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 4: Conceptual Data Warehouse Design

1. MultiDim: A Conceptual Model for Data Warehouses

2. Dimension Hierarchies

3. Advanced Modeling Aspects

4. Querying the Northwind Cube

Table of Contents

1. MultiDim: A Conceptual Model for Data Warehouses

2. Dimension Hierarchies

3. Advanced Modeling Aspects

4. Querying the Northwind Cube

Conceptual Multidimensional Models

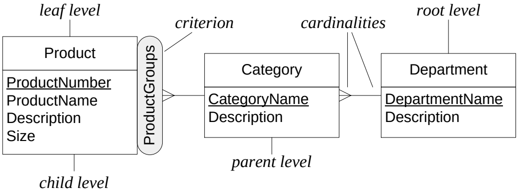
- **Conceptual models**
  - Allow better communication between designers and users to understand application requirements
  - More stable than implementation-oriented (logical) schema, which changes with the platform
  - Provide better support for visual user interfaces
- No well-established conceptual model for multidimensional data
- Several proposals based on UML, on the ER model, or using specific notations
- Problems:
  - Cannot express complex kinds of hierarchies
  - Lack of a mapping to the implementation platform
- Currently, data warehouses are designed using mostly logical models (star and snowflake schemas)
  - Difficult to express requirements (technical knowledge required)
  - Limit users to defining only elements that the underlying implementation systems can manage
  - Example: Users constrained to use only the simple hierarchies supported in current tools

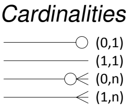
MultiDim: A Conceptual Multidimensional Model

- Based on the entity-relationship model
- Includes concepts like:
  - **dimensions**
  - **hierarchies**
  - **facts**
  - **measures**
- Supports various kinds of hierarchies existing in real-world applications
- Can be mapped to star or snowflake relational structures

MultiDim Model: Notation

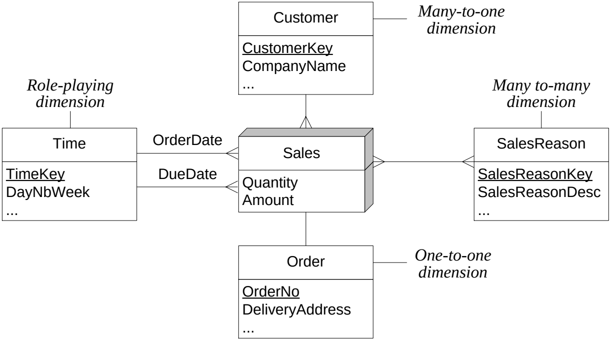
- **Dimension**: level or one or more hierarchies
- **Hierarchy**: several related levels
- **Level**: entity type
- **Member**: every instance of a level
- **Child** and **parent** levels: the lower and higher levels
- **Leaf** and **root** levels: first and last levels in a hierarchy
- **Cardinality**: Minimum/maximum numbers of members in a level related to members in another level
- **Criterion**: Expresses different hierarchical structures used for analysis
- **Key attribute**: Indicates how child members are grouped
- **Descriptive attributes**: Describe characteristics of members





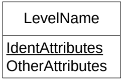
MultiDim Model: Notation

- **Fact**: Relates measures to leaf levels in dimensions
- Dimensions can be related to fact with **one-to-one**, **one-to-many**, of **many-to-many**
- Dimension can be related several times to a fact with **different roles**

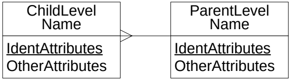


MultiDim Model: Notation (Summary)

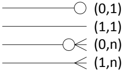
Level



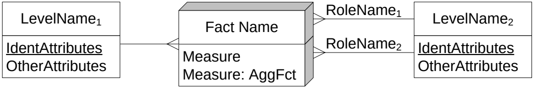
Hierarchy



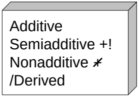
Cardinalities



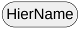
Fact with measures and associated levels



