Data Warehouse Systems: Design and Implementation

Second Edition

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

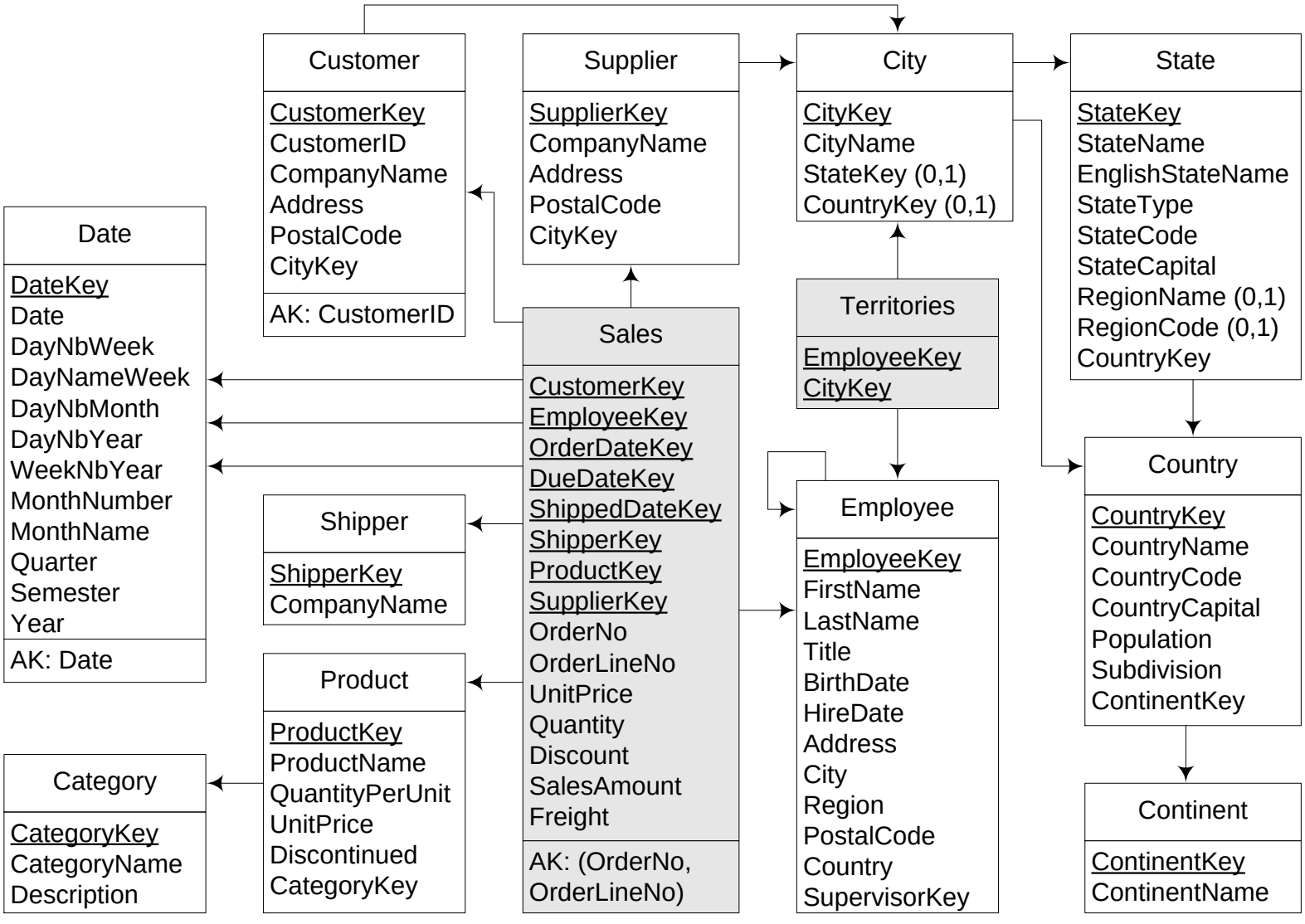
Esteban ZIMÁNYI
Department of Computer & Decision Engineering (CoDE)
Université libre de Bruxelles
esteban.zimanyi@ulb.be

Chapter 7: Data Analysis in the Northwind Data Warehouse

1. Querying the Northwind Cube in SQL

Table of Contents

Schema of the Northwind Data Warehouse



Querying the Northwind Cube in SQL

- Views for enabling month-related calculations

```
CREATE VIEW YearMonth AS
        SELECT DISTINCT Year, MonthNumber, MonthName
        FROM    Date
```

