# Chapter 11: Data Analytics

Database System Concepts, 7th Ed.

# Chapter 11: Data Analytics

- Overview
- Data Warehousing
- Online Analytical Processing
- ~~Data Mining~~

# Overview

- **Data analytics**: the processing of data to infer patterns, correlations, or models for prediction
- Primarily used to make business decisions
  - Per individual customer
    - E.g., what product to suggest for purchase
  - Across all customers
    - E.g., what products to manufacture/stock, in what quantity
- Critical for businesses today

# Overview (Cont.)

- Common steps in data analytics
  - Gather data from multiple sources into one location
    - Data warehouses also integrated data into common schema
    - Data often needs to be **extracted** from source formats, **transformed** to common schema, and **loaded** into the data warehouse
      - Can be done as **ETL (extract-transform-load)**, or **ELT (extract-load-transform)**
  - Generate aggregates and reports summarizing data
    - Dashboards showing graphical charts/reports
    - **Online analytical processing (OLAP) systems** allow interactive querying
    - Statistical analysis using tools such as R/SAS/SPSS
      - Including extensions for parallel processing of big data
  - Build **predictive models** and use the models for decision making

# Overview (Cont.)

- Predictive models are widely used today
  - E.g., use customer profile features (e.g. income, age, gender, education, employment) and past history of a customer to predict likelihood of default on loan
    - and use prediction to make loan decision
  - E.g., use past history of sales (by season) to predict future sales
    - And use it to decide what/how much to produce/stock
    - And to target customers
- Other examples of business decisions:
  - What items to stock?
  - What insurance premium to change?
  - To whom to send advertisements?

# Overview (Cont.)

- **Machine learning** techniques are key to finding patterns in data and making predictions
- **Data mining** extends techniques developed by machine-learning communities to run them on very large datasets
- The term **business intelligence (BI)** is synonym for data analytics
- The term **decision support** focuses on reporting and aggregation

