



Chapter 11: Data Analytics

Database System Concepts, 7th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use



Chapter 11: Data Analytics

- Overview
- Data Warehousing
- Online Analytical Processing (OLAP)
- Data Mining



Overview

- Decision-making tasks benefit greatly by using data about the past to predict the future and using the predictions to make decisions.
- **Data analytics**: refers broadly to 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
- The financial benefits of making correct decisions can be substantial, as can the costs of making wrong decisions.
- Critical for businesses today. Organizations therefore invest a lot of money to gather or purchase required data and build systems for data analytics.



Overview (Cont.)

- Large companies have diverse sources of data that they need to use for making business decisions.
 - The sources may store the data under different schemas.
 - For performance reasons (as well as for reasons of organization control), the data sources usually will not permit other parts of the company to retrieve data on demand
 - Organizations therefore typically gather data from multiple sources into one location, referred to as a *data warehouse*.
- Common steps in data analytics
 - 1. Gather data from multiple sources into one location (Data Warehouse)
 - Data warehouses also integrated data into common schema
 - Data warehouses today also collect and store data from non-relational sources, where schema unification is not possible
 - 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)**



Overview (Cont.)

- 2. Generate aggregates and reports summarizing data
 - Analysts need to get a number of different aggregates and compare them to understand patterns in the data
 - Dashboards show graphical charts/reports summarizing key organizational parameters, such as sales, expenses, product returns, and so forth (health of an organization)
 - Analysts also need to visualize data in ways that can highlight anomalies or give insights into causes for changes in the business
 - **Online analytical processing (OLAP) systems** allow interactive querying
 - OLAP support very efficient analysis, where aggregate queries on large data are answered in almost real time (as opposed to being answered after tens of minutes or multiple hours) are popular with analysts.
 - Statistical analysis is also done using tools such as R/SAS/SPSS
 - Including extensions for parallel processing of big data
- 3. 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 from these patterns.
- **Data mining** combines knowledge-discovery techniques developed by machine-learning communities to run them on extremely large databases
- The term **business intelligence (BI)** is synonym for data analytics
- The term **decision support** is also used in a related but narrower sense, with a focus on reporting and aggregation, but not including machine learning/data mining.
- Decision-support tasks typically use SQL queries to process large amounts of data. Decision-support queries are contrasted with queries for *online transaction processing*, where each query typically reads only a small amount of data and may perform a few small updates.



DATA WAREHOUSING

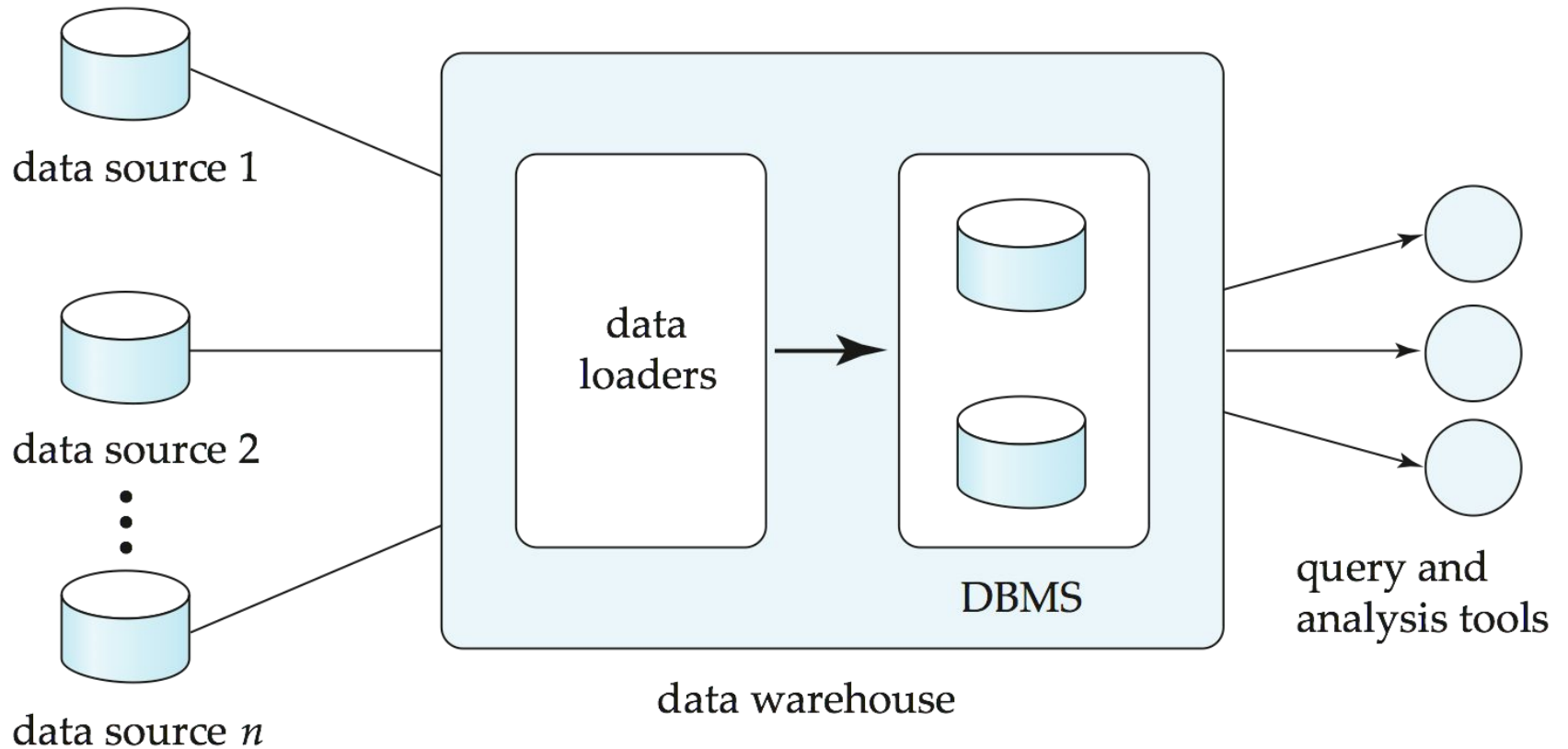


Data Warehousing

- 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.
- Data may also be purchased externally.
- A **data warehouse** is a repository (archive) of information gathered from multiple sources, stored under a unified schema, at a single site. Thus they provide the user a single uniform interface to data.
 - Greatly simplifies querying, permits study of historical trends
 - 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.
 - Shifts decision support query load away from transaction processing systems



Data Warehousing





Design Issues

- Issues to be addressed in building a data warehouse
- *When and how to gather data*
 - **Source driven architecture:** data sources transmit new information to warehouse
 - either continuously (as transaction processing takes place) 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 from online transaction processing (OLTP) systems.