- YearMonth contains all the year and months from the Date dimension

```
CREATE VIEW YearMonthPM AS
        WITH YearMonthPrevMonth AS (
                SELECT Year, MonthNumber, MonthName,
                        LAG(Year * 100 + MonthNumber) OVER (ORDER BY
                        Year * 100 + MonthNumber) AS PrevMonth
                FROM    YearMonth )
        SELECT Year, MonthNumber, MonthName, PrevMonth / 100 AS PM_Year,
                PrevMonth % 100 AS PM_MonthNumber
        FROM    YearMonthPrevMonth
```

- YearMonthPM associates with each year and month in the Date dimension the same month of the previous year
- YearMonthPrevMonth uses the LAG window function to associate each month with the previous one represented by a numerical expression in the format YYYYMM
- For example, the expression associated with January 2017 would be 201612
- Main query splits this expression into the year and the month

Querying the Northwind Data Warehouse in SQL

- **Query 7.1**: *Total sales amount per customer, year, and product category*

```
SELECT    C.CompanyName, D.Year, A.CategoryName,
          SUM(SalesAmount) AS SalesAmount
FROM      Sales S, Customer C, Date D, Product P, Category A
WHERE     S.CustomerKey = C.CustomerKey AND
          S.OrderDateKey = D.DateKey AND
          S.ProductKey = P.ProductKey AND
          P.CategoryKey = A.CategoryKey
GROUP BY C.CompanyName, D.Year, A.CategoryName
```

- We join the fact tables with the involved dimension tables, and aggregate the results by company, year, and category

Querying the Northwind Cube in SQL

- **Query 7.2**: *Yearly sales amount for each pair of customer country and supplier countries*

```
SELECT    CO.CountryName AS Country, SO.CountryName AS Country,
          D.Year, SUM(SalesAmount) AS SalesAmount
FROM      Sales F, Customer C, City CC, State CS, Country CO,
          Supplier S, City SC, State SS, Country SO, Date D
WHERE     F.CustomerKey = C.CustomerKey AND C.CityKey = CC.CityKey AND
          CC.StateKey = CS.StateKey AND CS.CountryKey = CO.CountryKey AND
          F.SupplierKey = S.SupplierKey AND S.CityKey = SC.CityKey AND
          SC.StateKey = SS.StateKey AND SS.CountryKey = SO.CountryKey AND
          F.OrderDateKey = D.DateKey
GROUP BY  CO.CountryName, SO.CountryName, D.Year
ORDER BY  CO.CountryName, SO.CountryName, D.Year
```

- The tables of the geography dimension are joined twice with the fact table for obtaining the countries of the customer and the supplier

Querying the Northwind Data Warehouse in SQL

- **Query 7.3**: *Monthly sales by customer state compared to those of the previous year*

```
WITH StateMonth AS (
        SELECT   DISTINCT StateName, Year, MonthNumber, MonthName
        FROM     Customer C, City Y, State S, YearMonth M
        WHERE    C.CityKey = Y.CityKey AND Y.StateKey = S.StateKey ),
    SalesStateMonth AS (
        SELECT   StateName, Year, MonthNumber, SUM(SalesAmount) AS SalesAmount
        FROM     Sales F, Customer C, City Y, State S, Date D
        WHERE    F.CustomerKey = C.CustomerKey AND C.CityKey = Y.CityKey AND
                 Y.StateKey = S.StateKey AND F.OrderDateKey = D.DateKey
        GROUP BY S.StateName, D.Year, D.MonthNumber )
SELECT   S.StateName, S.MonthName, S.Year, M1.SalesAmount,
         M2.SalesAmount AS SalesAmountPY
FROM     StateMonth S LEFT OUTER JOIN SalesStateMonth M1 ON
         S.Year = M1.Year AND S.MonthNumber = M1.MonthNumber
         LEFT OUTER JOIN SalesStateMonth M2 ON S.StateName = M2.StateName AND
         S.Year - 1 = M2.Year AND S.MonthNumber = M2.MonthNumber
WHERE    M1.SalesAmount IS NOT NULL OR M2.SalesAmount IS NOT NULL
ORDER BY S.StateName, S.Year, S.MonthNumber
```

- StateMonth makes the Cartesian product of the all the customer states and all months in YearMonth
- SalesStateMonth computes the monthly sales by state
- Main query performs two left outer joins of these tables to compute the result

