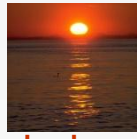
Chapter 18: Data Analysis and Mining

Database System Concepts

Chapter 18: Data Analysis and Mining

- Decision Support Systems
  - Data Analysis and OLAP
  - Data Warehousing
  - Data Mining



Decision Support Systems

- Database applications can be broadly classified into transaction processing and decision-support system.
- Transaction processing systems are systems that record information about transactions, such as product sales information for the companies, or course registration and grade information for universities. These are widely used today and organizations have accumulated a vast amount of information generated by these systems.
- Decision-support systems are used to make business decisions, often based on data collected by on-line transaction-processing systems. These aims to get high level information out of the detailed information stored in transaction processing systems. Decision support system helps managers to decide what product to stock in a shop, what products to manufacture in a factory or which of the applicants should be admitted to a university.
- Examples of business decisions:
  - What items to stock?
  - What insurance premium to change?
  - To whom to send advertisements?

Decision Support Systems (Cont.)

- Transaction information for a retailer may include:
  - Sales transaction details (the name or identifier (such as credit card number) of the customer, item purchased, price paid, date)
  - Purchased item details (name, manufacturer, model number, color, size)
  - Customer profiles (credit history, income, residence, age, gender, educational background etc.)
- The storage and retrieval of data for decision support raises several issues:
- **Data analysis** tasks are simplified by specialized tools and SQL extensions. OLAP – online analytical processing deals with tools and techniques for data analysis that can give nearly instantaneous answers to queries requesting summarized data, even though the database is extremely large.

Example tasks:

- For each region, what were the total sales in the last quarter and how do they compare with the same quarter last year.
- As above, for each product category and each customer category

Decision-Support Systems (Cont.)

- **Statistical analysis** packages (e.g., SAS and S++) can be interfaced with databases.
  - Statistical analysis is a large field, but not covered here.
- A **data warehouse** archives information gathered from multiple sources, and stores it under a unified schema, at a single site. Thus they provide the user a single uniform interface to data.
  - Important for large businesses that generate data from multiple divisions, possibly at multiple sites under different schemas.
  - Data may also be purchased externally.
- **Data mining** seeks to discover knowledge semi-automatically in the form of statistical rules and patterns from large databases. The field of data mining combines knowledge discovery techniques invented by artificial intelligence researchers and statistical analysts, with efficient implementation techniques that enable them to be used on extremely large databases.

Data Analysis and OLAP

- Although complex statistical analysis is best left to statistical packages, database should support simple, commonly used forms of data analysis. Since the data stored in the database are usually large in volume, they need to be summarized in some fashion if we are to derive information that human can use.
- **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)
- Consider the following relation *sales* with the schema:

  sales (item-name, color, size, number)

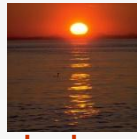
  item-name can take the values (skirt, dress, shirt, pant)

  color can take the values (dark, pastel, white)

  size can take the values (small, medium, large)
- Given a relation for data analysis, some of its attributes can be identified as **measure** attributes and some (or all) the other attributes of the relation are identified as **dimension** attributes.

OLAP

- **Measure attributes**
  - measure some value and can be aggregated upon
  - e.g. the attribute *number* of the *sales* relation (measure the number of sold item)
- **Dimension attributes**
  - define the dimensions on which measure attributes (or aggregates thereof) are viewed
  - e.g. the attributes *item_name, color,* and *size* of the *sales* relation.
- Additional dimension may be *time* and *sales location* and additional measures may be *monetary value*
- Data that can be modeled as dimension attributes and measure attributes are called **multidimensional data**.



Cross Tabulation of *sales* by *item-name* and *color*

size: **all**

item-name \ *color*	dark	pastel	white	Total
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pant	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**. This is defined as:
  - 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.
  - Most of the cross-tab also has an extra column and row showing the totals of the cell in the row/column.