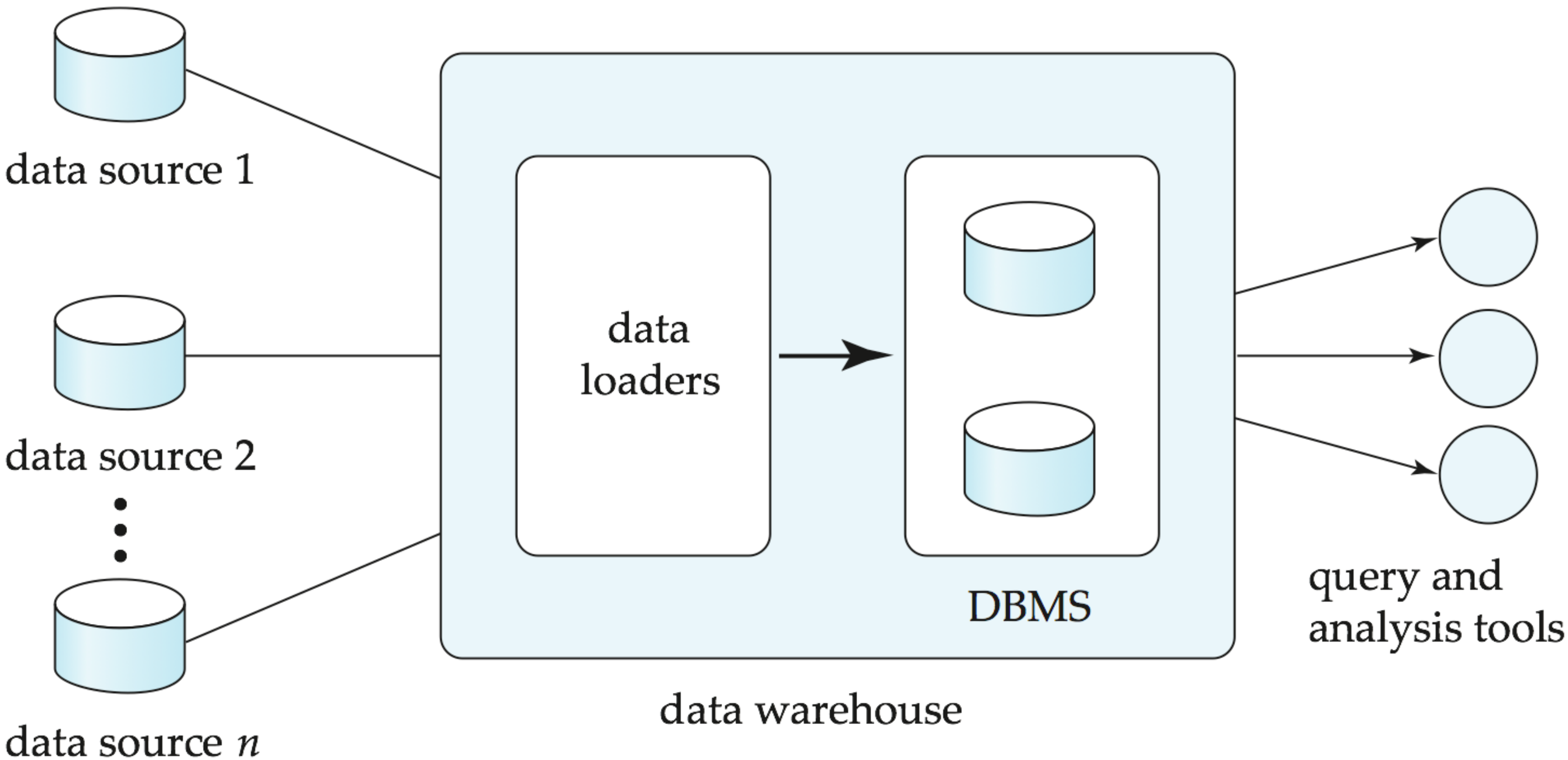
# DATA WAREHOUSING

# Data Warehousing

- Data sources often store only current data, not historical data
- Corporate decision making requires a unified view of all organizational data, including historical data
- A **data warehouse** is a repository (archive) of information gathered from multiple sources, stored under a unified schema, at a single site
  - Greatly simplifies querying, permits study of historical trends
  - Shifts decision support query load away from transaction processing systems

# Data Warehousing



# Design Issues

- *When and how to gather data*
  - **Source driven architecture**: data sources transmit new information to warehouse
    - either continuously or periodically (e.g., at night)
  - **Destination driven architecture:** warehouse periodically requests new information from data sources
  - **Synchronous** vs **asynchronous replication**
    - Keeping warehouse exactly synchronized with data sources (e.g., using two-phase commit) is often too expensive
    - Usually OK to have slightly out-of-date data at warehouse
    - Data/updates are periodically downloaded form online transaction processing (OLTP) systems.
- *What schema to use*
  - Schema integration