Querying the Northwind Cube in SQL

- **Query 7.4**: *Monthly sales growth per product, that is, total sales per product compared to those of the previous month*

```
WITH ProdYearMonthPM AS (
        SELECT   DISTINCT ProductName, Year, MonthNumber, MonthName,
                 PM_Year, PM_MonthNumber
        FROM     Product, YearMonthPM ),
     SalesProdMonth AS (
        SELECT   ProductName, Year, MonthNumber,
                 SUM(SalesAmount) AS SalesAmount
        FROM     Sales S, Product P, Date D
        WHERE    S.ProductKey = P.ProductKey AND S.OrderDateKey = D.DateKey
        GROUP BY ProductName, Year, MonthNumber )
SELECT   P.ProductName, P.MonthName, P.Year, S1.SalesAmount,
         S2.SalesAmount AS SalesPrevMonth, COALESCE(S1.SalesAmount,0) -
         COALESCE(S2.SalesAmount,0) AS SalesGrowth
FROM     ProdYearMonthPM P LEFT OUTER JOIN SalesProdMonth S1 ON
         P.ProductName = S1.ProductName AND P.Year = S1.Year AND
         P.MonthNumber = S1.MonthNumber LEFT OUTER JOIN SalesProdMonth S2
         ON P.ProductName = S2.ProductName AND P.PM_Year = S2.Year AND
         P.PM_MonthNumber = S2.MonthNumber
ORDER BY ProductName, P.Year, P.MonthNumber
```

- ProdYearMonthPM: Cartesian product of Product with YearMonthPM
- SalesProdMonth: monthly sales by product
- Main query performs two left outer joins of these tables to compute the result

Querying the Northwind Cube in SQL

- **Query 7.5**: *Three best-selling employees*

```
SELECT    TOP(3) E.FirstName + ' ' + E.LastName AS EmployeeName,
          SUM(S.SalesAmount) AS SalesAmount
FROM      Sales S, Employee E
WHERE     S.EmployeeKey = E.EmployeeKey
GROUP BY E.FirstName, E.LastName
ORDER BY SalesAmount DESC
```

- Sales by employee are grouped and SUM function is applied to each group
- Result is then sorted in descending order of the aggregated sales and the TOP function is used to obtain the first three tuples

Querying the Northwind Cube in SQL

- **Query 7.6**: *Best-selling employee per product and year*

```
WITH SalesProdYearEmp AS (
                SELECT    P.ProductName, D.Year, SUM(S.SalesAmount) AS SalesAmount,
                          E.FirstName + ' ' + E.LastName AS EmployeeName
                FROM      Sales S, Employee E, Date D, Product P
                WHERE     S.EmployeeKey = E.EmployeeKey AND
                          S.OrderDateKey = D.DateKey AND
                          S.ProductKey = P.ProductKey
                GROUP BY P.ProductName, D.Year, E.FirstName, E.LastName )
SELECT    ProductName, Year, EmployeeName AS TopEmployee,
          SalesAmount AS TopSales
FROM      SalesProdYearEmp S1
WHERE     S1.SalesAmount = (
          SELECT    MAX(SalesAmount)
          FROM      SalesProdYearEmp S2
          WHERE     S1.ProductName = S2.ProductName AND S1.Year = S2.Year )
ORDER BY ProductName, Year
```

- SalesProdYearEmp: yearly sales by product and employee
- Main query selects the tuples of this table such that the total sales equals the maximum total sales for the product and the year

Querying the Northwind Data Warehouse in SQL

- **Query 7.7**: *Countries that account for top 50% of sales amount.*

```
WITH TotalSales AS (
        SELECT   SUM(SalesAmount) AS SalesAmount
        FROM     Sales),
    SalesCountry AS (
        SELECT   CountryName, SUM(SalesAmount) AS SalesAmount
        FROM     Sales S, Customer C, City Y, State T, Country O
        WHERE    S.CustomerKey = C.CustomerKey AND
                 C.CityKey = Y.CityKey AND Y.StateKey = T.StateKey AND
                 T.CountryKey = O.CountryKey
        GROUP BY CountryName ),
    CumulSalesCountry AS (
        SELECT   S.*, SUM(SalesAmount) OVER (ORDER BY SalesAmount DESC
                 ROWS UNBOUNDED PRECEDING) AS CumulSales
        FROM     SalesCountry S )
SELECT   C.CountryName, C.SalesAmount, C.SalesAmount / T.SalesAmount AS
         PercSales, C.CumulSales, C.CumulSales / T.SalesAmount AS CumulPerc
FROM     CumulSalesCountry C, TotalSales T
WHERE    CumulSales <= (
         SELECT   MIN(CumulSales) FROM CumulSalesCountry
         WHERE    CumulSales >= (
                  SELECT 0.5 * SUM(SalesAmount) FROM SalesCountry ) )
ORDER BY SalesAmount DESC
```