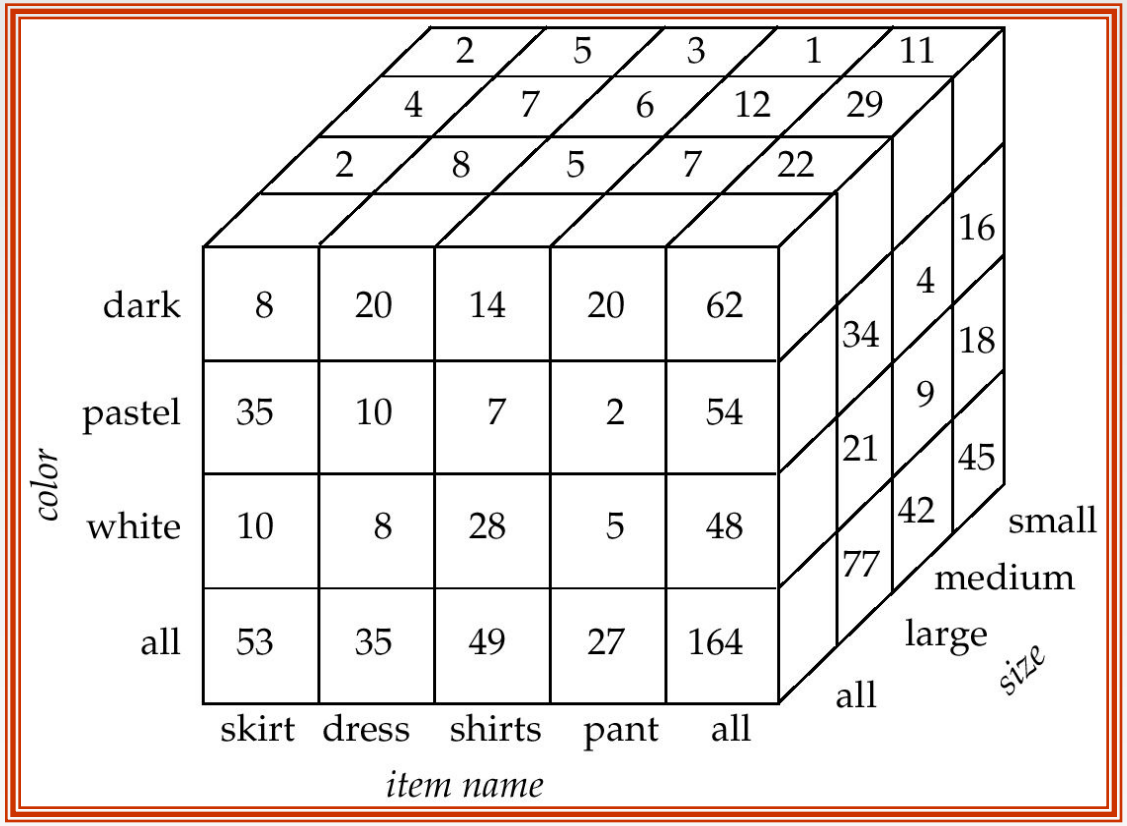
Relational Representation of Cross-tabs

- A cross-tab is different from relational tables usually stored in databases, since the number of columns in the cross-tab depends on the actual data. A change in the data value may results in adding more columns.
- Cross-tabs can be represented as relations
  - A cross-tab with summary row/columns can be represented by using a special value **all** to represent aggregates/subtotals
  - The SQL:1999 standard actually uses null values in place of **all** despite confusion with regular null values

item-name	*color*	*number*
skirt	dark	8
skirt	pastel	35
skirt	white	10
skirt	**all**	53
dress	dark	20
dress	pastel	10
dress	white	5
dress	**all**	35
shirt	dark	14
shirt	pastel	7
shirt	white	28
shirt	**all**	49
pant	dark	20
pant	pastel	2
pant	white	5
pant	**all**	27
all	dark	62
all	pastel	54
all	white	48
all	**all**	164

Data Cube

- The generalization of a cross-tab, which is two-dimensional, to n dimensions can be visualized as an n-dimensional cube, called the **data cube.** Below shows a data cube on *sales* relation with three dimensions.
- Cross-tabs can be used as views on a data cube. Each cell in the data cube contains a value, just as in a cross-tab. The value contained in a cell is shown on one of the faces of the cell; other faces of the cell are shown blank if they are visible. All cells contains values, even if they are not visible.

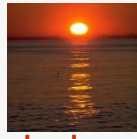


Data Cube (Cont.)