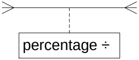
Types of measures



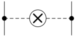
Analysis criterion



Distributing factor



Exclusive relationships



MultiDim Conceptual Schema of the Northwind Data Warehouse

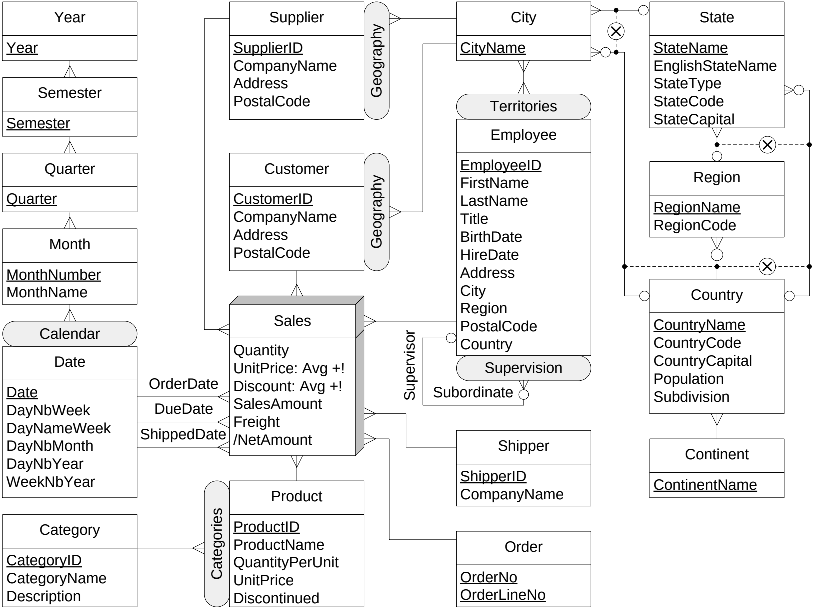


Table of Contents

1. MultiDim: A Conceptual Model for Data Warehouses

2. Dimension Hierarchies

3. Advanced Modeling Aspects

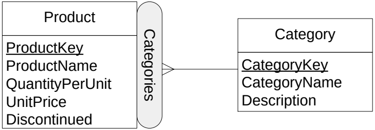
4. Querying the Northwind Cube

Dimension Hierarchies

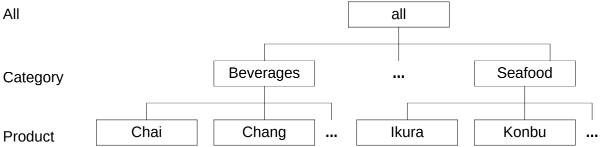
- Crucial in analytical applications
- Enable analysis at various abstraction levels
- In real-world situations, users must deal with complex hierarchies of various kinds
- Logical models of current DW and OLAP systems allow only a limited set of kinds of hierarchies
  - Users unable to capture the essential semantics of multidimensional applications
  - They must limit their analysis to the predefined set of hierarchies supported by the tools
- At the conceptual level, focus is to establish sequences of levels that should be traversed during roll-up and drill-down
- Distinction between the various kinds of hierarchies should also be made at the instance level
- Cardinalities in parent-child relationships must be considered
- MultiDim includes classification of hierarchies at the schema and instance level and proposes a graphical notation

Balanced Hierarchies

- At **schema level**: only one path where all parent-child relationships are many-to-one and mandatory

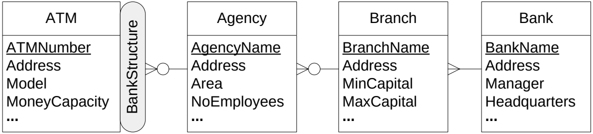


- At **instance level**: members form a balanced tree (all the branches have the same length)
- All parent members have at least one child member, and a child belongs exactly to one parent

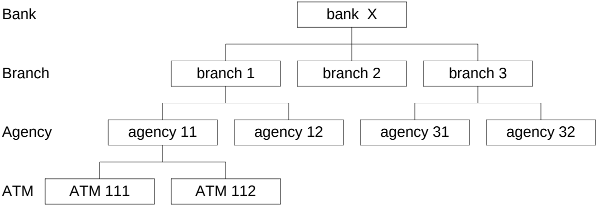


Unbalanced Hierarchies

- At **schema level**: one path where all parent-child relationships are many-to-one, but some are optional

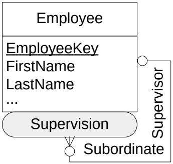


- At **instance level**: members form a unbalanced tree



Recursive Hierarchies

- A special case of unbalanced hierarchies
- The **same level** is linked by the two roles of a parent-child relationship
- Used when all hierarchy levels express the same semantics
- The characteristics of the parent and child are similar (or the same)
- **Schema level**



Employee

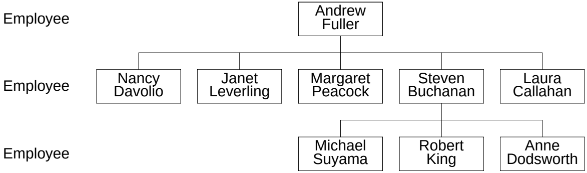
EmployeeKey
FirstName
LastName
...