- SalesCountry aggregates the sales amount by country
- CumSalesCountry defines a window for each row in SalesCountry, with rows sorted in decreasing value of sales amount, and sum the current row and all the preceding rows in the window
- Main query selects the countries which satisfy the query condition

Querying the Northwind Cube in SQL

- **Query 7.8**: *Total sales and average monthly sales by employee and year*

```
WITH MonthlySalesEmp AS (
                SELECT    E.FirstName + ' ' + E.LastName AS EmployeeName,
                          D.Year, D.MonthNumber,
                          SUM(SalesAmount) AS SalesAmount
                FROM      Sales S, Employee E, Date D
                WHERE     S.EmployeeKey = E.EmployeeKey AND
                          S.OrderDateKey = D.DateKey
                GROUP BY E.FirstName, E.LastName, D.Year, D.MonthNumber )
SELECT    EmployeeName, Year, SUM(SalesAmount) AS SalesAmount,
          AVG(SalesAmount) AS AvgMonthlySales
FROM      MonthlySalesEmp
GROUP BY EmployeeName, Year
ORDER BY EmployeeName, Year
```

- MonthlySalesEmp computes the monthly sales by employee
- Main query groups tuples of this table by employee and year, applies the SUM and AVG functions to obtain the total yearly sales and the average monthly sales

Querying the Northwind Cube in SQL

- **Query 7.9**: *Total sales amount and discount amount per product and month*

```
SELECT    P.ProductName, D.Year, D.MonthNumber,
          SUM(S.SalesAmount) AS SalesAmount,
          SUM(S.UnitPrice * S.Quantity * S.Discount) AS TotalDisc
FROM      Sales S, Date D, Product P
WHERE     S.OrderDateKey = D.DateKey AND S.ProductKey = P.ProductKey
GROUP BY P.ProductName, D.Year, D.MonthNumber
ORDER BY P.ProductName, D.Year, D.MonthNumber
```

- Sales are grouped by product and month
- SUM function computes the total sales and the total discount amount

Querying the Northwind Data Warehouse in SQL

- **Query 7.10**: *Monthly year-to-date sales for each product category*

```
WITH SalesCategoryMonth AS (
        SELECT   CategoryName, Year, MonthNumber, SUM(SalesAmount) AS SalesAmount
        FROM     Sales S, Product P, Category C, Date D
        WHERE    S.OrderDateKey = D.DateKey AND S.ProductKey = P.ProductKey AND
                 P.CategoryKey = C.CategoryKey
        GROUP BY CategoryName, Year, MonthNumber ),
    CategorySales AS (
        SELECT   DISTINCT CategoryName
        FROM     SalesCategoryMonth ),
    CategoryMonth AS (
        SELECT   *
        FROM     CategorySales, YearMonth )
SELECT   C.CategoryName, C.MonthName, C.Year, SalesAmount, SUM(SalesAmount)
         OVER (PARTITION BY C.CategoryName, C.Year ORDER BY
         C.MonthNumber ROWS UNBOUNDED PRECEDING) AS YTDSalesAmount
FROM     CategoryMonth C LEFT OUTER JOIN SalesCategoryMonth S ON
         C.CategoryName = S.CategoryName AND C.Year = S.Year AND C.MonthNumber = S.MonthNumber
ORDER BY CategoryName, Year, C.MonthNumber
```

- Year-to-date is computed over all the months independently of sales
- SalesCategoryMonth: sales amount by category and month
- CategorySales: all categories in the previous table
- CategoryMonth: Cartesian product of CategorySales with YearMonth
- Main query: for each row in the left outer join, defines a window with all rows with the same category and year, orders them by month, and computes the sum of the current and preceding rows

Querying the Northwind Cube in SQL

- **Query 7.11**: *Moving average over the last 3 months of the sales amount by product category*

```
WITH SalesCategoryMonth AS ( ... ),
     CategorySales AS ( ... ),
     CategoryMonth AS ( ... )
SELECT   C.CategoryName, C.MonthName, C.Year, SalesAmount,
         AVG(SalesAmount) OVER (PARTITION BY C.CategoryName ORDER BY
         C.Year, C.MonthNumber ROWS 2 PRECEDING) AS MovAvg3M
FROM     CategoryMonth C LEFT OUTER JOIN SalesCategoryMonth S ON
         C.CategoryName = S.CategoryName AND C.Year = S.Year AND
         C.MonthNumber = S.MonthNumber
ORDER BY CategoryName, Year, C.MonthNumber
```

