

Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de



Chapter 5 – Data Analysis in SQL



Outline

Overview

I. Object-Relational Database Concepts

1. User-defined Data Types and Typed Tables
2. Object-relational Views and Collection Types
3. User-defined Routines and Object Behavior
4. Application Programs and Object-relational Capabilities

II. Online Analytic Processing

5. **Data Analysis in SQL**
6. Windows and Query Functions in SQL

III. XML

7. XML and Databases
8. SQL/XML
9. XQuery

IV. More Developments (if there is time left)

temporal data models, data streams, databases and uncertainty, ...



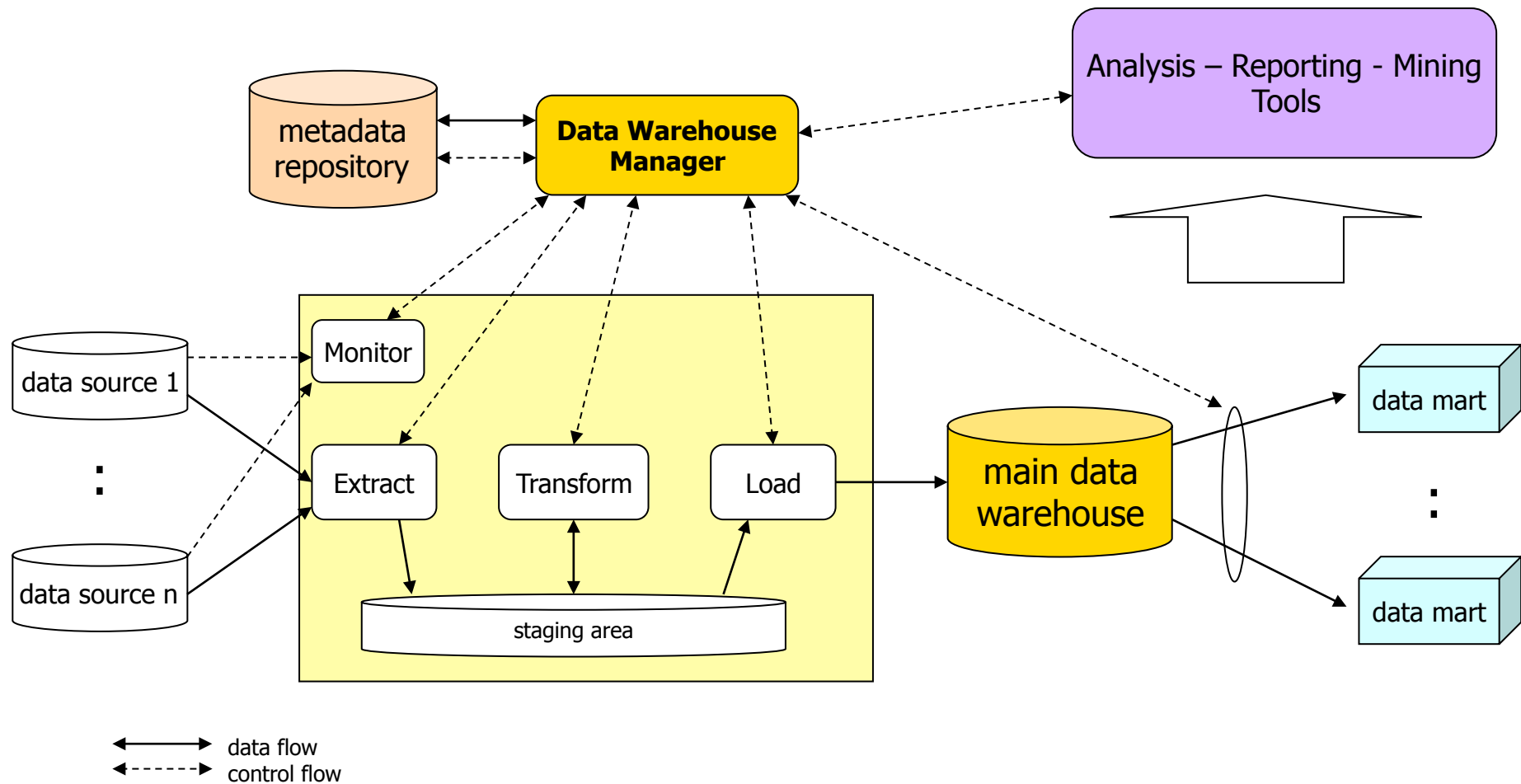
Online Analytic Processing (OLAP)