Supervision

Supervisor

Subordinate

- **Instance level**



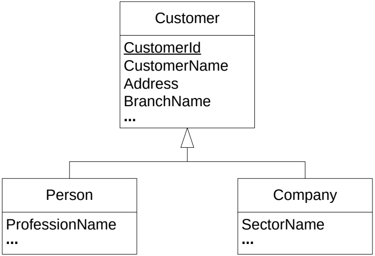
Employee — Andrew Fuller

Employee — Nancy Davolio, Janet Leverling, Margaret Peacock, Steven Buchanan, Laura Callahan

Employee — Michael Suyama, Robert King, Anne Dodsworth

Generalized Hierarchies

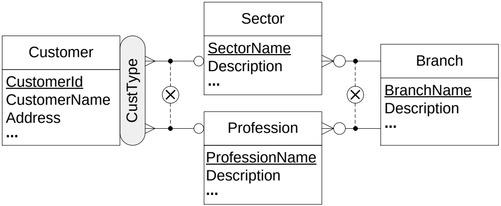
- Sometimes the members of a hierarchy are of different type
- Entity-relationship representation of customer types



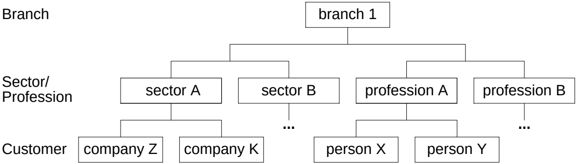
- Hierarchical paths cannot be clearly distinguished
- Higher hierarchy levels (e.g., Branch) in a supertype not related to other hierarchy levels in its subtypes
- Not clear that measures related to a customer that is a person can be aggregated using a hierarchy Customer–Profession–Branch

Generalized Hierarchies

- At **schema level**: multiple exclusive paths sharing at least the leaf level; may also share other levels
- Two aggregation paths, one for each type of customer



- At **instance level**: each member belongs to only one path

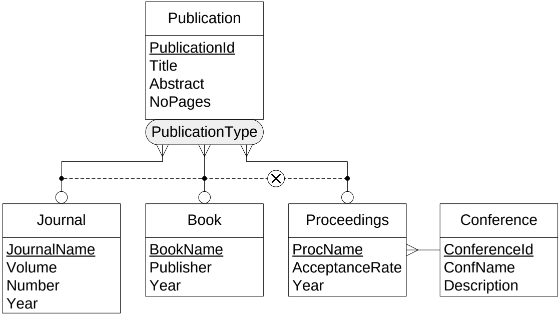


Generalized Hierarchies

- **Supertype** of the generalization/specialization relationship is used in generalized hierarchies for representing a **leaf level**
- It only includes those attributes that represent concepts at the lowest granularity
  - E.g., CustomerId, CustomerName, and Address
- This kind of hierarchy **does not satisfy the summarizability conditions**
  - The mapping from the splitting level to the parent levels is incomplete
    - E.g., not all customers roll up to the Sector level
    - E.g., not all customers are mapped to the Profession level
- Conventional aggregation mechanism should be modified when a splitting and joining levels are reached in a drill-down and roll-up operations
- Traditional approach can be used for aggregating measures for common hierarchy levels

Generalized Hierarchies

- In generalized hierarchies, it is not necessary that splitting levels must be joined

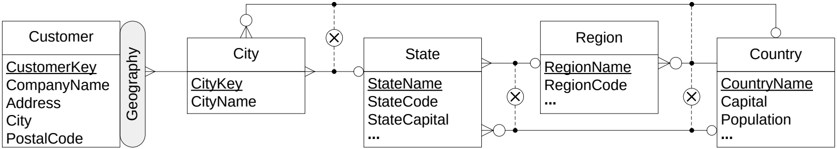


Generalized Hierarchies

- Not all generalization/specialization hierarchies can be represented
- **Partial specializations**: Induce an additional path in the generalized hierarchy, relating the common levels
- **Overlapping specializations**: Various options are possible according to the users' requirements and the availability of measures
  - Example: An overlapping generalization where a person who owns a company buys products either for his/her individual use or for the company
  - If measures are known only for the superclass Customer, only the hierarchy with common levels will be represented, e.g., the Customer and Area levels
  - If measures are known only for each subclass, e.g., for Person and Company:
    - Separate dimensions and fact relationships with corresponding measures can be created for each specialization → difficult to manage dimensions with overlapping sets of members
    - Another solution: Disallow overlapping generalizations

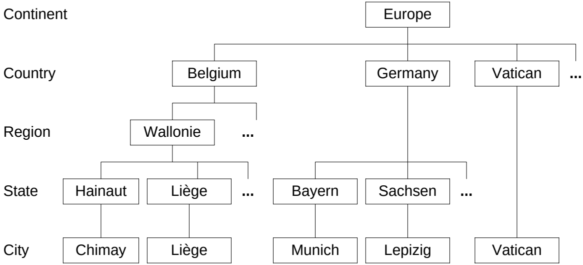
Noncovering Hierarchies

- Also known as **ragged** or **level-skipping hierarchies**
- A **special case of generalized hierarchies**
- At the **schema level**: Alternative paths are obtained by skipping one or several intermediate levels