- Variation of the previous query that defines the same temporary tables
- Main query: for each row in the left outer join, defines a window containing all the rows with the same category, orders them by year and month, and computes the average of the current row and the two preceding ones

Querying the Northwind Cube in SQL

- **Query 7.12**: *Personal sales amount made by an employee compared with the total sales amount made by herself and her subordinates during 2017*

```
WITH Supervision AS (
        SELECT    EmployeeKey, SupervisorKey
        FROM      Employee
        WHERE     SupervisorKey IS NOT NULL
        UNION ALL
        SELECT    E.EmployeeKey, S.SupervisorKey
        FROM      Supervision S, Employee E
        WHERE     S.EmployeeKey = E.SupervisorKey ),
    SalesEmp2017 AS (
        SELECT    EmployeeKey, SUM(S.SalesAmount) AS PersonalSales
        FROM      Sales S, Date D
        WHERE     S.OrderDateKey = D.DateKey AND D.Year = 2017
        GROUP BY EmployeeKey ),
    SalesSubord2017 AS (
        SELECT    SupervisorKey AS EmployeeKey, SUM(S.SalesAmount) AS SubordinateSales
        FROM      Sales S, Supervision U, Date D
        WHERE     S.EmployeeKey = U.EmployeeKey AND S.OrderDateKey = D.DateKey AND D.Year = 2017
        GROUP BY SupervisorKey )
SELECT    FirstName + ' ' + LastName AS EmployeeName, S1.PersonalSales,
          COALESCE(S1.PersonalSales + S2.SubordinateSales, S1.PersonalSales) AS PersSubordSales
FROM      Employee E JOIN SalesEmp2017 S1 ON E.EmployeeKey = S1.EmployeeKey
          LEFT OUTER JOIN SalesSubord2017 S2 ON S1.EmployeeKey = S2.EmployeeKey
ORDER BY EmployeeName
```

- Supervision: recursive query computing the transitive closure of supervision
- SalesEmp2017: total sales by employee
- SalesSubord2017: total sales of the subordinates of an employee
- Main query computes personal sales and subordinates sales
- COALESCE takes into account if an employee has no subordinates

Querying the Northwind Cube in SQL

- **Query 7.13**: *Total sales amount, number of products, and sum of the quantities sold for each order*

```
SELECT    OrderNo, SUM(SalesAmount) AS SalesAmount,
          MAX(OrderLineNo) AS NbProducts, SUM(Quantity) AS Quantity
FROM      Sales
GROUP BY OrderNo
ORDER BY OrderNo
```

- Sales fact table contains both the order number and the order line number
- It is a fact dimension since it is stored in the fact table
- Main query groups the sales by order number and applies the SUM and MAX aggregation functions for obtaining the requested values

Querying the Northwind Cube in SQL

- **Query 7.14**: *For each month, total number of orders, total sales amount, and average sales amount by order*

```
WITH OrderAgg AS (
                SELECT    OrderNo, OrderDateKey,
                          SUM(SalesAmount) AS SalesAmount
                FROM      Sales
                GROUP BY OrderNo, OrderDateKey )
SELECT    Year, MonthNumber, COUNT(OrderNo) AS NoOrders,
          SUM(SalesAmount) AS SalesAmount
          AVG(SalesAmount) AS AvgAmount
FROM      OrderAgg O, Date D
WHERE     O.OrderDateKey = D.DateKey
GROUP BY Year, MonthNumber
ORDER BY Year, MonthNumber
```

- OrderAgg computes the sales amount of each order and skeep the key of the time dimension
- Main query joins the fact and the time dimension tables, groups the tuples by year and month, and computes the aggregate values

Querying the Northwind Data Warehouse in SQL

- **Query 7.15**: *For each employee, total sales amount, number of cities, and number of states to which she is assigned*

```
SELECT FirstName + ' ' + LastName AS FullName,
       SUM(SalesAmount) / COUNT(DISTINCT CityName) AS TotalSales,
       COUNT(DISTINCT CityName) AS NoCities,
       COUNT(DISTINCT StateName) AS NoStates
FROM   Sales F, Employee E, Territories T, City C, State S
WHERE  F.EmployeeKey = E.EmployeeKey AND E.EmployeeKey = T.EmployeeKey AND
       T.CityKey = C.CityKey AND C.StateKey = S.StateKey
GROUP BY FirstName + ' ' + LastName
ORDER BY FirstName + ' ' + LastName
```

- Territories: many-to-many relationship between employees and cities
- Main query joins the tables and groups the result by employee
- SELECT clause divides the sum of SalesAmount by the number of distinct CityName assigned to an employee in the Territories table
- This solves the double-counting problem