More Warehouse Design Issues

- ***What schema to use*** (Schema integration)
 - Data sources that have been constructed independently are likely to have different schemas. In fact, they may even use different data models.
 - 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 data cleansing***
 - Some sources of data have errors that can be detected and corrected using business constraints; further, organizations may collect data from multiple sources, and there may be duplicates in the data collected from different sources.
 - E.g., correct mistakes in addresses (misspellings, zip code errors)
 - **Merge** address lists from different sources and **purge** duplicates (deduplication)
 - Changing the units of measures or converting the data to a different schema by joining data from multiple source relations.



More Warehouse Design Issues

■ *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 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
- These steps of collecting data, cleaning/deduplicating the data, and loading the data into a warehouse are referred to as *extract, transform and load* (ETL).

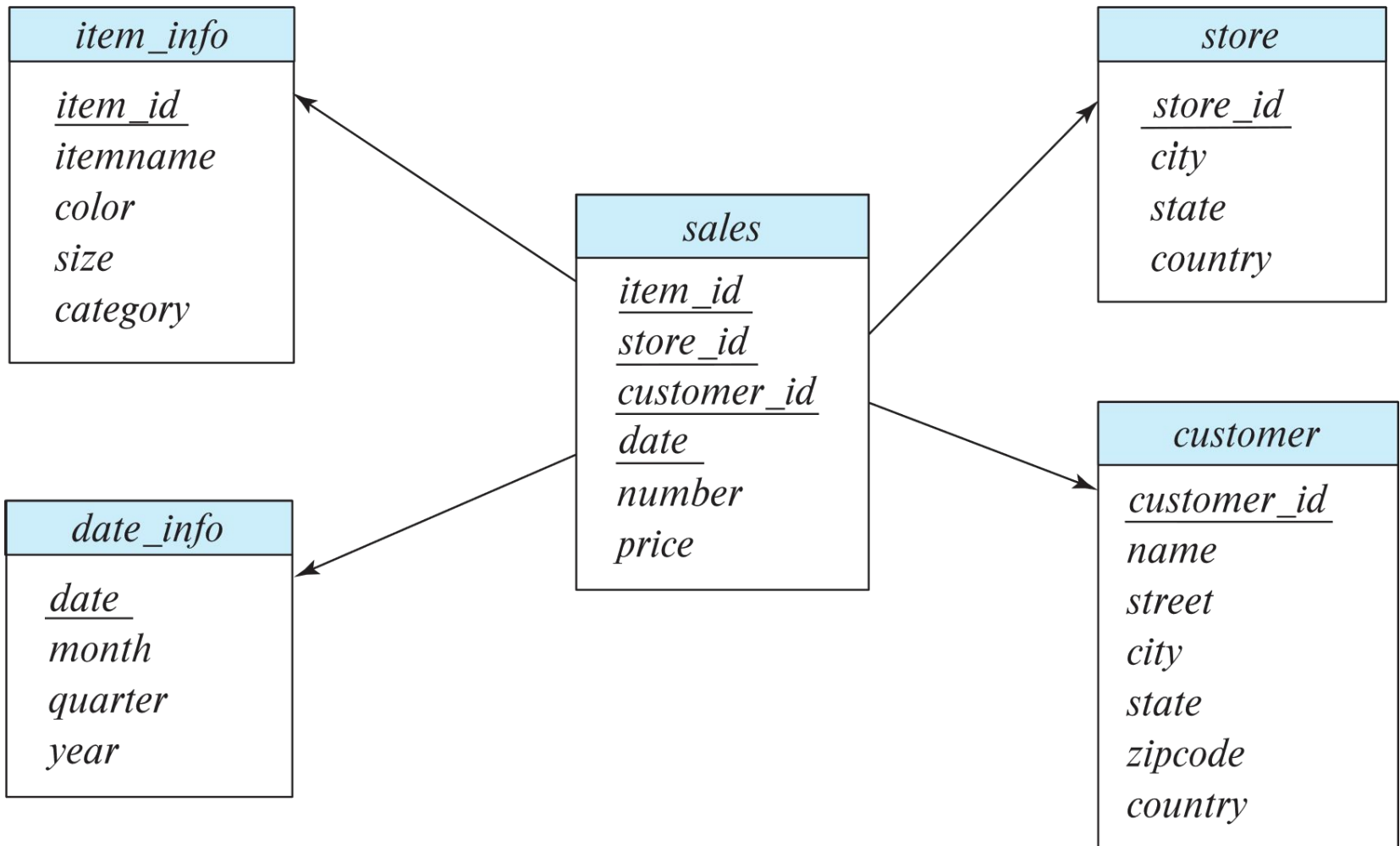


Multidimensional Data and Warehouse Schemas

- Data warehouse typically have schemas that are designed for data analysis, using tools such as OLAP. Thus, the data are usually multidimensional data, with dimension attributes and measure attributes.
- 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 grouped and viewed
 - e.g., attributes *item_id*, *date*, *location* and *customer* of the *sales* rela.
 - 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**. 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 schema**: 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



Data Lakes

- While data warehouses pay a lot of attention to ensuring a common data schema to ease the job of querying the data, there are situations where organizations want to store data without paying the cost of creating a common schema and transforming data to the common schema.
- The term data lake is used to refer to a repository where data can be stored in multiple formats, including structured records and unstructured file formats.
- Unlike data warehouses, data lakes do not require up-front effort to preprocess data, but they do require more effort when creating queries.
- Since data may be stored in many different formats, querying tools also need to be quite flexible.
- Apache Hadoop and Apache Spark are popular tools for querying such data, since they support querying of both unstructured and structured data.