- Goal: analyze all (or large amounts of) company data to discover patterns or trends of importance
 - explore numerical measures along categories
- Queries (OLAP queries, decision support queries) are very complex
 - joins, aggregations, ...
- Often based on an integrated data warehouse
 - organize and centralize corporate data from various sources
 - can involve additional integration work, data cleansing and normalization
 - provide an execution platform for complex, potentially long-running OLAP queries without slowing down regular transaction processing systems
 - periodically refreshed from the various data sources
- In contrast to On-Line Transaction Processing (OLTP) applications

OLTP vs. OLAP

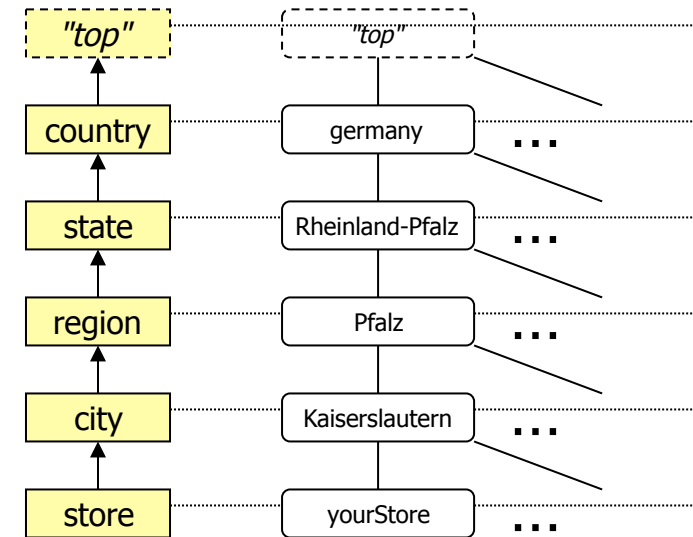
OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
many simple queries	few (repeated) complex queries
small amounts of data accessed	large amounts of data accessed
usually operate on the current data	operate on current and historical data
optimize for fast update, high throughput	optimize for fast calculation
→ hard (sometimes impossible) for DBMS to optimize for both OLTP and OLAP at the same time	
Parallel execution of OLAP and OLTP queries on operational database may impact OLTP performance	

Data Warehousing Architecture

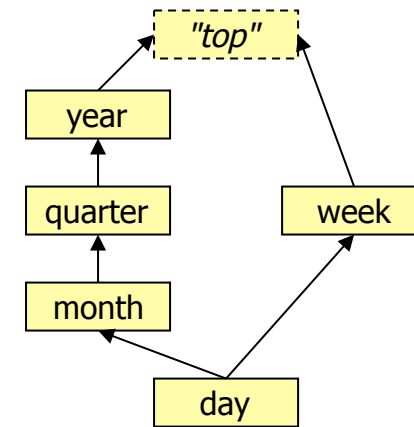


Multidimensional Data Modelling – Motivation

- Types of information for explorative data analysis
 - exploration paths ("drill-paths")
 - provided based on category attributes, concept hierarchies
 - e.g., sales events have attributes describing geography, product, and time period dimensions
 - hierarchical nature
 - geography: country, state, region, city, store
 - product: category, family, group, article
 - time: year, quarter, month, day
 - **dimensions**, describing a "navigation basis" for analysis steps
 - "qualifying" nature of category attributes
 - numerical fact data
 - the subject(s) of the analysis process
 - **measures**, summary attributes about events or objects of interest
 - e.g., the sales amount for a sales event
 - "quantifying" nature of summary attributes