- The value for a dimension may be all, in which case the cell contains a summary over all the values of that dimension, as in the case of cross-tab.
- The number of different ways in which the tuples can be grouped for aggregation can be large. In fact, for a table with n dimensions, aggregation can be performed with grouping on each of the 2^n subsets of the n dimensions.

Online Analytical Processing

- An online analytical processing or OLAP system is an interactive system that permits an analyst to view different summaries of multidimensional data. The word *online* indicates that an analyst must be able to request new summaries and get responses online, within few seconds and should not be forced to wait for a long time to see the result of the query.
- With an OLAP system, a data analyst can look at different cross-tabs on the same data by interactively selecting the attributes in the cross-tabs. For instance, the analyst may wish to see a cross-tab on *item-name* and *size* or a cross-tab on *color* and *size* or a cross-tab on *color* and *item-name*.
- **Pivoting**: The operation of changing the dimensions used in a cross-tab is called pivoting.
- **Slicing:** creating a cross-tab for fixed values of some dimension only.
  - The analyst may wish to see a cross-tab on *item-name* and *color* for a fixed value of *size* (say large), instead of the sum across all *sizes*. It can be thought of as viewing a slice of a data cube.
  - Sometimes called **dicing**, particularly when values for multiple dimensions are fixed.

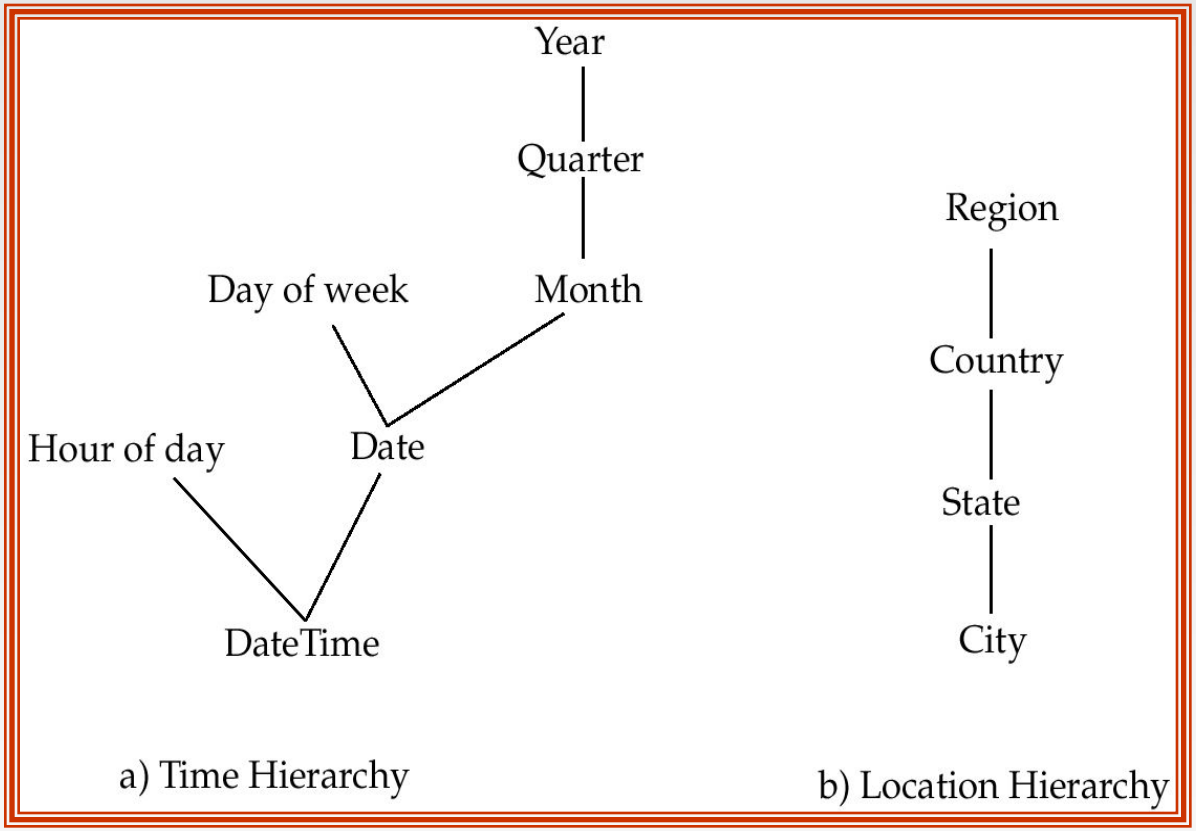


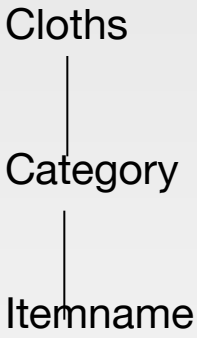
Online Analytical Processing (Cont.)