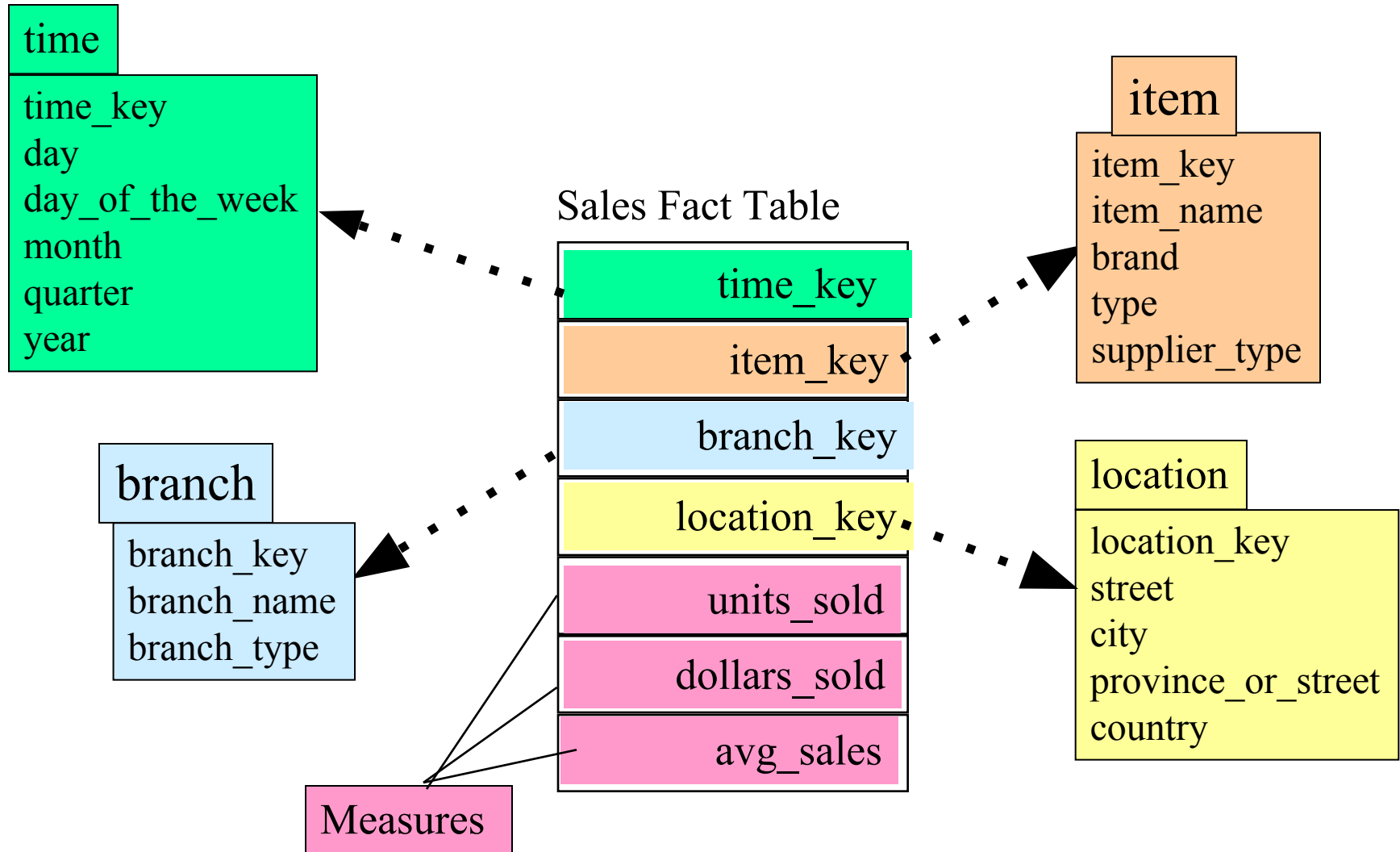


Conceptual Modeling of Data Warehouses

- Modeling data warehouses: dimensions and measures
 - Star schema: A fact table in the middle connected to a set of dimension tables
 - Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
 - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