simple hierarchy



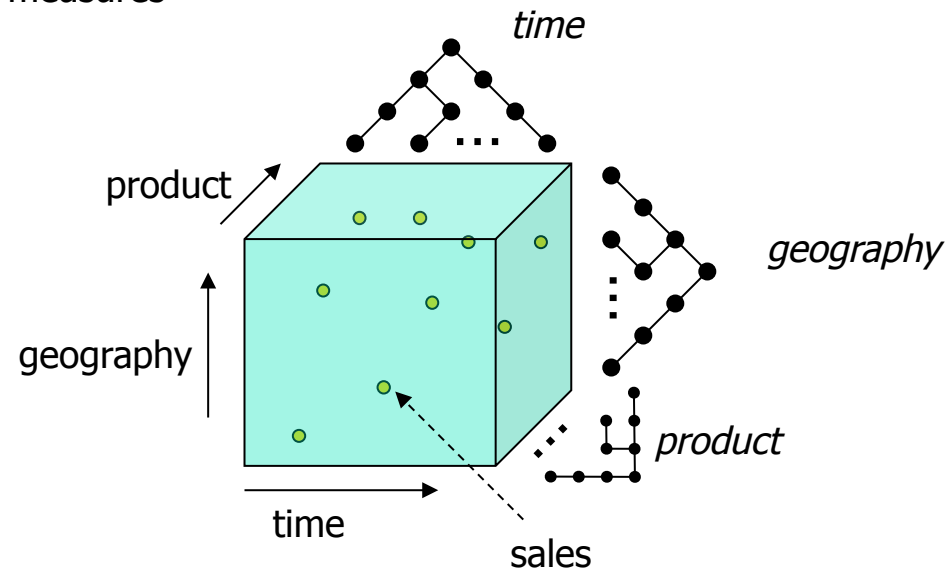
parallel hierarchies

Multidimensional Data Modelling – Motivation (cont.)

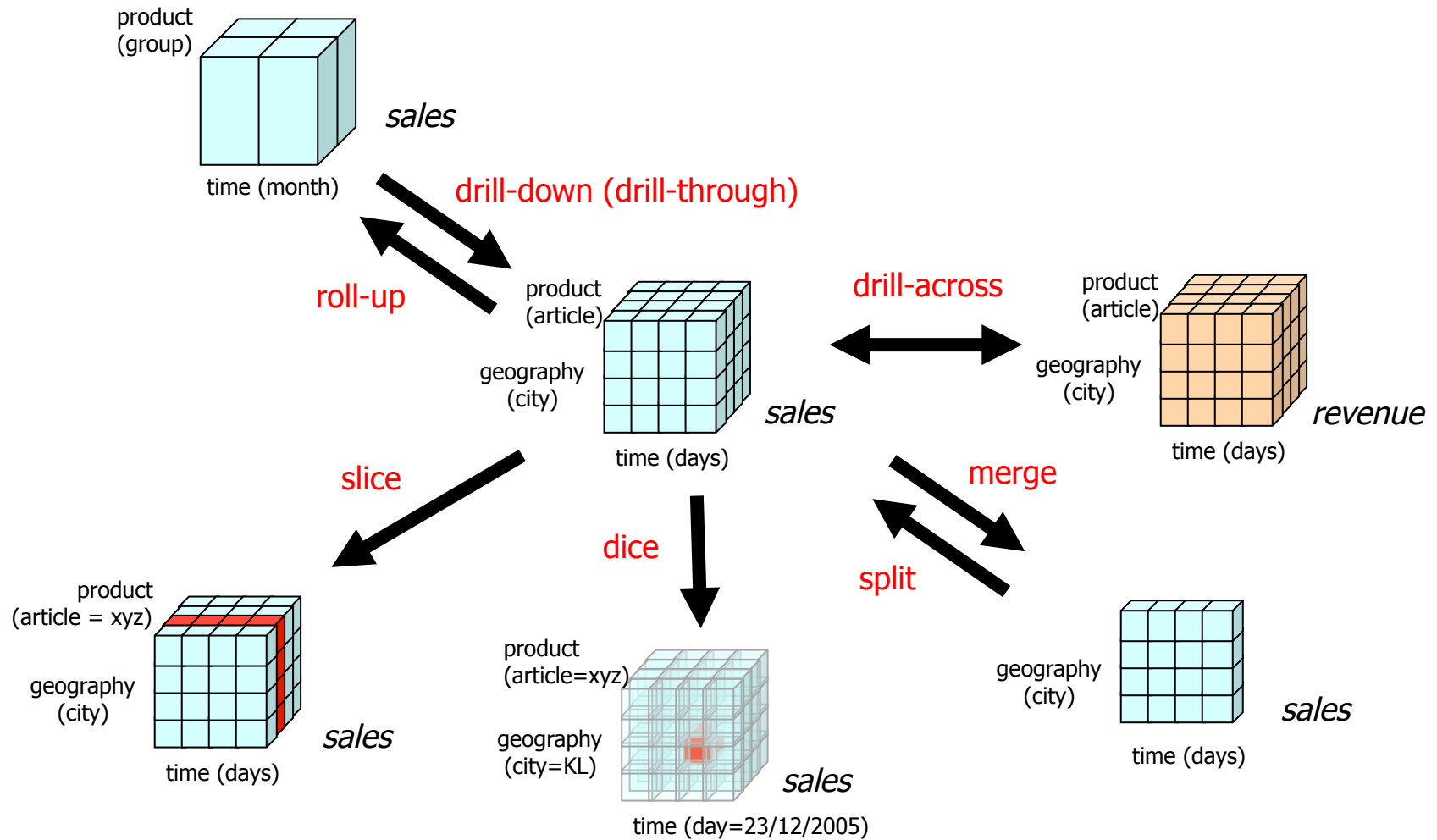
- Analysis operations
 - use the dimensions to qualify and group numerical fact data
 - aggregate the measures on a per-group basis
 - e.g., for 2005, determine the sum of all sales amounts per sales region and product category
 - sequences of operations to "refine" the analysis based on previous steps
 - involves additional qualification predicates, grouping based on different level in a dimension hierarchy
- Multi-dimensional data models have been developed to explicitly model and capture the above aspects
- Multi-dimensional OLAP (MOLAP) servers
 - directly support multi-dimensional data models through special storage engine
 - provide superior indexing capabilities
 - usually lack ad-hoc query support
- Relational OLAP (ROLAP)
 - leverages/extends RDBMS capabilities
 - has triggered various efforts to enhance SQL query processing capabilities for OLAP