- When a cross-tab is used to view a multidimensional cube, the values of dimension attributes that are not part of the cross-tab are shown above the cross-tab. The value of such an attribute can be **all**, indicating that the data in the cross-tab are a summary over all values of the attribute. Slicing/dicing simply consists of selecting specific values for these attributes, which are then displayed on top of the cross-tab.
- **Rollup:** OLAP system permits users to view data at any desired level of granularity. The operation of moving from finer-granularity data to a coarser granularity (by means of aggregation) is called a rollup. In our example, starting from the data cube on the *sales* table, we got the example cross-tab by rolling up on the attribute *size*.
- **Drill down:** The opposite operation - that of moving from coarser-granularity data to finer-granularity data – called a drill down.
- Clearly, finer-granularity data can not be generated from coarse-granularity data; they must be generated either from the original data or from even finer-granularity summary data.

Hierarchies on Dimensions

- **Hierarchy** on dimension attributes: lets dimensions to 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





c) Clothe Hierarchy

Cross Tabulation With Hierarchy

- Cross-tabs can be easily extended to deal with hierarchies
  - Can drill down or roll up on a hierarchy

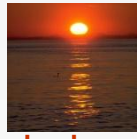
category	*item-name*					
		dark	pastel	white	total	
womenswear	skirt	8	8	10	53	
	dress	20	20	5	35	
	subtotal	28	28	15		88
menswear	pants	14	14	28	49	
	shirt	20	20	5	27	
	subtotal	34	34	33		76
total		62	62	48		164

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



Extended Aggregation in SQL:1999

- 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.

Extended Aggregation (Cont.)

- 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 *item-name, color, size*, **sum**(*number*),
  grouping(*item-name*) **as** *item-name-flag*,
  grouping(*color*) **as** *color-flag*,
  grouping(*size*) **as** *size-flag*,
  from *sales*
  group by cube(*item-name, color, size*)

- 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 the 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
- E.g.

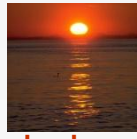
  select *item-name*, *color*, *size*, **sum**(*number*)
  from *sales*
  group by rollup(*item-name, color, size*)

  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), ()*}

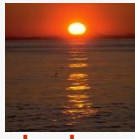
  = { (*item-name, color, size*), (*item-name, color*), (*item-name*), (*color, size*), (*color*), ( ) }

Ranking

- Finding the position of a value in a larger set is a common operation. For instance, we may wish to assign students a rank in the class based on their total marks. Such queries are difficult to express and inefficient to evaluate in SQL-92. Programmers often resort to writing the query partly in SQL and partly in a programming language. To find the percentile is also this type of query. SQL-1999 supports these type of queries.
- Ranking is done in conjunction with an order by specification.
- Given a relation student-marks(student-id, marks) find the rank of each student.

  select *student-id*, **rank**( ) **over** (**order by** *marks* **desc) as** *s-rank*
  from *student-marks*
- An extra **order by** clause is needed to get them in sorted order

  select *student-id*, **rank** ( ) **over** (**order by** *marks* **desc**) **as** *s-rank*
  from *student-marks*
  order by *s-rank*
- Ranking may leave gaps: e.g. if 2 students have the same top mark, both have rank 1, and the next rank is 3
  - **dense_rank** does not leave gaps, so next dense rank would be 2



Ranking (Cont.)

- Ranking can be done within partition of the data.
- Suppose additional relation *student-section (student-id, section)*
- "Find the rank of students within each section."

```
select student-id, section,
rank ( ) over (partition by section order by marks desc)
        as sec-rank
from student-marks, student-section
where student-marks.student-id = student-section.student-id
order by section, sec-rank
```

- Multiple **rank** clauses can occur in a single **select** clause; thus we can obtain the overall rank and rank within the section.
- When ranking occurs along with a **group by** clause, ranking is done *after* applying **group by** clause. Thus aggregate values can be used for ranking.
- Example: Students with marks in several subjects.