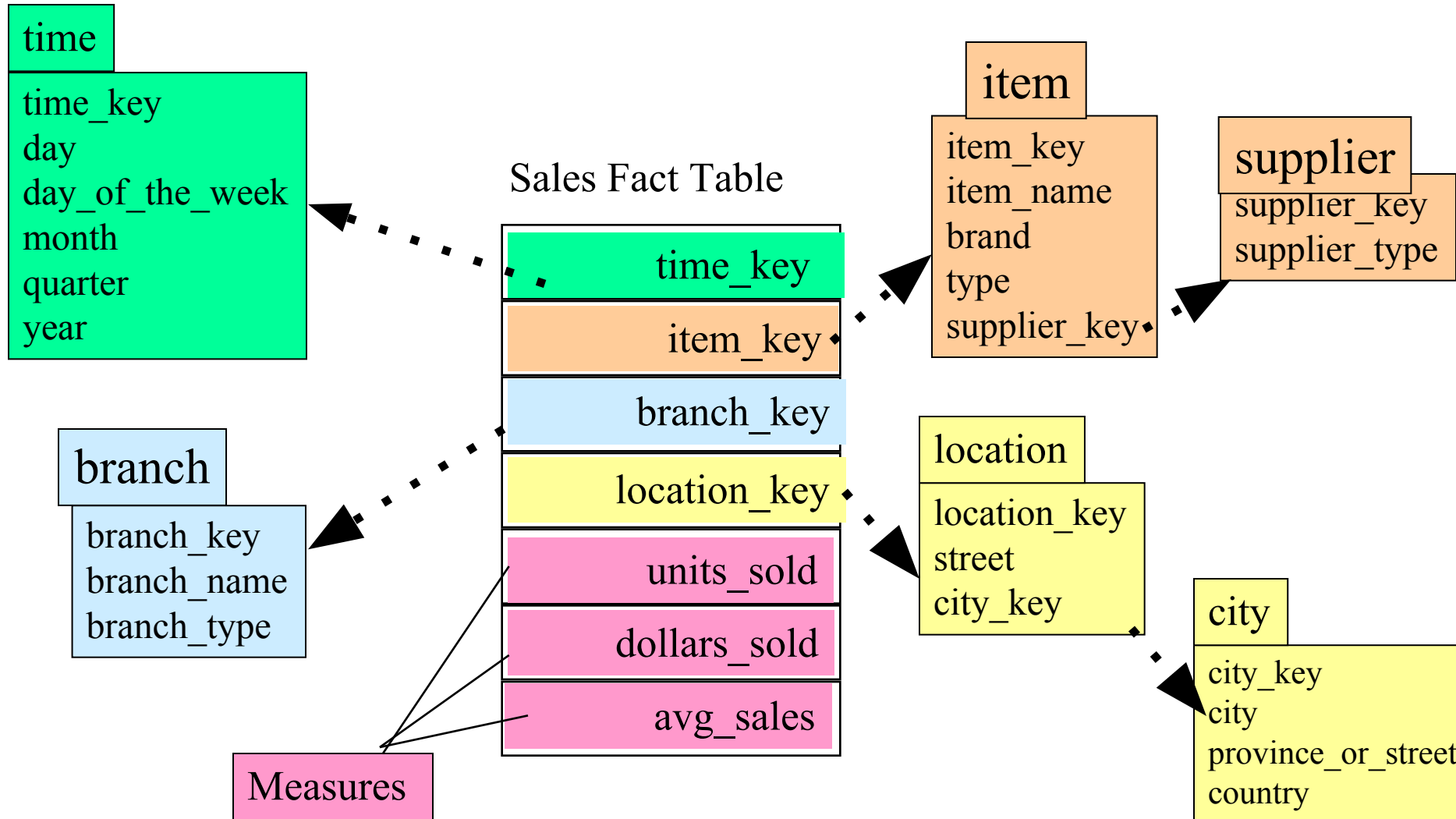


Example of Star Schema



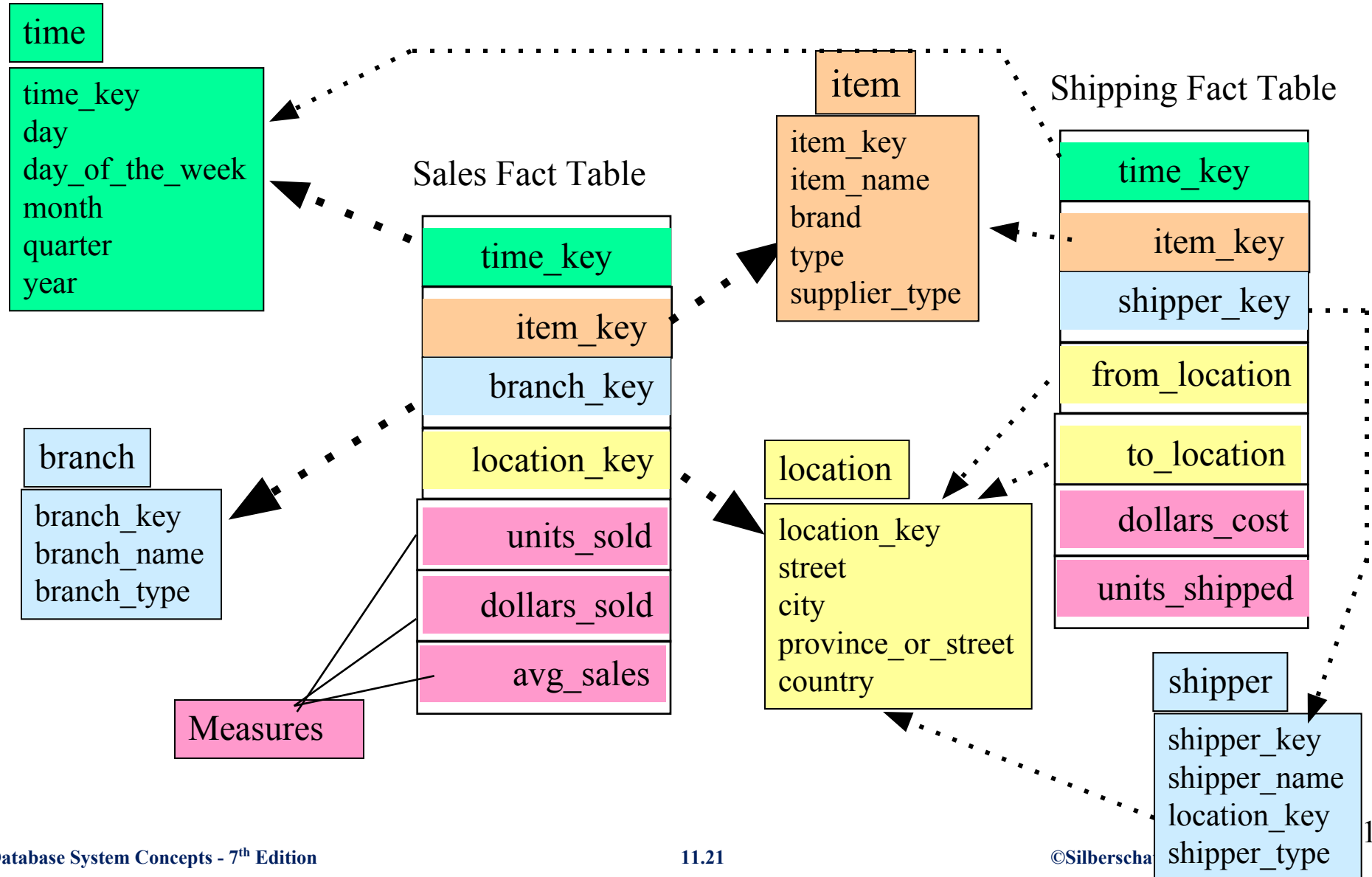


Example of Snowflake Schema





Example of Fact Constellation





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

- Data analysis often involves looking for patterns that arise when data values are grouped in “interesting” ways. In a retail business, we might group sales by product, the date or month of the sale, the color or size of the product, or the profile (such as age group and gender) of the customer who bought the product .
- **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 an application where a shop wants to find out what kinds of clothes are popular. 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.



Data Analysis and 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**.



Example sales relation

<i>item_name</i>	<i>color</i>	<i>clothes_size</i>	<i>quantity</i>
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**