Multidimensional Data Cube

- Information can often be thought of as arranged in a multi-dimensional space, or (hyper-) **cube**
 - dimensions define the axes
 - measures define the data points in cube "cells"
 - base measure (facts)
 - derived measures



Data Cube Operations



Concepts of Multi-dimensional Data Models

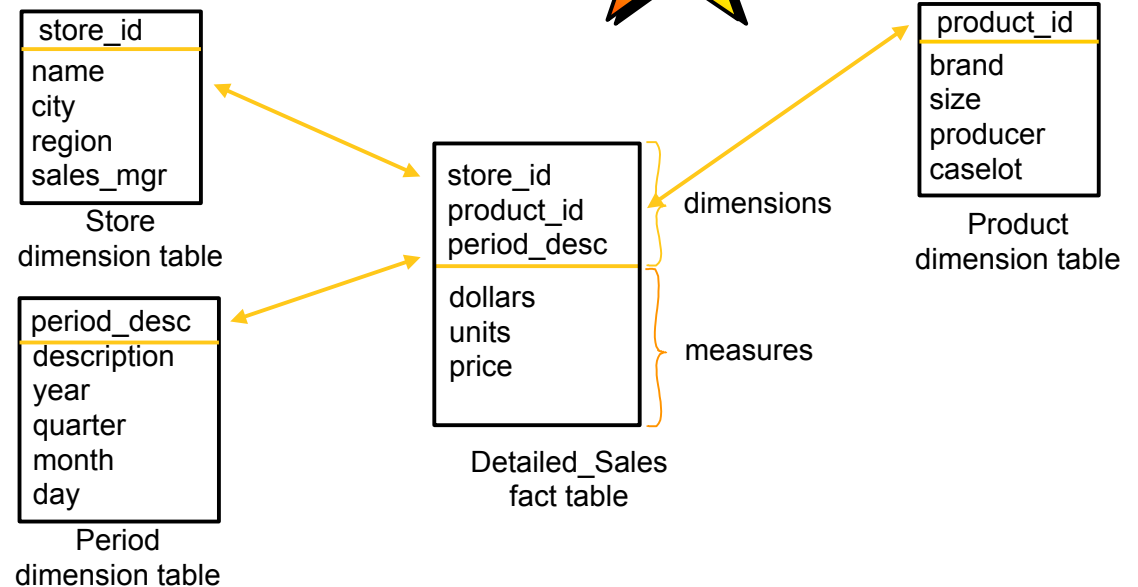
- Multi-dimensional schema consists of
 - a set of dimensions
 - a set of measures
- Dimension
 - partially ordered set of category attributes
 - primary attribute
 - characterizes the finest analysis granularity for a dimension (e.g., store)
 - classification attributes
 - realize a classification hierarchy for the dimension (e.g., city, region, state, country)
 - dimensional attributes
 - provide additional information at specific levels of the dimension (e.g., address of store)
- Measures (of a cube)
 - basic measures (facts), derived measures
 - derived measures include scalar expression defining the derivation
 - granularity
 - subset of category attributes, functionally independent
 - aggregation function