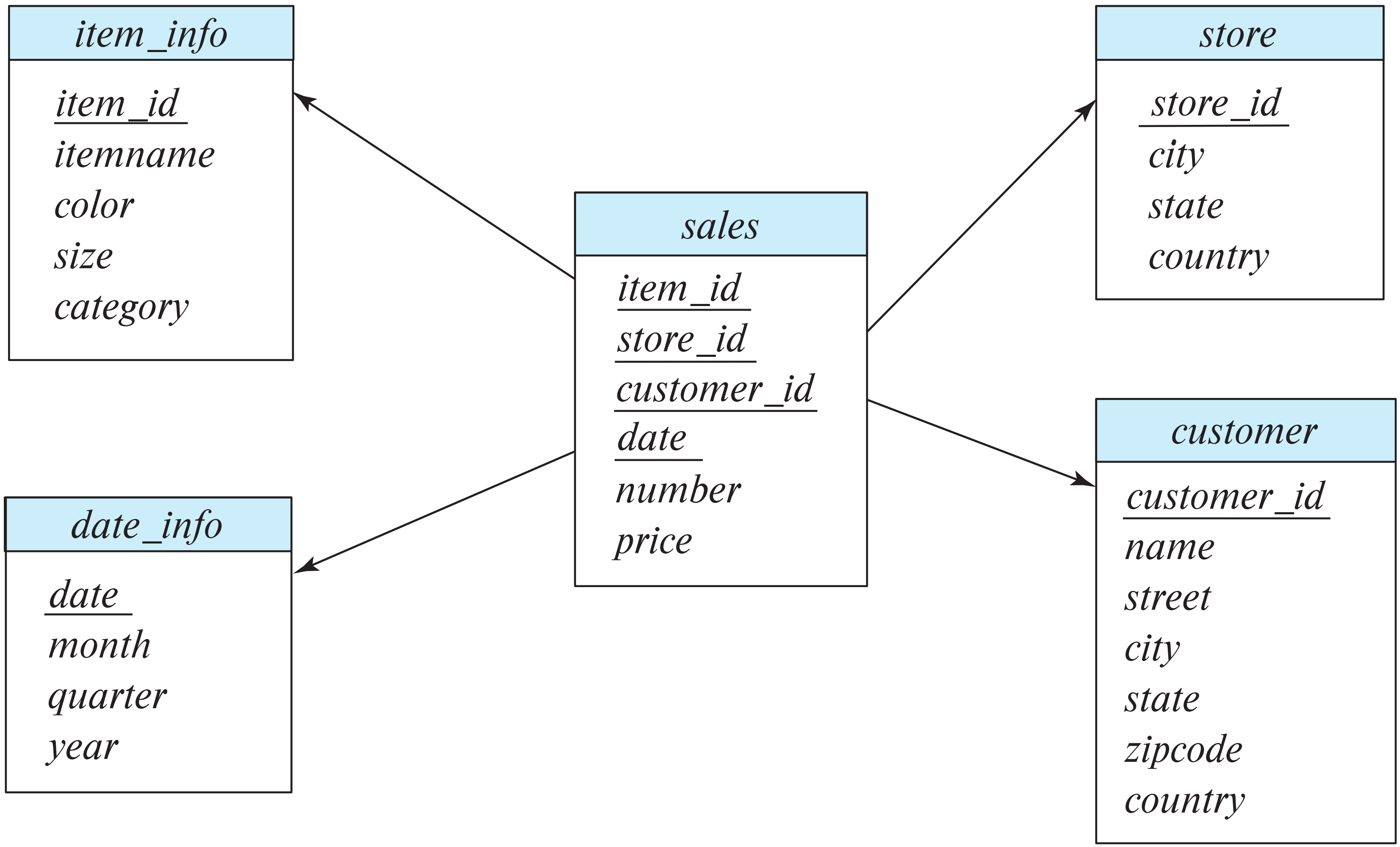
# More Warehouse Design Issues

- **Data transformation** and **data cleansing**
  - E.g., correct mistakes in addresses (misspellings, zip code errors)
  - Merge address lists from different sources and purge duplicates
- *How to propagate updates*
  - Warehouse schema may be a (materialized) view of schema from data sources
    - View maintenance
- *What data to summarize*
  - Raw data may be too large to store on-line
  - Aggregate values (totals/subtotals) often suffice
  - Queries on raw data can often be transformed by query optimizer to use aggregate values

# Multidimensional Data and Warehouse Schemas

- Data in warehouses can usually be divided into
  - **Fact tables**, which are large
    - E.g, *sales*(*item_id, store_id, customer_id, date, number, price*)
  - **Dimension tables**, which are relatively small
    - Store extra information about stores, items, etc.
- Attributes of fact tables can be usually viewed as
  - **Measure attributes**
    - measure some value, and can be aggregated upon
    - e.g., the attributes *number* or *price* of the *sales* relation
  - **Dimension attributes**
    - dimensions on which measure attributes are viewed
    - e.g., attributes *item_id, color,* and *size* of the *sales* relation
    - Usually small ids that are foreign keys to dimension tables

# Data Warehouse Schema



# Multidimensional Data and Warehouse Schemas

- Resultant schema is called a **star schema**
  - More complicated schema structures
    - **Snowflake schema**: multiple levels of dimension tables
    - May have multiple fact tables
- Typically
  - fact table joined with dimension tables and then
  - group-by on dimension table attributes, and then
  - aggregation on measure attributes of fact table
- Some applications do not find it worthwhile to bring data to a common schema
  - **Data lakes** are repositories which allow data to be stored in multiple formats, without schema integration
  - Less upfront effort, but more effort during querying

# Database Support for Data Warehouses

- Data in warehouses usually append only, not updated
  - Can avoid concurrency control overheads
- Data warehouses often use **column-oriented storage**
  - E.g., a sequence of *sales* tuples is stored as follows
    - Values of item_id attribute are stored as an array
    - Values of store_id attribute are stored as an array,
    - And so on
  - Arrays are compressed, reducing storage, IO and memory costs significantly
  - Queries can fetch only attributes that they care about, reducing IO and memory cost
  - More details in Section 13.6
- Data warehouses often use parallel storage and query processing infrastructure
  - Distributed file systems, Map-Reduce, Hive, …