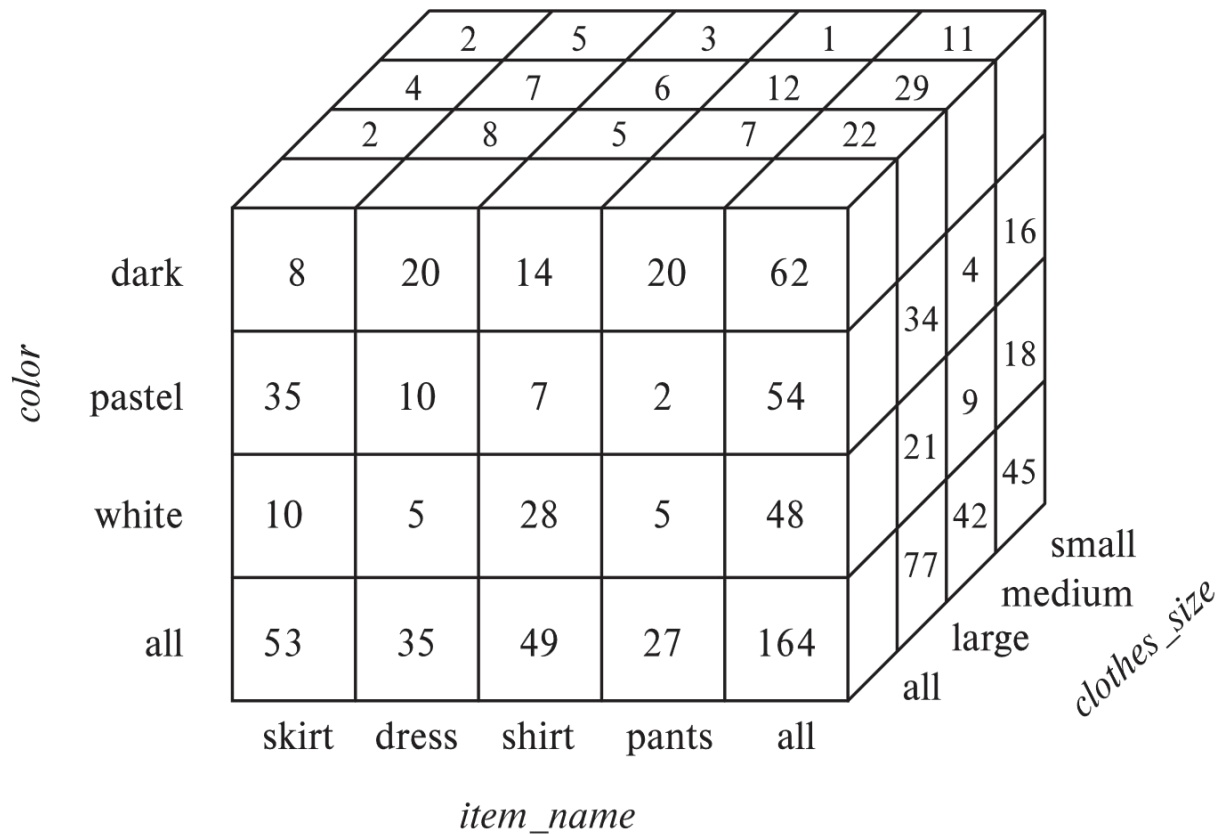
		<i>color</i>			
		dark	pastel	white	total
<i>item_name</i>	skirt	8	35	10	53
	dress	20	10	5	35
	shirt	14	7	28	49
	pants	20	2	5	27
	total	62	54	48	164

- The table above is an example of a **cross-tabulation** (**cross-tab**), also referred to as a **pivot-table**. In general, a cross-tab is a table derived from a relation, where
 - 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



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





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.
- 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 Operations

- 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 *color* and *size* or *color* and *item-name*.
- **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., 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.



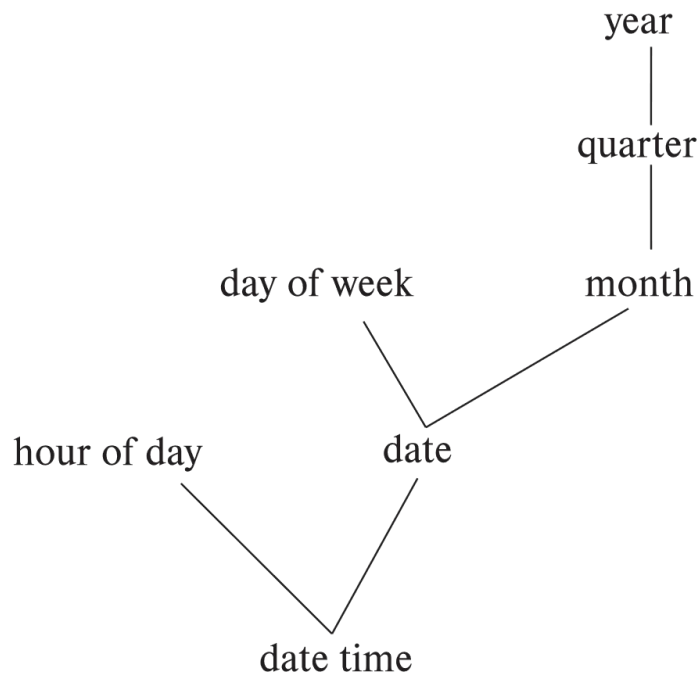
Online Analytical Processing Operations

- 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.
 - 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
- 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

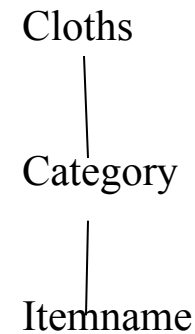
- **Hierarchy** on dimension attributes: lets dimensions be viewed at different levels of detail
- E.g., the dimension *datetime* can be used to aggregate by hour of day, date, day of week, month, quarter or year



(a) time hierarchy



(b) location hierarchy



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
- E.g. hierarchy: *item_name* □ *category*

clothes_size: **all**

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



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
- We use the value **all** to represent aggregates/sub-totals.
- The SQL standard actually uses *null* values in place of **all** but can cause confusion with regular null values.

<i>item_name</i>	<i>color</i>	<i>clothes_size</i>	<i>quantity</i>
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

- Microsoft SQL Server and Oracle, support a **pivot** clause that allows creation of cross-tabs.
- select ***
from *sales*
pivot (
 sum(*quantity*)
 for *color* **in** ('dark','pastel','white')
)
order by *item name*;

<i>item_name</i>	<i>clothes_size</i>	<i>dark</i>	<i>pastel</i>	<i>white</i>
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

- SQL supports generalizations of the **group by** construct to perform the **cube** and **rollup** operations.
- 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.
- Data cube relations are often very large. The cube query above, with 3 possible colors, 4 possible item names, and 3 sizes and with summary values has 80 tuples.



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 relation of right side is generated by doing a **group by cube** on *item name* and *color*, with an extra column specified in the **select** clause showing **all** for *clothes_size*.