(Relational) OLAP Schema

- Typically uses a **Star** structure
 - Dimension tables (linked to fact table) tend to be small
 - Fact table tends to be huge
 - Measures (dependent attributes)



- Snowflake** schema
 - "normalized" dimensions
 - multiple tables to avoid redundancy
 - requires additional joins for OLAP queries

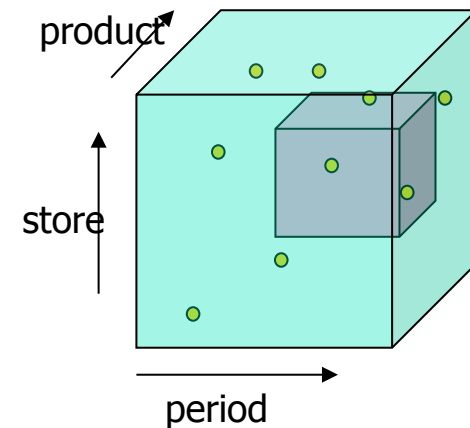


```
CREATE VIEW Sales AS
(SELECT ds.*
```

```
FROM (Detailed_Sales NATURAL JOIN Store NATURAL JOIN Product NATURAL JOIN Period) ds
```

Slicing and Dicing


- The set of points in the cube are partitioned along each dimension at some level of granularity
 - brand, size, producer for product
 - name, city, region for store
 - month, week, year for time period
- General form of statement
 - SELECT** *grouping attributes and aggregates*
 - FROM** *fact table joined with zero or more dimension tables*
 - WHERE** *classification attributes are constant or restricted*
 - GROUP BY** *grouping attributes*
- Focusing on particular partitions (through WHERE clause)
 - **"slice"**: fix one of the dimensions to a constant value
 - **"dice"**: provide (interval) restrictions on one or more dimension
- Choice of partitioning (GROUP BY)
 - divides slice/dice or complete cube into smaller cubes that contain points whose measure are aggregated



Drill-down, Roll-Up and Data Cubes

- Drill-down, Roll-up: common patterns in sequences of queries that slice & dice
 - Drill-down gradually partitions more finely along a hierarchy of a dimension or adds dimensions for grouping
 - Example:

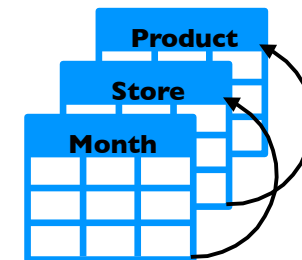
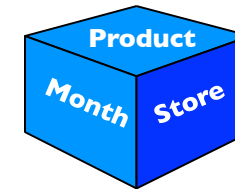
```
SELECT month, region, producer, SUM(units) AS sum_units
FROM Sales
WHERE year = 1998
GROUP BY month, region, producer
```



```
SELECT month, city, producer, SUM(units) AS sum_units
FROM Sales
WHERE year = 1998
GROUP BY month, city, producer
```
 - Roll-up gradually partitions more coarsely
 - Data Cube operator performs systematic (pre-)aggregation along all dimensions

SQL99 OLAP SQL Extensions

- Extension to GROUP BY clause
 - ROLLUP
 - CUBE
 - GROUPING SETS equivalent to multiple GROUP BYs
- Provides "data cube" collection capability
 - Often used with data visualization tool



GROUPING SETS

- Multiple "groupings" in a single pass
 - Used in conjunction with usual aggregation (MAX, MIN, SUM, AVG, COUNT, ...)
 - Allows multiple groups e.g. (month, region) and (month, sales_mgr)
 - Result can be further restricted via HAVING clause
- Example:
Find the total sales during each month of 1996, per region and per sales manager:

```
SELECT month, region, sales_mgr, SUM(price)
FROM Sales
WHERE year = 1996
GROUP BY GROUPING SETS ((month, region), (month, sales_mgr))
```
- Equivalent to:

```
(SELECT month, region, NULL, SUM(price)
FROM Sales WHERE year = 1996
GROUP BY (month, region))
UNION
(SELECT month, NULL, sales_mgr, SUM(price)
FROM Sales WHERE year = 1996
GROUP BY (month, sales_mgr))
```

Grand Total Rows

- Special syntax available to include a "grand total" row in the result
 - Syntax allows grand totals to be generated without additional aggregates
- Get total sales as in the previous query, and also the overall total sales:

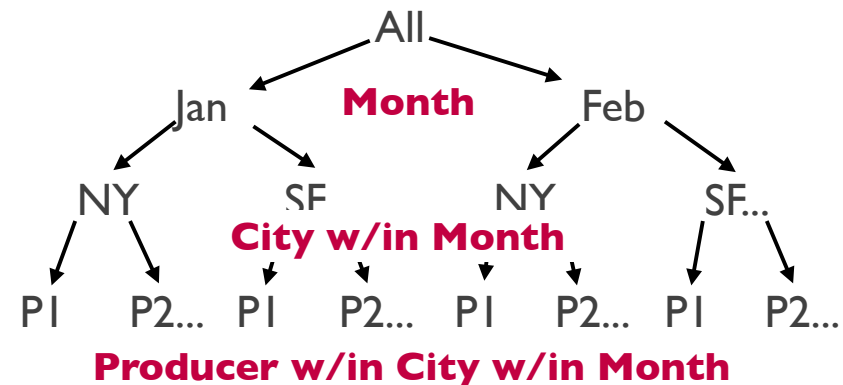
```
SELECT month, region, sales_mgr,  
       SUM (price) AS sum  
FROM Sales  
WHERE year = 1996  
GROUP BY GROUPING SETS (  
  (month, region),  
  (month, sales_mgr),  
  ( ))
```

MONTH	REGION	SALES_MGR	SUM
April	Central	-	40 000
April	NorthWest	-	15 000
April	-	Chow	25 000
April	-	Smith	30 000
May	Central	-	25 000
May	NorthWest	-	15 000
May	-	Chow	25 000
May	-	Smith	15 000
-	-	-	90 000

ROLLUP

- Extends grouping/aggregation semantics to combination of groups
 - Produces "regular" groups/aggregated rows
 - Repeatedly combines/aggregates groups, "dropping" grouping attributes at the end of the list, up to grand total
- Equivalences:
GROUP BY ROLLUP (month, city, producer) ... equivalent to
GROUP BY GROUPING SETS ((month, city, producer), (month, city), (month), ())

```
SELECT month, city, producer, SUM(units) AS sum_units  
FROM Sales  
WHERE year = 1998  
GROUP BY ROLLUP (month, city, producer)
```



ROLLUP

- Find the total sales per region and sales manager during each month of 1996, with subtotals for each month, and concluding with the grand total:

```
SELECT month, region, sales_mgr, SUM (price)
FROM Sales
WHERE year = 1996
GROUP BY ROLLUP
        (month, region, sales_mgr)
```

MONTH	REGION	SALES_MGR	SUM(price)
April	Central	Chow	25 000
April	Central	Smith	15 000
April	Central	-	40 000
April	NorthWest	Smith	15 000
April	NorthWest	-	15 000
April	-	-	55 000
May	Central	Chow	25 000
May	Central	-	25 000
May	NorthWest	Smith	15 000
May	NorthWest	-	15 000
May	-	-	40 000
-	-	-	95 000

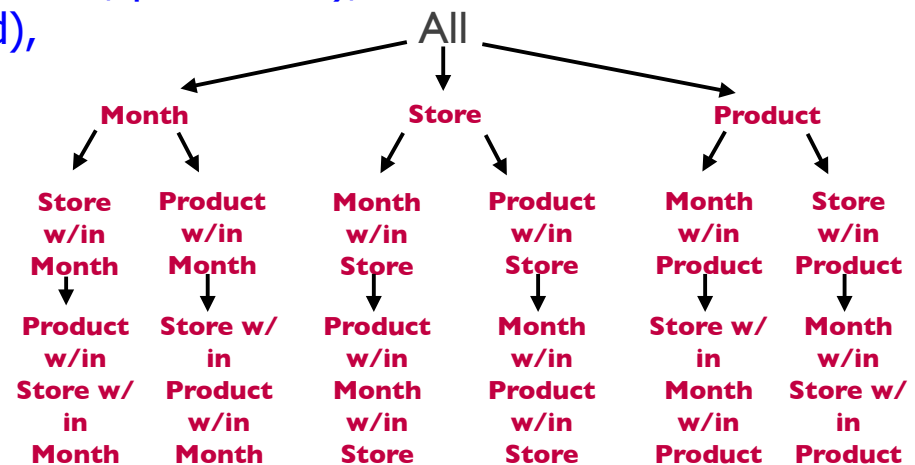
CUBE

- Further extends grouping semantics to produce multidimensional grouping and "subtotal" rows
 - Superset of ROLLUP
 - Produces "regular" grouped rows
 - Produces same groupings reapplied up to grand total
 - Produces additional groupings on all variants of the CUBE clause