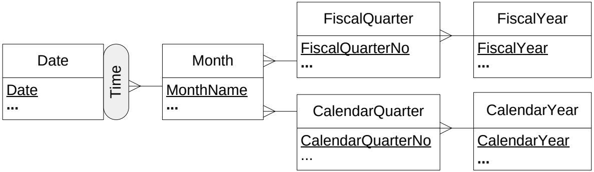
Noncovering Hierarchies

- At **instance level**: Path length from the leaves to the same parent can be different for different members

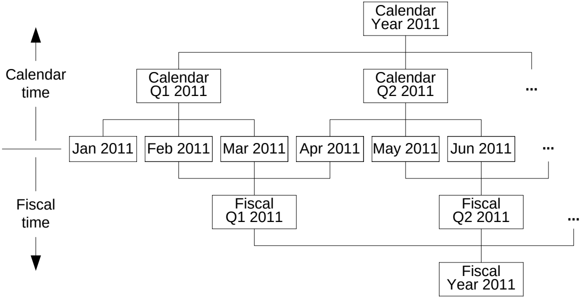


Alternative Hierarchies

- At **schema level**: Multiple nonexclusive hierarchies that share at least the leaf level and account for the same analysis criterion



- At **instance level**: Members form graph

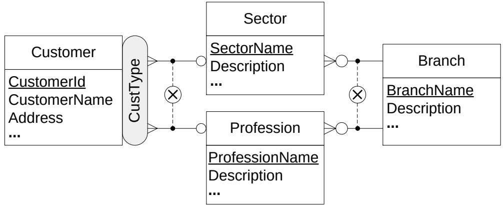


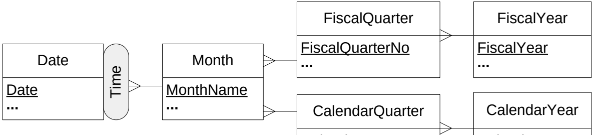
Alternative Hierarchies

- Needed to analyze measures from an unique perspective (e.g., time) using alternative paths
- Measures will participate totally in each component hierarchy ⇒ conventional aggregation procedures
- It is not semantically correct to simultaneously combine different component hierarchies
- Combination can give meaningless intersections, i.e., a combination of members that do not have values for aggregated measures, e.g., B1-2001 and Q2-2001
- Users must choose only one of the alternative paths for their analysis and switch to other one if required

Generalized vs. Alternative Hierarchies

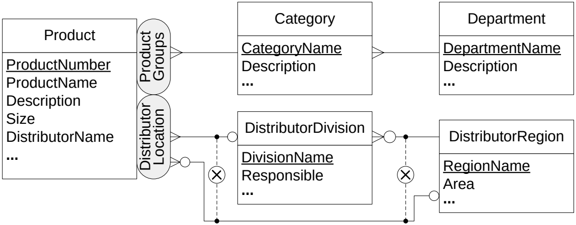
- Both hierarchies
  - Share some levels
  - Use one analysis criterion
- A child member
  - Related to only one path in generalized hierarchies
  - Related to all paths in alternative hierarchies and users must choose one for analysis





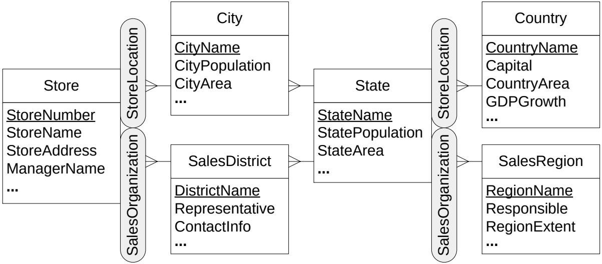
Parallel Hierarchies

- Dimension has associated **several hierarchies** accounting for **different analysis criteria**
- Two different types
  - Parallel **independent** hierarchies
  - Parallel **dependent** hierarchies
- Parallel **independent** hierarchies
  - Composed of disjoint hierarchies, i.e., hierarchies that **do not share levels**
  - Component hierarchies may be of different kinds



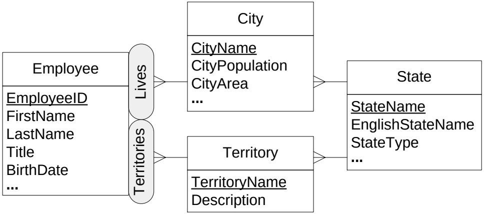
Parallel Hierarchies

- Parallel **dependent** hierarchies
- Composed of several hierarchies that account for different analysis criteria and **share some levels**
- Component hierarchies may be of different kinds



Parallel Hierarchies

- Parallel dependent hierarchies leading to different parent members of the shared level



Employee

EmployeeID
FirstName
LastName
Title
BirthDate
...