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



Online Analytical Processing Operations

- 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);
```

- 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 is the same as the **cube** construct except that **rollup** generates fewer **group by** queries. 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

$$\{item_name, ()\} \times \{(color, size), (color), ()\}$$
$$= \{ (item_name, color, size), (item_name, color), (item_name), (color, size), (color), () \}$$



Extended Aggregation (Cont.)

- Neither the **rollup** nor the **cube** clause gives complete control on the groupings that are generated.
- For instance, we cannot use them to specify that we want only groupings $\{(color, clothes_size), (clothes_size, item_name)\}$. Such restricted groupings can be generated by using the **grouping sets** construct, in which one can specify the specific list of groupings to be used.
- To obtain only groupings $\{(color, clothes_size), (clothes_size, item_name)\}$, we would write:
- ```
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 | 1,500,000 |
|             | Computer Software | 500,000   |           |
|             | All categories    |           |           |
| South       | Computer Hardware | 200,000   | 600,000   |
|             | Computer Software | 400,000   |           |
|             | All categories    |           |           |
| Total Sales |                   |           | 2,100,000 |



# DATA MINING



# Data Mining

- **Data mining** is the process of semi-automatically analyzing large databases to find useful patterns
  - Similar goals to machine learning, but on very large volumes of data
- Part of the larger area of **knowledge discovery in databases (KDD)**
- Some types of knowledge can be represented as rules. “*Young women with annual incomes greater than 50000 K are the most likely people to buy small sport cars.*”
- Other types of knowledge are represented by equations relating different variables to each others or by mechanisms for predicting outcomes when the values of some variables are known.
- There are a variety of possible types of pattern that may be useful and different techniques are used to find different types of patterns.
- More generally, knowledge is discovered by applying machine learning techniques on past instances of data, to form a **model**
  - Model is then used to make predictions for new instances



# Data Mining

- Usually there is a manual component to data mining, consisting of preprocessing data to a form acceptable to the algorithms and post-processing of discovered patterns to find novel ones that could be useful.
- There may also be more than one type of pattern that can be discovered from a given database and manual interaction may be needed to pick useful types of patterns. For this reason, data mining is really a semiautomatic process in real life.

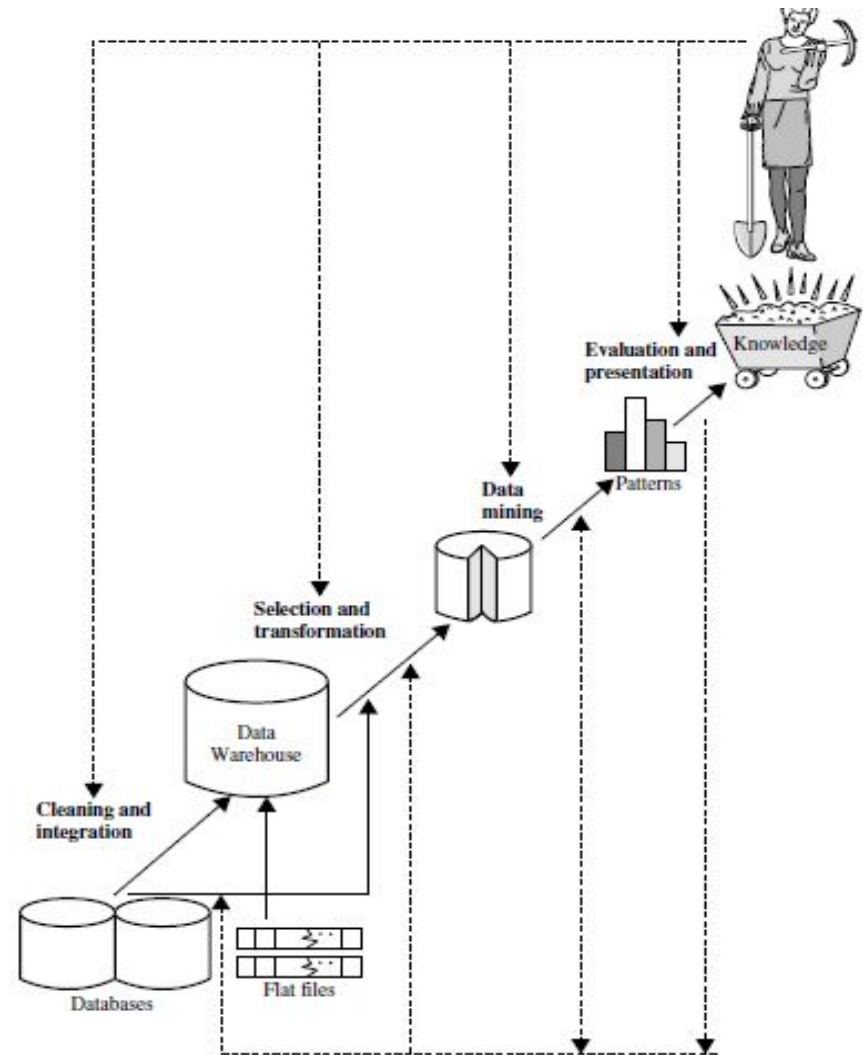


Figure 1.4 Data mining as a step in the process of knowledge discovery.





# Types of Data Mining Tasks

- The discovered knowledge has numerous applications:
- **Prediction** based on past history
  - Predict if a credit card applicant poses a good credit risk, based on some attributes (income, job type, age, debt and past debt payment history)
  - Predict if customers may switch over to a competitor
  - Predict which people are likely to respond to promotional mail (“junk mail”)
  - Predict if a pattern of phone calling card usage is likely to be fraudulent
- Some examples of prediction mechanisms:
  - **Classification**
    - Items (with associated attributes) belong to one of several classes
    - **Training instances** have attribute values and classes provided
    - Given a new item whose class is unknown, predict to which class it belongs based on its attribute values



# Data Mining (Cont.)

- **Regression** formulae
  - Given a set of mappings for an unknown function, predict the function result for a new parameter value
- **Descriptive Patterns**
  - **Associations**
    - Find books that are often bought by “similar” customers. If a new such customer buys one such book, suggest the others too.
    - Associations may be used as a first step in detecting **causation**
      - E.g., association between exposure to chemical X and cancer
      - E.g. discovery of an unexpected association between a newly introduced medicine and cardiac problems led to the finding that the medicine may cause cardiac problems in some people. The medicine was then withdrawn from the market.
  - **Clusters**
    - E.g., typhoid cases were clustered in an area surrounding a contaminated well
    - Detection of clusters remains important in detecting epidemics

