 5.33 Browsing the Northwind cube in Saiku

Summary

Summary

- We have studied the problem of logical design of data warehouses, specifically relational data warehouse design.
- Several alternatives were discussed: the star, snowflake, starflake, and constellation schemas. Like in the case of operational databases, we provided rules for translating conceptual multidimensional schemas to logical schemas.
- Particular importance was given to the representation of the various kinds of hierarchies that can occur in practice.
- The problem of slowly changing dimensions was also addressed in detail.
- …

Review Questions

- **5.1** Describe the differences between the following concepts:

  (a) Relational OLAP (ROLAP), multidimensional OLAP (MOLAP), and hybrid OLAP (HOLAP).

  (b) Star schema, snowflake schema, starflake schema, and constellation schema.

- **5.2** Discuss the mapping rules for translating a MultiDim schema into a relational schema. Are these rules similar to those used for translating an ER schema into a relational schema?

- **5.3** Explain how a balanced hierarchy can be mapped into either normalized or denormalized tables. Discuss the advantages and disadvantages of these alternative mappings.

- **5.4** How do you transform at the logical level an unbalanced hierarchy into a balanced one?
- **5.5** Describe different approaches for representing generalized hierarchies at the logical level.
- **5.6** Is it possible to distinguish between generalized, alternative, and parallel dependent hierarchies at the logical level?
- **5.7** Explain how a nonstrict hierarchy can be represented in the relational model.
- **5.8** Analyze and discuss the pros and cons of the alternatives for representing slowly changing dimensions.

- **5.9** Define the kinds of SQL/OLAP window functions: partitioning, window ordering, and window framing. Write, in English, queries of each class over the Northwind data warehouse.
- **5.10** Identify the kind of hierarchies that can be directly represented in Analysis Services.
- **5.11** Identify the kind of hierarchies that can be directly represented in Mondrian.
- **5.12** Discuss how snowflake schemas are represented in Analysis Services and in Mondrian.

Exercises

- **5.1** Consider the data warehouse of a telephone provider given in Ex. 3.1. Draw a star schema diagram for the data warehouse.
- **5.2** For the star schema obtained in the previous exercise, write in SQL the queries given in Ex. 3.1.
- **5.3** Consider the data warehouse of the train application given in Ex. 3.2. Draw a snowflake schema diagram for the data warehouse with hierarchies for the train and station dimensions.
- **5.4** For the snowflake schema obtained in the previous exercise, write in SQL the queries given in Ex. 3.2.

- **5.5** Consider the university data warehouse described in Ex. 3.3. Draw a constellation schema for the data warehouse taking into account the different granularities of the time dimension.
- **5.6** For the constellation schema obtained in the previous exercise, write in SQL the queries given in Ex. 3.3.
- **5.7** Translate the MultiDim schema obtained for the French horse race application in Ex. 4.5 into the relational model.
- **5.8** Translate the MultiDim schema obtained for the Formula One application in Ex. 4.7 into the relational model.

- **5.9** The Research and Innovative Technology Administration (RITA) coordinates the US Department of Transportation's (DOT) research programs. It collects several statistics about many kinds of transportation means, including the information about flight segments between airports summarized by month. There is a set of tables T T100I Segment All Carrier XXXX, one by year, ranging from 1990 up until now. These tables include information about the scheduled and actually departured flights, the number of seats sold, the freight transported, and the distance traveled, among other ones. The schema and description of these tables is given in Table 5.1. A set of lookup tables given in Table 5.2 include information about airports, carriers, and time. The schemas of these lookup tables are composed of just two columns called Code and Description. The mentioned web site describes all tables in detail.

From the information above, construct an appropriate data warehouse schema. Analyze the input data and motivate the choice of your schema.

- **5.10** Implement in Analysis Services the MultiDim schema obtained for the French horse race application in Ex. 4.5 and the relational data warehouse obtained in Ex. 5.7.
- **5.11** Implement in Mondrian the MultiDim schema obtained for the Formula One application in Ex. 4.7 and the relational data warehouse obtained in Ex. 5.8.

Summaries	
DepScheduled	Departures scheduled
DepPerformed	Departures performed
Payload	Available payload (pounds)
Seats	Available seats
Passengers	Non-stop segment passengers transported
Freight	Non-stop segment freight transported (pounds)
Mail	Non-stop segment mail transported (pounds)
Distance	Distance between airports (miles)
RampTime	Ramp to ramp time (minutes)
AirTime	Airborne time (minutes)

Carrier	
UniqueCarrier	Unique carrier code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years
AirlineID	An identification number assigned by US DOT to identify a unique airline (carrier). A unique airline (carrier) is defined as one holding and reporting under the same DOT certificate regardless of its code, name, or holding company/corporation
UniqueCarrierName	Unique carrier name. When the same name has been used by multiple carriers, a numeric suffix is used for earlier users, for example, Air Caribbean, Air Caribbean (1)
UniqCarrierEntity	Unique entity for a carrier's operation region
CarrierRegion	Carrier's operation region. Carriers report data by operation region
Carrier	Code assigned by IATA and commonly used to identify a carrier. As the same code may have been assigned to different carriers over time, the code is not always unique. For analysis, use the unique carrier code
CarrierName	Carrier name
CarrierGroup	Carrier group code. Used in legacy analysis
CarrierGroupNew	Carrier group new

Origin	
OriginAirportID	Origin airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused
OriginAirportSeqID	Origin airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time
OriginCityMarketID	Origin airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market
Origin	Origin airport
OriginCityName	Origin city
OriginCountry	Origin airport, country
OriginCountryName	Origin airport, country name
OriginWAC	Origin airport, world area code

Destination	
DestAirportID	Destination airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused
DestAirportSeqID	Destination airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time
DestCityMarketID	Destination airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market
Dest	Destination airport
DestCityName	Destination city
DestCountry	Destination airport, country
DestCountryName	Destination airport, country name
DestWAC	Destination airport, world area code

Aircraft	
AircraftGroup	Aircraft group
AircraftType	Aircraft type
AircraftConfig	Aircraft configuration
Time Period	
Year	Year
Quarter	Quarter
Month	Month
Other	
DistanceGroup	Distance intervals, every 500 Miles, for flight segment
Class	Service Class

Enjoy the Course…!