Lives

Territories

City

CityName
CityPopulation
CityArea
...

Territory

TerritoryName
Description

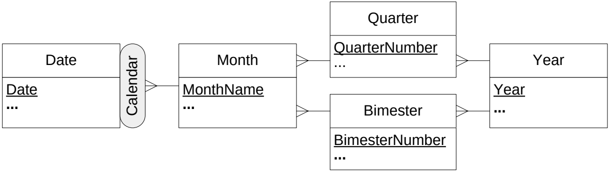
State

StateName
EnglishStateName
StateType
...

Alternative vs. Parallel Hierarchies

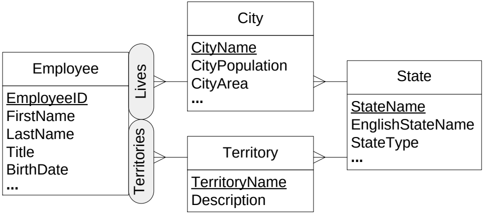
- Both hierarchies
  - Share some levels
  - May include several simple hierarchies
- Criterion
  - Only one for alternative hierarchies
  - Several for parallel hierarchies
- Combining hierarchies
  - Meaningless for alternative hierarchies
  - Useful for parallel hierarchies
- Reusing aggregated measures for common levels
  - Can be done for alternative hierarchies
  - Cannot be done for parallel hierarchies

Alternative vs. Parallel Hierarchies



- Aggregated measure for the Month level can be reused between both paths
- Traversing the Calendar hierarchy from a specific day in the Date level will end up in the same year independently of which path is used

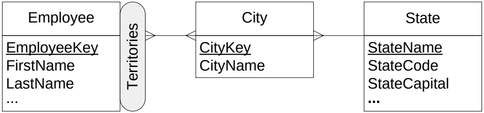
Alternative vs. Parallel Hierarchies



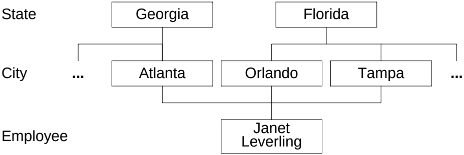
- Aggregated measure for State level cannot be reused between both paths
- Traversing the hierarchies Live and Work from the Employee to the State level will lead to different states for employees who live in one state and work in another

Nonstrict Hierarchies

- At **schema level**: At least one many-to-many cardinality



- At **instance level**: Members form a graph



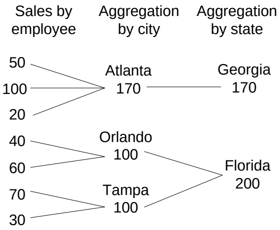
Dimension Hierarchies

- **Nonstrict Hierarchies**
- Abuse of terminology: “nonstrict hierarchy" for “acyclic classification graph"
- The term “hierarchy" conveys the notion that users need to analyze measures at different levels of detail, which is less clear with the term “acyclic classification graph"
- The term “hierarchy" is already used by practitioners and some tools, in particular SQL Server Analysis Services, allow many-to-many parent-child relationships
- The term “hierarchy" is also used by several researchers

Nonstrict Hierarchies: Double Counting

- Problem: **Double counting** of measures when a roll-up operation reaches a many-to-many relationship
- Examples of aggregation

Strict hierarchy



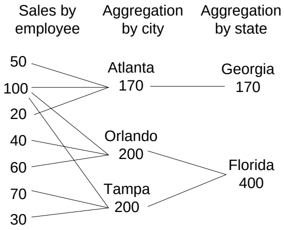
Sales by employee | Aggregation by city | Aggregation by state

50, 100, 20 → Atlanta 170 → Georgia 170

40, 60 → Orlando 100 → Florida 200

70, 30 → Tampa 100 → Florida 200

Nonstrict hierarchy



Sales by employee | Aggregation by city | Aggregation by state

50, 100, 20 → Atlanta 170 → Georgia 170

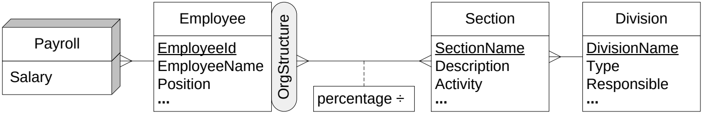
100, 40, 60 → Orlando 200 → Florida 400

100, 70, 30 → Tampa 200 → Florida 400

Nonstrict Hierarchies: Solutions for Double Counting

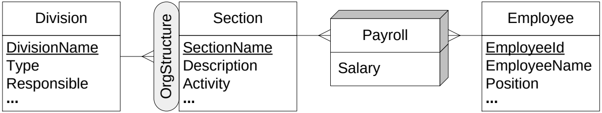
- Include a **distributing factor**
- Calculate **approximate** values of a distributing factor
- **Transform** a nonstrict hierarchy into a strict one:
  - **Create a new parent member** for each group of parent members linked to a single child member in a many-to-many relationship
  - **Choose one parent member as primary** and ignore the existence of other parent members
  - **Split the hierarchy in two** at the many-to-many relationship, where the levels from the parent level and beyond become a new dimension
- Each solution has its advantages and disadvantages and requires special aggregation procedures
- Appropriate solution must be chosen according to the situation at hand and user's requirements