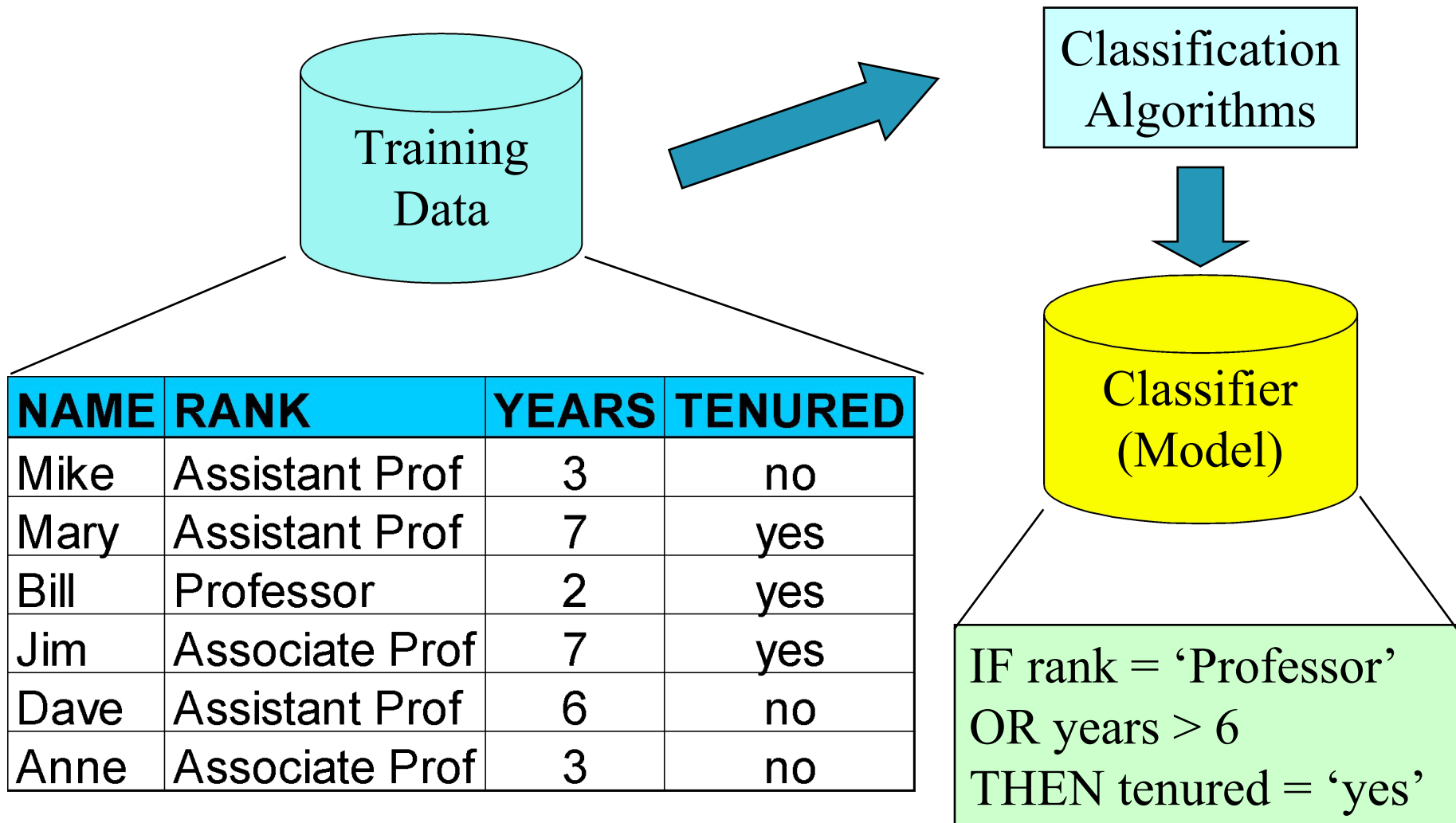


# Classification Rules

- The **classification** problem is this: Given that items belong to one of the several classes and given past instances (called **training instances**) of items along with the classes to which they belong, the problem is to predict the class to which a new item belongs. The class of the new instance is not known, so other attributes of the instance must be used to predict the class.
- Classification rules help assign new objects to classes. Classification can be done by finding rules that partition the given data into disjoint groups.
  - E.g., given a new automobile insurance applicant, should he or she be classified as low risk, medium risk or high risk?
- Classification rules for above example could use a variety of data, such as educational level, salary, age, etc.
  - $\forall$  person P, P.degree = masters **and** P.income > 75,000  
 $\Rightarrow$  P.credit = excellent
  - $\forall$  person P, P.degree = bachelors **and**  
(P.income  $\geq$  25,000 and P.income  $\leq$  75,000)  
 $\Rightarrow$  P.credit = good
- Rules are not necessarily exact: there may be some misclassifications
- Classification rules can be shown compactly as a decision tree.