Ranking (Cont.)

- SQL:1999 specifies several other ranking functions:
  - **percent_rank** (within partition, if partitioning is done): gives the rank of the tuple as a fraction. If there are n tuples in the partition and the rank of the tuple is r, then its percent rank is defined as (r-1)/(n-1)
  - **cume_dist** (cumulative distribution) is defined for a tuple as *p/n*, where *p* is the number of tuples in the partition with ordering values preceding or equal to the ordering value of the tuple and **n** is the number of tuple in the partition.
  - **row_number** sorts the rows and gives each row a unique number corresponding to its position in the sort order. (non-deterministic in presence of duplicates)
- The presence of the null values can complicate the definition of rank, since it is not clear where they should occur in the sort order. SQL:1999 permits the user to specify **nulls first** or **nulls last**

select *student-id*,
rank ( ) **over** (**order by** *marks* **desc nulls last**) **as** *s-rank*
from *student-marks*

Ranking (Cont.)

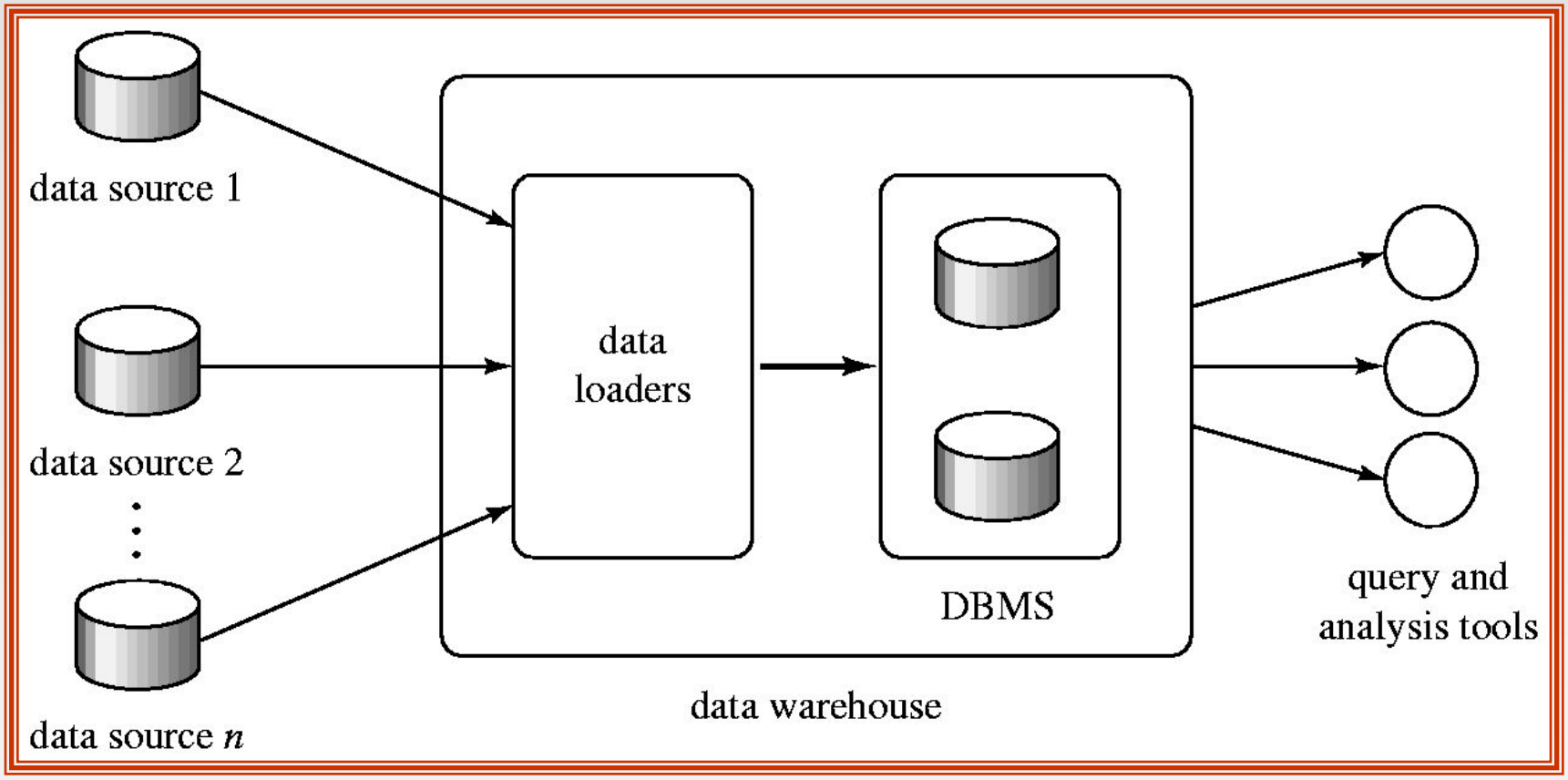
- For a given constant *n*, the ranking function ***ntile*(*n*)** takes the tuples in each partition in the specified order, and divides them into *n* buckets with equal numbers of tuples. For each tuple, ***ntile*(*n*)** then gives the number of the bucket in which it is placed, with bucket number starting with 1.
- Particularly useful for constructing histograms based on percentiles.
- E.g.:

```
  select threetile, sum(salary)
from (
  select salary, ntile(3) over (order by salary) as threetile
  from employee) as s
group by threetile
```