Nonstrict Hierarchies: Distributing Factor



- Employees may work in several sections
- A measure represents an employee's overall salary, i.e., the sum of the salaries paid in each section
- Distributing factor determines how measures are divided between several parent members
- Distributing factor is **not always known**
  - Percentage of time that an employee works in a section must be added to schema
- Sometimes this distribution is **impossible to specify**
  - E.g., participation of customer in joint account
- Distributing factor can be **approximated** by considering the total number of parent members with which the child member is associated
  - If an employee works in three sections, 1/3 of the value of the measure aggregated for each one

Nonstrict Hierarchies: Splitting the Hierarchy



- Transform a nonstrict hierarchy into a strict one with an **additional dimension**
- **Focus of analysis** has changed from employee's salaries to employee's salaries by section
- Can only be applied when the **measure distribution is known**
- Nevertheless, double counting problem still remains
- Example: calculate the number of employees by section or by division

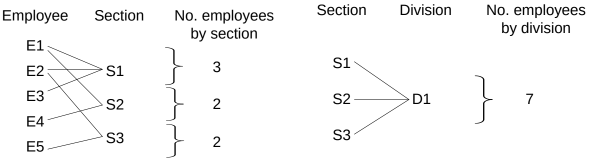


Table of Contents

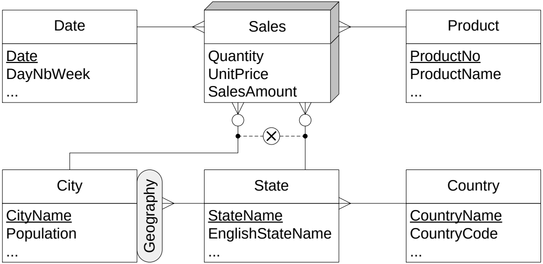
1. MultiDim: A Conceptual Model for Data Warehouses

2. Dimension Hierarchies

3. Advanced Modeling Aspects

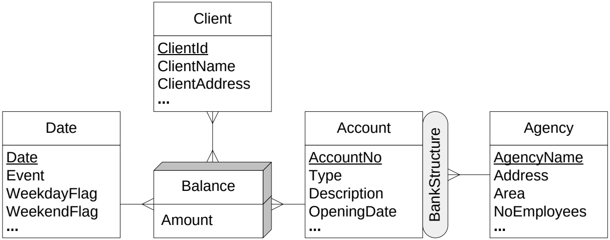
4. Querying the Northwind Cube

Advanced Modeling Aspects: Facts with Multiple Granularities



- Sales captured at the city level or at the state level

Advanced Modeling Aspects: Many-to-Many Dimensions



- Multidimensional schema for the analysis of bank accounts

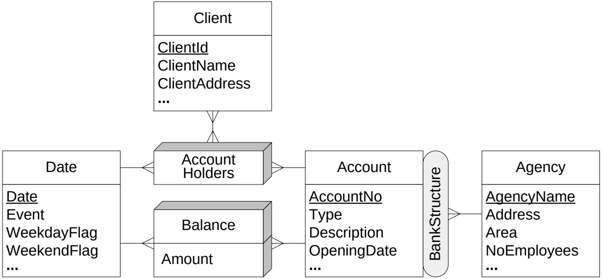
Advanced Modeling Aspects: Many-to-Many Dimensions

- Example of double-counting problem

Time	Account	Client	Balance
T1	A1	C1	100
T1	A1	C2	100
T1	A1	C3	100
T1	A2	C1	500
T1	A2	C2	500

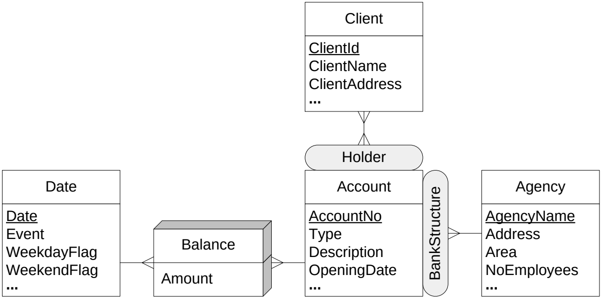
Advanced Modeling Aspects: Many-to-Many Dimensions