# OLAP

# Data Analysis and OLAP

- **Online Analytical Processing (OLAP)**
  - Interactive analysis of data, allowing data to be summarized and viewed in different ways in an online fashion (with negligible delay)
- We use the following relation to illustrate OLAP concepts
  - *sales* (*item_name*, *color*, *clothes_size*, *quantity*)

  This is a simplified version of the *sales* fact table joined with the dimension tables, and many attributes removed (and some renamed)

# Example sales relation

| *item_name* | *color* | *clothes_size* | *quantity* |
|---|---|---|---|
| dress | dark | small | 2 |
| dress | dark | medium | 6 |
| dress | dark | large | 12 |
| dress | pastel | small | 4 |
| dress | pastel | medium | 3 |
| dress | pastel | large | 3 |
| dress | white | small | 2 |
| dress | white | medium | 3 |
| dress | white | large | 0 |
| pants | dark | small | 14 |
| pants | dark | medium | 6 |
| pants | dark | large | 0 |
| pants | pastel | small | 1 |
| pants | pastel | medium | 0 |
| pants | pastel | large | 1 |
| pants | white | small | 3 |
| pants | white | medium | 0 |
| pants | white | large | 2 |
| shirt | dark | small | 2 |
| shirt | dark | medium | 6 |
| shirt | dark | large | 6 |
| shirt | pastel | small | 4 |
| shirt | pastel | medium | 1 |
| shirt | pastel | large | 2 |
| shirt | white | small | 17 |
| shirt | white | medium | 1 |
| shirt | white | large | 10 |
| skirt | dark | small | 2 |
| skirt | dark | medium | 5 |
| … | … | … | … |
| … | … | … | … |