Data Warehousing

- Data sources often store only current data, not historical data.
- Large organizations have a complex internal organization structure and therefore different data may be present in different locations or on different operational systems or under different schemas.
- 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
  - Once gathered, the data are stored for a long time, permitting access to historical data
  - Provides the user a single consolidated interface to data, making decision support queries easier to write.
  - By accessing information for decision support from a data warehouse, the decision makers ensures that online-transaction processing system are not affected by the decision-support workload.

Data Warehouse Architecture



Component of a data warehouse

- Among the issues to be addressed in building a data warehouse are the following:
- ***When and how to gather data***
  - Source driven architecture: data sources transmit new information to warehouse, either continually (as transaction processing takes place) or periodically (e.g. at night)
  - Destination driven architecture: warehouse periodically requests new information from data sources
  - Keeping warehouse exactly synchronized with data sources (e.g. using two-phase commit) is 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***
  - Data sources that have been constructed independently are likely to have different schemas. In fact, they may even use different data models.

Component of a data warehouse (Cont.)

- Part of the task of the data warehouse is to perform schema integration and to convert data to the integrated schema before they are used.
- As a result, the data stored in the warehouse are not just a copy of the data at the sources. Instead, they can be thought of as a materialized view of the data at the sources.

- ***Data transformation and cleansing***
  - The task of correcting and preprocessing data is called data cleansing.
  - E.g. correct mistakes in addresses (misspellings, zip code errors)
  - Merge address lists from different sources and purge duplicates (deduplication)
  - Records for multiple individuals in a house may be grouped together so only one mailing is sent to each house; this operation is called **householding**.
  - Data may be **transformed** in ways other than cleansing, such as changing the units of measures or converting the data to a different schema by joining data from multiple source relations.