- Two possible decompositions of the fact
  1. Creating two facts



Advanced Modeling Aspects: Many-to-Many Dimensions

- Two possible decompositions of the fact
  2. Including a nonstrict hierarchy



Advanced Modeling Aspects: Many-to-Many Dimensions

- Alternative decomposition of the schema

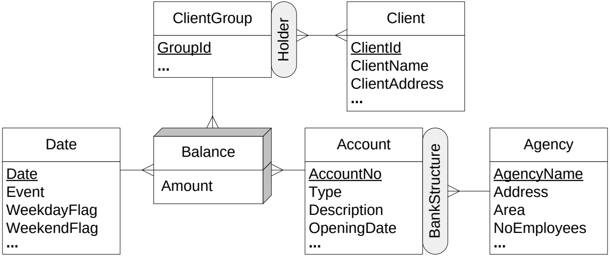
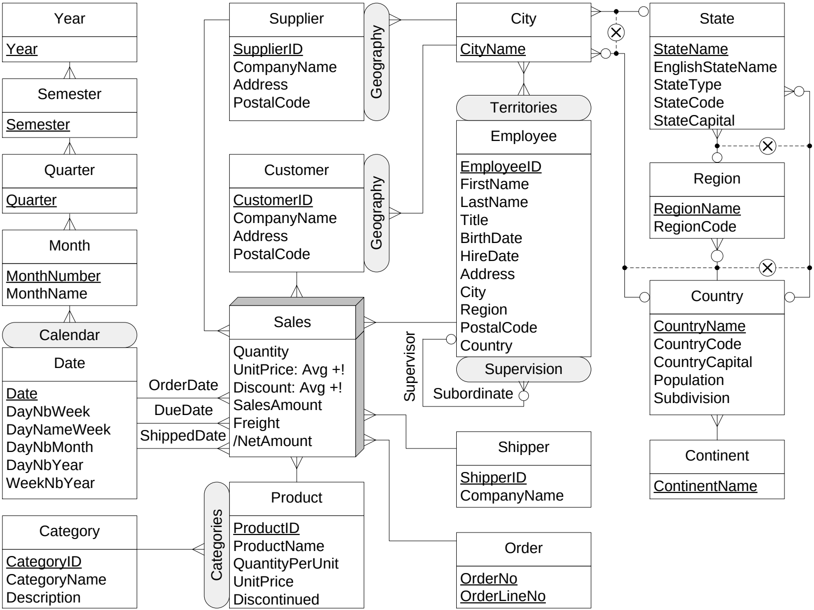


Table of Contents

1. MultiDim: A Conceptual Model for Data Warehouses
2. Dimension Hierarchies
3. Advanced Modeling Aspects
4. Querying the Northwind Cube

Conceptual Schema of the Northwind Cube



Year
Year

Semester
Semester

Quarter
Quarter

Month
MonthNumber
MonthName

Calendar

Date
Date
DayNbWeek
DayNameWeek
DayNbMonth
DayNbYear
WeekNbYear

Category
CategoryID
CategoryName
Description

Supplier
SupplierID
CompanyName
Address
PostalCode

Geography

Customer
CustomerID
CompanyName
Address
PostalCode

Geography

Sales
Quantity
UnitPrice: Avg +!
Discount: Avg +!
SalesAmount
Freight
/NetAmount

OrderDate
DueDate
ShippedDate

Categories

Product
ProductID
ProductName
QuantityPerUnit
UnitPrice
Discontinued

City
CityName

Territories

Employee
EmployeeID
FirstName
LastName
Title
BirthDate
HireDate
Address
City
Region
PostalCode
Country

Supervision
Supervisor
Subordinate

Shipper
ShipperID
CompanyName

Order
OrderNo
OrderLineNo

State
StateName
EnglishStateName
StateType
StateCode
StateCapital

Region
RegionName
RegionCode

Country
CountryName
CountryCode
CountryCapital
Population
Subdivision

Continent
ContinentName

Querying the Northwind Cube

- **Query 4.1**: *Total sales amount per customer, year, and product category*

  ROLLUP*(Sales, Customer → Customer, OrderDate → Year, Product → Category,
  SUM(SalesAmount))

- **Query 4.2**: *Yearly sales amount per pair customer country and supplier country*

  ROLLUP*(Sales, OrderDate → Year, Customer → Country, Supplier → Country,
  SUM(SalesAmount))

- **Query 4.3**: *Monthly sales by customer state compared to those of the previous year*