# Cross Tabulation of *sales* by *item_name* and *color*

*clothes_size* **all**

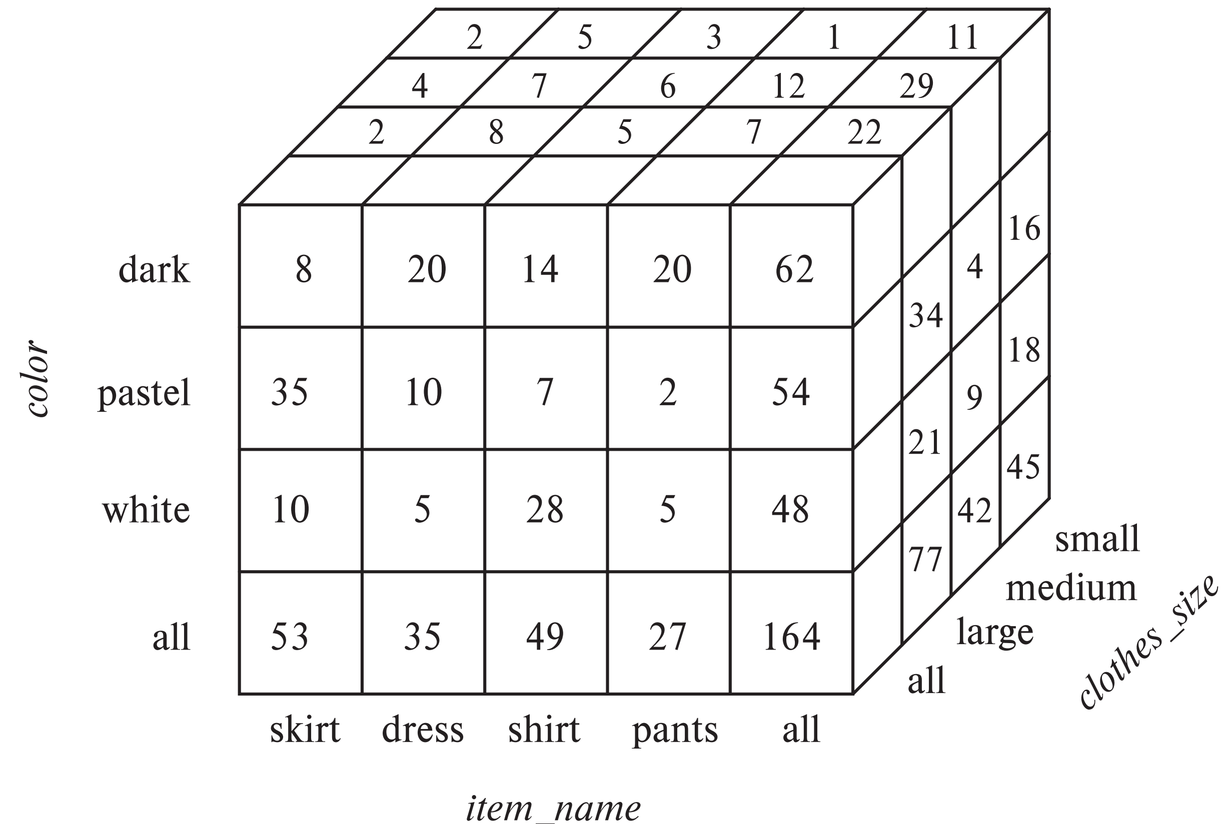
*color*

| *item_name* | dark | pastel | white | total |
|---|---|---|---|---|
| skirt | 8 | 35 | 10 | 53 |
| dress | 20 | 10 | 5 | 35 |
| shirt | 14 | 7 | 28 | 49 |
| pants | 20 | 2 | 5 | 27 |
| total | 62 | 54 | 48 | 164 |

- The table above is an example of a **cross-tabulation** (**cross-tab**), also referred to as a **pivot-table**.
  - Values for one of the dimension attributes form the row headers
  - Values for another dimension attribute form the column headers
  - Other dimension attributes are listed on top
  - Values in individual cells are (aggregates of) the values of the dimension attributes that specify the cell.

# Data Cube

- A **data cube** is a multidimensional generalization of a cross-tab
- Can have n  dimensions; we show 3 below
- Cross-tabs can be used as views on a data cube

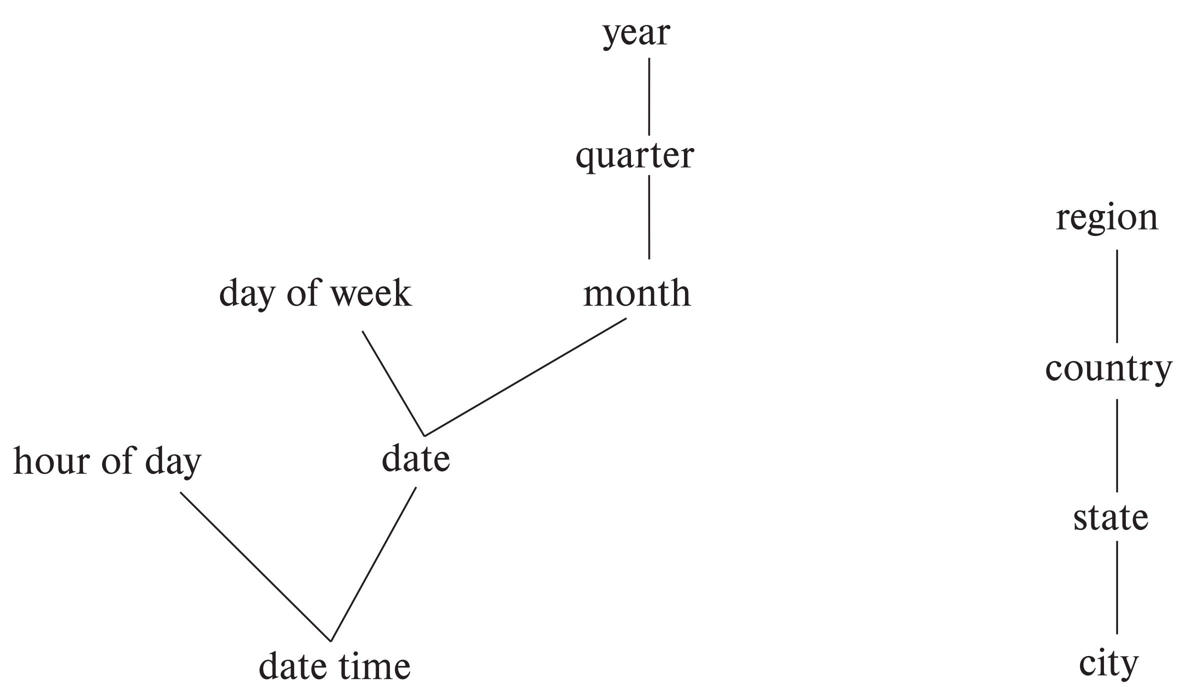


# Online Analytical Processing Operations

- **Pivoting:** changing the dimensions used in a cross-tab
  - E.g., moving colors to column names
- **Slicing:** creating a cross-tab for fixed values only
  - E.g., fixing color to white and size to small
  - Sometimes called **dicing**, particularly when values for multiple dimensions are fixed.
- **Rollup:** moving from finer-granularity data to a coarser granularity
  - E.g., aggregating away an attribute
  - E.g., moving from aggregates by day to aggregates by month or year
- **Drill down:** The opposite operation - that of moving from coarser-granularity data to finer-granularity data