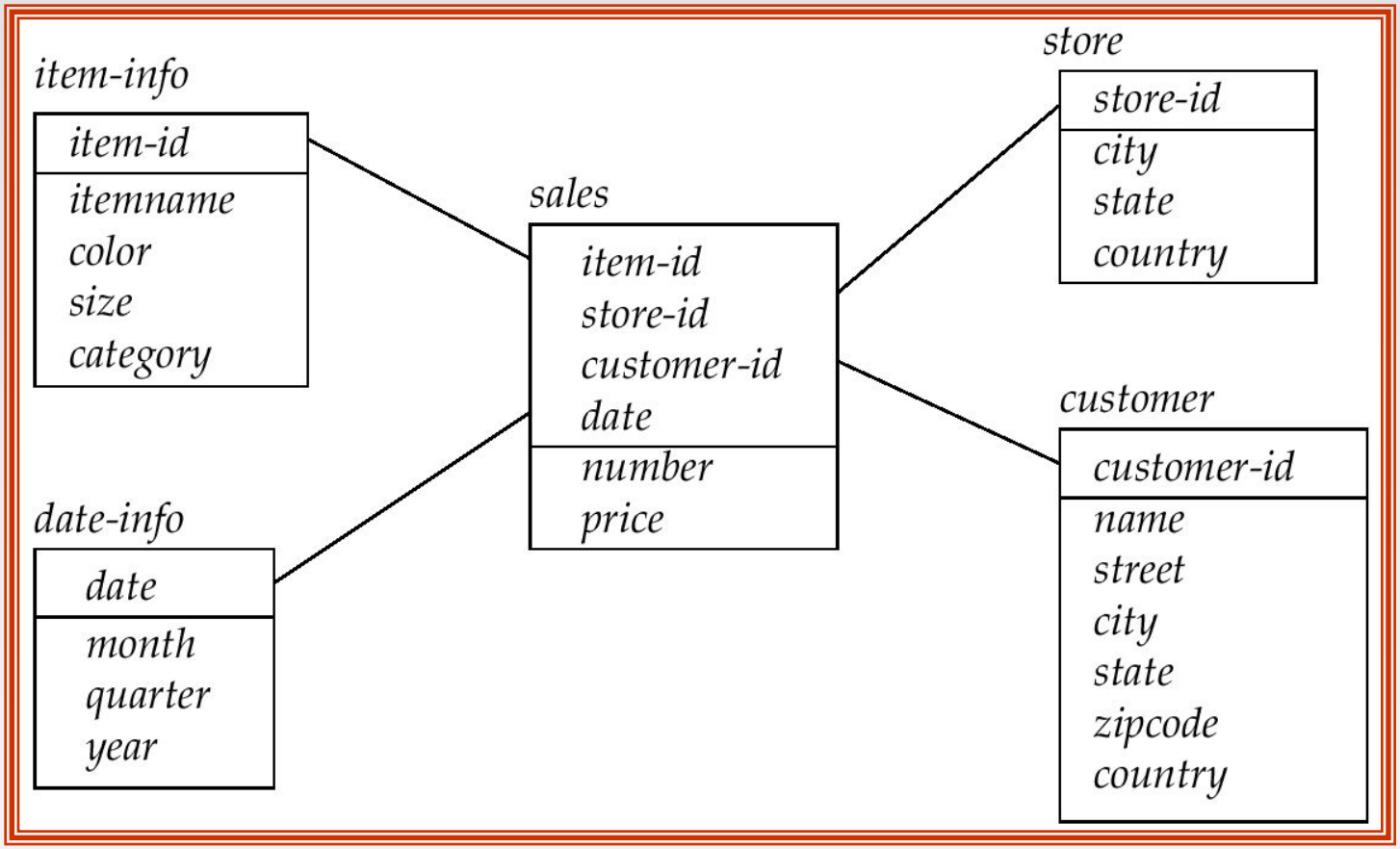
Component of a data warehouse (Cont.)

- ***How to propagate updates***
  - Updates on relations at the data sources must be propagated to the data warehouse.
  - If the relations are exactly same in both the places, the propagation is straight-forward.
  - Warehouse schema may be a (materialized) view of schema from data sources – propagating updates is simply view maintenance.
- ***What data to summarize***
  - Raw data generated by a transaction-processing system may be too large to store on-line
  - Aggregate values (totals/subtotals) often suffice, rather than maintaining the entire relation.
  - Queries on raw data can often be transformed by query optimizer to use aggregate values
- The different steps involved in getting data into a data warehouse are called as ***extract, transform and load or ETL.***

Warehouse Schemas

- Data warehouse typically have schemas that are designed for data analysis, using tools such as OLAP tools. Thus, the data are usually multidimensional data, with dimension attributes and measure attributes. Tables containing multidimensional data are called **fact tables** and are usually very large.
- Dimension values are usually encoded using small integers and mapped to full values via dimension tables
- Resultant schema is called a **star schema. A schema, with a fact table, multiple dimension tables and foreign keys from the fact table to dimension tables, is called a star schema.**
  - More complicated schema structures
    - Snowflake schema: multiple levels of dimension tables
    - Constellation: multiple fact tables

Data Warehouse Schema



Star schema for a data warehouse