- Equivalences:

GROUP BY CUBE (month, store, product_id) ... equivalent to
 GROUP BY GROUPING SETS ((month, store, product_id),
 (month, store), (month, product_id),
 (region, product_id), (month),
 (store), (product_id), ())

```
SELECT month, city, product_id, SUM(units)
FROM Sales
WHERE year = 1998
GROUP BY CUBE (month, store, product_id)
```



... GROUP BY CUBE

```
SELECT month, region, sales_mgr, SUM(price)
FROM Sales
WHERE year = 1996
GROUP BY CUBE (month, region, sales_mgr)
```

MONTH	REGION	SALES_MGR	SUM(price)
April	Central	Chow	25 000
April	Central	Smith	15 000
April	Central	-	40 000
April	NorthWest	Smith	15 000
April	NorthWest	-	15 000
April	-	Chow	25 000
April	-	Smith	30 000
April	-	-	55 000
May	Central	Chow	25 000
May	Central	-	25 000
May	NorthWest	Smith	15 000
May	NorthWest	-	15 000
May	-	Chow	25 000
May	-	Smith	15 000
May	-	-	40 000
-	Central	Chow	50 000
-	Central	Smith	15 000
-	Central	-	65 000
-	NorthWest	Smith	30 000
-	NorthWest	-	30 000
-	-	Chow	50 000
-	-	Smith	45 000
-	-	-	95 000



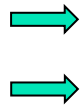
The GROUPING Function

- New column function
 - Allows detection of rows that were generated during the execution of grouping sets (incl. CUBE and ROLLUP), i.e. generated nulls to be distinguished from naturally occurring ones
- Example:

Run a rollup, and flag the rows with generated NULL values for "sales_mgr" to distinguish sales with unknown sales_mgr from month/region aggregates

```
SELECT month, region, sales_mgr, SUM(price), GROUPING(sales_mgr) AS  
    grouping  
FROM Sales  
WHERE year = 1996  
GROUP BY ROLLUP (month, region, sales_mgr)
```

Result...



MONTH	REGION	SALES_MGR	SUM(price)	GROUPING
April	Central	Chow	25 000	0
April	Central	Smith	15 000	0
April	Central	-	10 000	0
April	Central	-	50 000	1
April	NorthWest	Smith	15 000	0
April	NorthWest	-	15 000	1
April	-	-	65 000	1
May	Central	Chow	25 000	0
May	Central	-	25 000	1
May	NorthWest	Smith	15 000	0
May	NorthWest	-	15 000	1
May	-	-	40 000	1
-	-	-	105 000	1



Selecting Nongrouped Columns

- Nongrouped columns can sometimes be selected based on functional dependencies:

```
SELECT e.deptno, d.location, AVG (e.salary) AS average  
FROM Emp e , Dept d  
WHERE e.deptno = d.deptno  
GROUP BY e.deptno
```



e.deptno determines d.deptno (equals in WHERE clause), and d.deptno determines d.location (deptno is PK of Dept); therefore, d.deptno and d.location are consistent within any group. This is functional dependency analysis in action.



```
SELECT e.deptno, e.name, AVG (e.salary) AS Average  
FROM Emp e, Dept d  
WHERE e.deptno =d.deptno  
GROUP BY e.deptno
```

Summary

- Online Analytic Processing (OLAP)
 - analyze large amounts of corporate data
 - complex, long-running queries
 - usually supported by data warehousing architecture, data marts
- Multi-dimensional data models
 - dimensions for qualifying sets of data, exploring/navigating the data space
 - numerical measures as the subject of analysis, aggregation
 - multi-dimensional data cube as a conceptual paradigm
 - operators for roll-up, drill-down, slice, dice, ...
- Relational OLAP (ROLAP)
 - star, snowflake schema for representing multi-dimensional data
- SQL enhancements for OLAP
 - extensions of GROUP BY support for performing extensive aggregations suitable for computing multi-dimensional data cube with a single query
 - large potential for optimization by DBMS query engine
 - can be utilized by OLAP tools, middleware