# Hierarchies on Dimensions

- **Hierarchy** on dimension attributes: lets dimensions be viewed at different levels of detail
- E.g., the dimension *datetime* can be used to aggregate by hour of day, date, day of week, month, quarter or year




(a) time hierarchy

(b) location hierarchy

# Cross Tabulation With Hierarchy

- Cross-tabs can be easily extended to deal with hierarchies
- Can drill down or roll up on a hierarchy
- E.g. hierarchy: *item_name* → *category*

*clothes_size:* **all**

| *category* | *item_name* | *color*: dark | pastel | white | total | |
|---|---|---|---|---|---|---|
| womenswear | skirt | 8 | 8 | 10 | 53 | |
| | dress | 20 | 20 | 5 | 35 | |
| | subtotal | 28 | 28 | 15 | | 88 |
| menswear | pants | 14 | 14 | 28 | 49 | |
| | shirt | 20 | 20 | 5 | 27 | |
| | subtotal | 34 | 34 | 33 | | 76 |
| total | | 62 | 62 | 48 | | 164 |

# Relational Representation of Cross-tabs

- Cross-tabs can be represented as relations
- We use the value **all** to represent aggregates.
- The SQL standard actually uses *null* values in place of **all**
  - Works with any data type
  - But can cause confusion with regular null values.

| *item_name* | *color* | *clothes_size* | *quantity* |
|---|---|---|---|
| skirt | dark | **all** | 8 |
| skirt | pastel | **all** | 35 |
| skirt | white | **all** | 10 |
| skirt | **all** | **all** | 53 |
| dress | dark | **all** | 20 |
| dress | pastel | **all** | 10 |
| dress | white | **all** | 5 |
| dress | **all** | **all** | 35 |
| shirt | dark | **all** | 14 |
| shirt | pastel | **all** | 7 |
| shirt | white | **all** | 28 |
| shirt | **all** | **all** | 49 |
| pants | dark | **all** | 20 |
| pants | pastel | **all** | 2 |
| pants | white | **all** | 5 |
| pants | **all** | **all** | 27 |
| **all** | dark | **all** | 62 |
| **all** | pastel | **all** | 54 |
| **all** | white | **all** | 48 |
| **all** | **all** | **all** | 164 |

# OLAP IN SQL

# Pivot Operation

- **select** *
  **from** *sales*
  **pivot** (
      **sum**(*quantity*)
      **for** *color* **in** ('dark','pastel','white')
  )
  **order by** *item name*;

| *item_name* | *clothes_size* | *dark* | *pastel* | *white* |
|---|---|---|---|---|
| dress | small | 2 | 4 | 2 |
| dress | medium | 6 | 3 | 3 |
| dress | large | 12 | 3 | 0 |
| pants | small | 14 | 1 | 3 |
| pants | medium | 6 | 0 | 0 |
| pants | large | 0 | 1 | 2 |
| shirt | small | 2 | 4 | 17 |
| shirt | medium | 6 | 1 | 1 |
| shirt | large | 6 | 2 | 10 |
| skirt | small | 2 | 11 | 2 |
| skirt | medium | 5 | 9 | 5 |
| skirt | large | 1 | 15 | 3 |

# Cube Operation

- The **cube** operation computes union of **group by**'s on every subset of the specified attributes
- E.g., consider the query

  **select** *item_name, color, size,* **sum**(*number*)
  **from** *sales*
  **group by cube**(*item_name, color, size*)

  This computes the union of eight different groupings of the *sales* relation:

  { (*item_name, color, size*), (*item_name, color*),
  (*item_name, size*), (*color, size*),
  (*item_name*), (*color*),
  (*size*), ( ) }

  where ( ) denotes an empty **group by** list.
- For each grouping, the result contains the null value for attributes not present in the grouping.

# Online Analytical Processing Operations

- Relational representation of cross-tab that we saw earlier, but with *null* in place of **all**, can be computed by

  **select** *item_name*, *color*, **sum**(*number*)
  **from** *sales*
  **group by cube**(*item_name, color*)

- The function **grouping()** can be applied on an attribute
  - Returns 1 if the value is a null value representing all, and returns 0 in all other cases.