  Sales1 ← ROLLUP*(Sales, OrderDate → Month, Customer → State, SUM(SalesAm
  Sales2 ← RENAME(Sales1, SalesAmount ← PrevYearSalesAmount)
  Result ← DRILLACROSS(Sales2, Sales1,
  Sales2.OrderDate.Month = Sales1.OrderDate.Month AND
  Sales2.OrderDate.Year+1 = Sales1.OrderDate.Year AND
  Sales2.Customer.State = Sales1.Customer.State)

Querying the Northwind Cube

- **Query 4.4**: *Total sales growth per month per product, that is, total sales per product compared to the previous month*

```
Sales1← ROLLUP*(Sales, OrderDate → Month, Product → Product, SUM(SalesAm
Sales2← RENAME(Sales1, SalesAmount ← PrevMonthSalesAmount)
Sales3← DRILLACROSS(Sales2, Sales1,
        ( Sales2.OrderDate.Month > 1 AND
        Sales2.OrderDate.Month+1 = Sales1.OrderDate.Month AND
        Sales2.OrderDate.Year = Sales1.OrderDate.Year ) OR
        ( Sales2.OrderDate.Month = 1 AND
        Sales1.OrderDate.Month = 12 AND
        Sales2.OrderDate.Year+1 = Sales1.OrderDate.Year )
Result ← ADDMEASURE(Sales3, SalesGrowth = SalesAmount - PrevMonthSalesA
```

Querying the Northwind Cube

- **Query 4.5**: *Top three best-selling employees*

  Sales1 ← ROLLUP*(Sales, Employee → Employee, SUM(SalesAmount))
  Result ← MAX(Sales1, SalesAmount, 3)

- **Query 4.6**: *Best selling employees per product per year*

  Sales1 ← ROLLUP*(Sales, Employee → Employee, Product → Product, OrderDate
  SUM(SalesAmount))
  Result ← MAX(Sales1, SalesAmount) BY Product, OrderDate

- **Query 4.7**: *Countries that account for top 50% of sales amount*

  Sales1 ← ROLLUP*(Sales, Customer → Country, SUM(SalesAmount))
  Sales2 ← SORT(Sales1, Customer, SalesAmount DESC)
  Result ← TOPPERCENT(Sales2, Customer, SalesAmount, 50)

Querying the Northwind Cube

- **Query 4.8**: *Total sales and average monthly sales by employee and year*

  Sales1 ← ROLLUP*(Sales, Employee → Employee, OrderDate → Month, SUM(Sale
  Result ← ROLLUP*(Sales1, Employee → Employee, OrderDate → Year,
  SUM(SalesAmount), AVG(SalesAmount))

- **Query 4.9**: *Total sales amount and total discount amount per product and month*

  Sales1 ← ROLLUP*(Sales, Product → Product, OrderDate → Month, SUM(SalesAm
  Result ← ADDMEASURE(Sales1, TotalDisc = Discount * Quantity * UnitPrice)

Querying the Northwind Cube

- **Query 4.10**: *Monthly year-to-date sales for each product category*

  ```
  Sales1 ← ROLLUP*(Sales, Product → Category, OrderDate → Month, SUM(SalesA
  Result ← ADDMEASURE(Sales1, YTD = SUM(SalesAmount) OVER Date.Year
           ALL CELLS PRECEDING)
  ```

- **Query 4.11**: *Three-month moving average of the sales amount by product category*

  ```
  Sales1 ← ROLLUP*(Sales, Product → Category, OrderDate → Month, SUM(SalesA
  Result ← ADDMEASURE(Sales1, MovAvg3M = AVG(SalesAmount) OVER Date
           2 CELLS PRECEDING)
  ```

- **Query 4.12**: *Total sales amount made by an employee and his/her subordinates during 2017*

  ```
  Sales1 ← ROLLUP*(Sales, Employee → Employee, OrderDate → Year, SUM(Sales
  Sales2 ← SLICE(Sales1, OrderDate.Year = 2017)
  Result ← RECROLLUP(Sales2, Employee → Employee, SUM(SalesAmount))
  ```

Querying the Northwind Cube

- **Query 4.13**: *Total sales amount, number of products, and number of units sold (i.e., the sum of the quantities) by order*

  ROLLUP*(Sales, Order → Order, SUM(SalesAmount), COUNT(Product) AS CountP

- **Query 4.14**: *Total number of orders, total sales amount, and average sales amount by order, all by month*

  Sales1 ← ROLLUP*(Sales, OrderDate → Month, Order → Order, SUM(SalesAmoun
  Result ← ROLLUP*(Sales1, OrderDate → Month, SUM(SalesAmount),
  AVG(SalesAmount) AS AvgSales, COUNT(Order) AS CountOrders)

- **Query 4.15**: *Number of cities and number of states assigned to each employee*

  ROLLUP*(Employee, Employee → State, COUNT(City), COUNT(State))