**select case when grouping**(*item_name*) = 1 **then 'all'**
          **else** *item_name* **end as** *item_name*,
     **case when grouping**(*color*) = 1 **then 'all'**
          **else** *color* **end as** *color*,
     **'all' as** *clothes size*, **sum**(*quantity*) **as** *quantity*
**from** *sales*
**group by cube**(*item name*, *color*);

# Online Analytical Processing Operations

- Can use the function **decode()** in the **select** clause to replace such nulls by a value such as **all**
  - E.g., replace *item_name* in first query by
    **decode**( **grouping**(item_*name*), 1, 'all' , *item_name*)

# Extended Aggregation (Cont.)

- The **rollup** construct generates union on every prefix of specified list of attributes
- **select** *item_name*, *color*, *size*, **sum**(*number*)
  **from** *sales*
  **group by rollup**(*item_name, color, size*)

  Generates union of four groupings:

  { (*item_name, color, size*), (*item_name, color*), (*item_name*), ( ) }
- Rollup can be used to generate aggregates at multiple levels of a hierarchy.
- E.g., suppose table *itemcategory*(*item_name, category*) gives the category of each item. Then

  **select** *category, item_name*, **sum**(*number*)
  **from** *sales, itemcategory*
  **where** *sales.item_name* = *itemcategory.item_name*
  **group by rollup**(*category, item_name*)

  would give a hierarchical summary by *item_name* and by *category.*

# Extended Aggregation (Cont.)

- Multiple rollups and cubes can be used in a single group by clause
  - Each generates set of group by lists, cross product of sets gives overall set of group by lists
- E.g.,

  **select** *item_name, color, size*, **sum**(*number*)
  **from** *sales*
  **group by rollup**(*item_name*), **rollup**(*color, size*)

  generates the groupings

  {*item_name, ()*} *X* {*(color, size), (color), ()*}

  = { (*item_name, color, size*), (*item_name, color*), (*item_name*), (*color, size*), (*color*), ( ) }

- **select** *item_name*, *color*, *clothes_size*, **sum**(*quantity*)
  **from** *sales*
  **group by grouping sets** ((*color*, *clothes_size*), (*clothes_size*, *item_name*));

# OLAP Implementation

- The earliest OLAP systems used multidimensional arrays in memory to store data cubes, and are referred to as **multidimensional OLAP (MOLAP)** systems.
- OLAP implementations using only relational database features are called **relational OLAP (ROLAP)** systems
- Hybrid systems, which store some summaries in memory and store the base data and other summaries in a relational database, are called **hybrid OLAP (HOLAP)** systems.

# OLAP Implementation (Cont.)

- Early OLAP systems precomputed *all* possible aggregates in order to provide online response
  - Space and time requirements for doing so can be very high
    - $2^n$ combinations of **group by**
  - It suffices to precompute some aggregates, and compute others on demand from one of the precomputed aggregates
    - Can compute aggregate on (*item_name, color*) from an aggregate on (*item_name, color, size*)
      - For all but a few "non-decomposable" aggregates such as *median*
      - is cheaper than computing it from scratch
- Several optimizations available for computing multiple aggregates
  - Can compute aggregate on (*item_name, color*) from an aggregate on (*item_name, color, size*)
  - Can compute aggregates on (*item_name, color, size*), (*item_name, color*) and (*item_name*) using a single sorting of the base data

# Reporting and Visualization

- **Reporting tools** help create formatted reports with tabular/graphical representation of data
  - E.g., SQL Server reporting services, Crystal Reports
- **Data visualization** tools help create interactive visualization of data
  - E.g., Tableau, FusionChart, plotly, Datawrapper, Google Charts, etc.
  - Frontend typically based on HTML+JavaScript

**Acme Supply Company, Inc.**
**Quarterly Sales Report**

Period: Jan. 1 to March 31, 2009

| Region | Category | Sales | Subtotal |
|---|---|---|---|
| North | Computer Hardware | 1,000,000 | |
| | Computer Software | 500,000 | |
| | All categories | | 1,500,000 |
| South | Computer Hardware | 200,000 | |
| | Computer Software | 400,000 | |
| | All categories | | 600,000 |
| | | **Total Sales** | 2,100,000 |