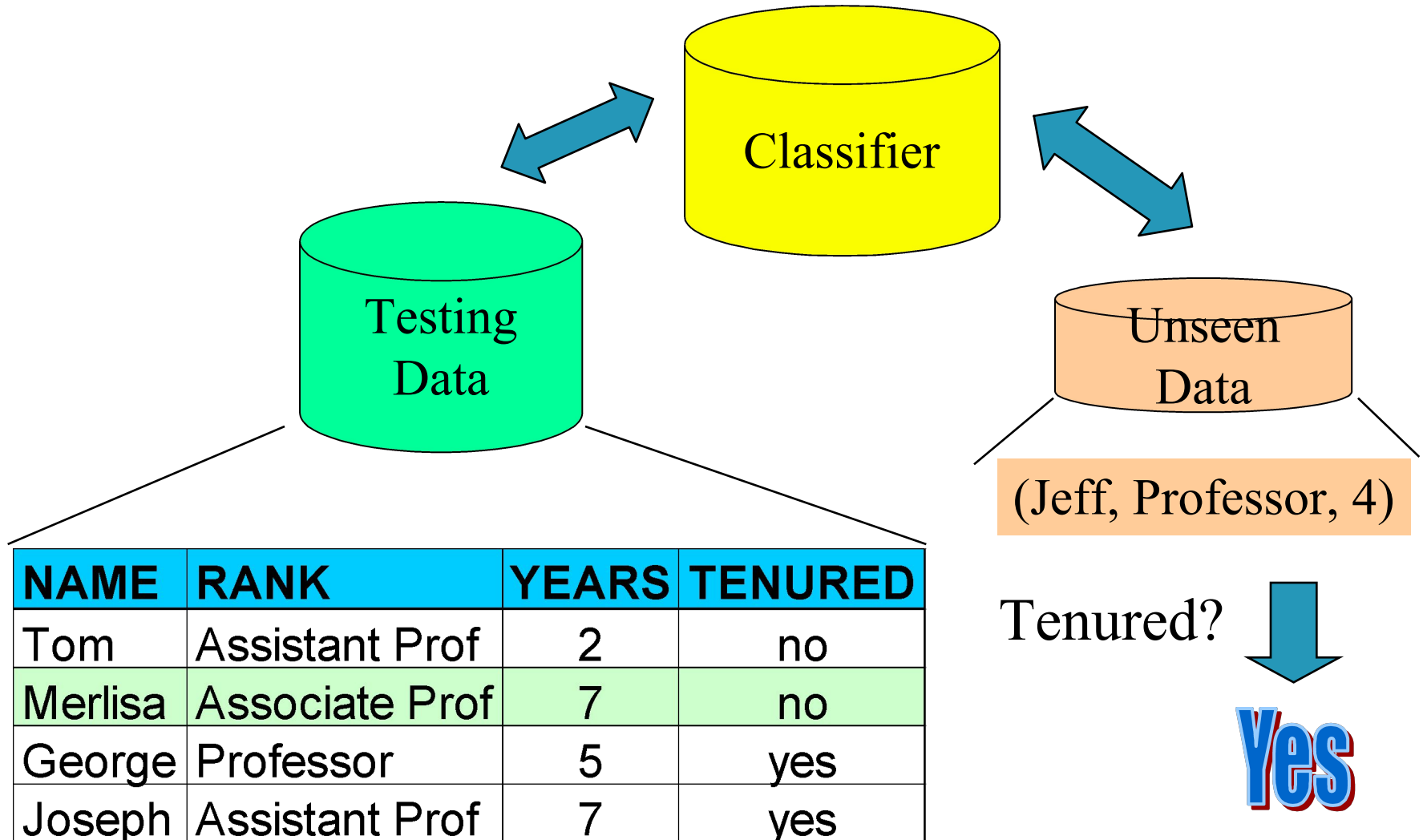


## Process (1): Model Construction





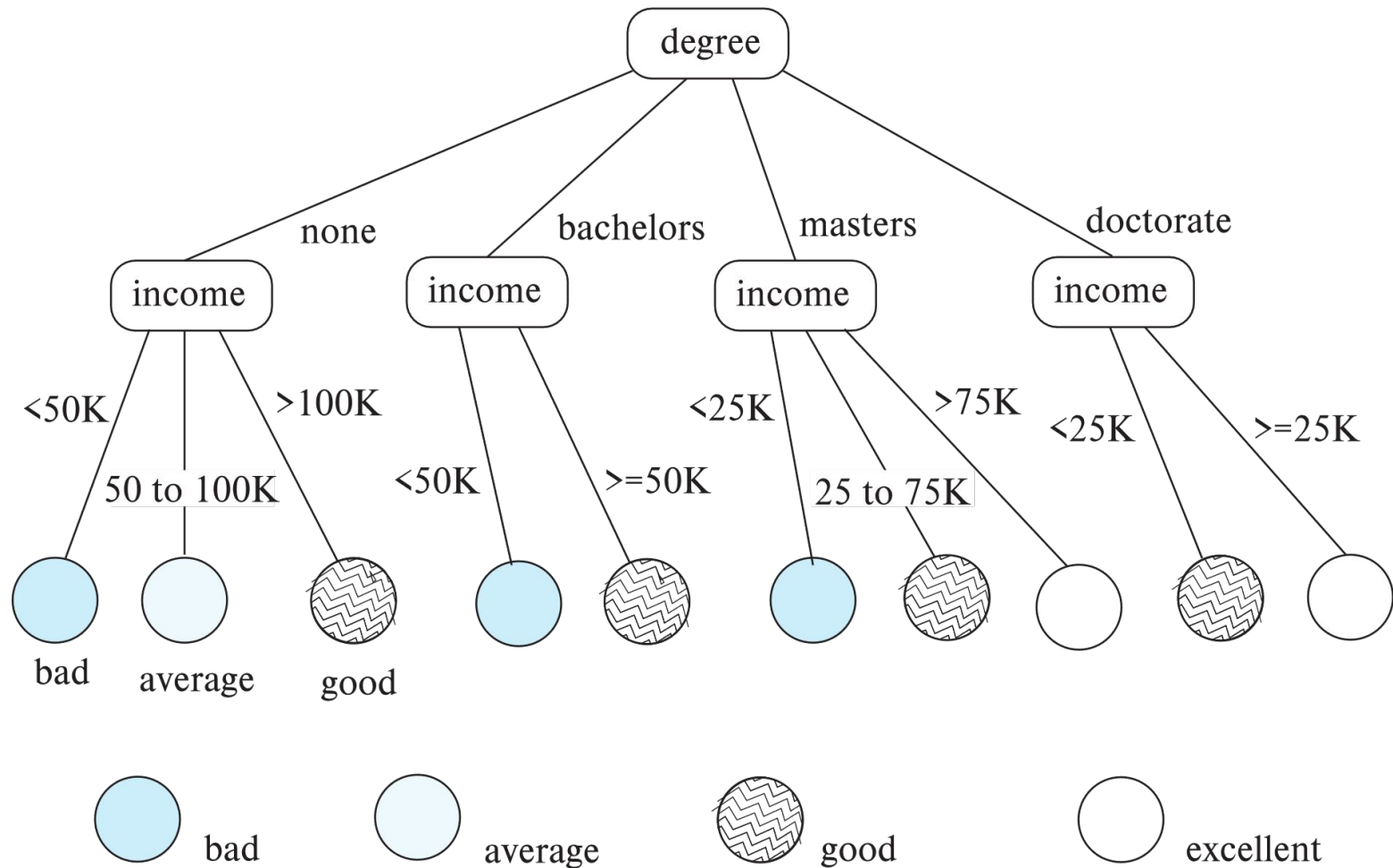
## Process (2): Using the Model in Prediction





# Decision Tree Classifiers

The decision tree classifier is a widely used technique for classification. It uses a tree; each leaf node has an associated class and each internal node has a predicate associated with it.





# Dataset for Playing Tennis

| Day | Outlook  | Temperature | Humidity | Wind   | Play Tennis |
|-----|----------|-------------|----------|--------|-------------|
| 1   | Sunny    | Hot         | High     | Weak   | No          |
| 2   | Sunny    | Hot         | High     | Strong | No          |
| 3   | Overcast | Hot         | High     | Weak   | Yes         |
| 4   | Rain     | Mild        | High     | Weak   | Yes         |
| 5   | Rain     | Cool        | Normal   | Weak   | Yes         |
| 6   | Rain     | Cool        | Normal   | Strong | No          |
| 7   | Overcast | Cool        | Normal   | Strong | Yes         |
| 8   | Sunny    | Mild        | High     | Weak   | No          |
| 9   | Sunny    | Cool        | Normal   | Weak   | Yes         |
| 10  | Rain     | Mild        | Normal   | Weak   | Yes         |
| 11  | Sunny    | Mild        | Normal   | Strong | Yes         |
| 12  | Overcast | Mild        | High     | Strong | Yes         |
| 13  | Overcast | Hot         | Normal   | Weak   | Yes         |
| 14  | Rain     | Mild        | High     | Strong | No          |

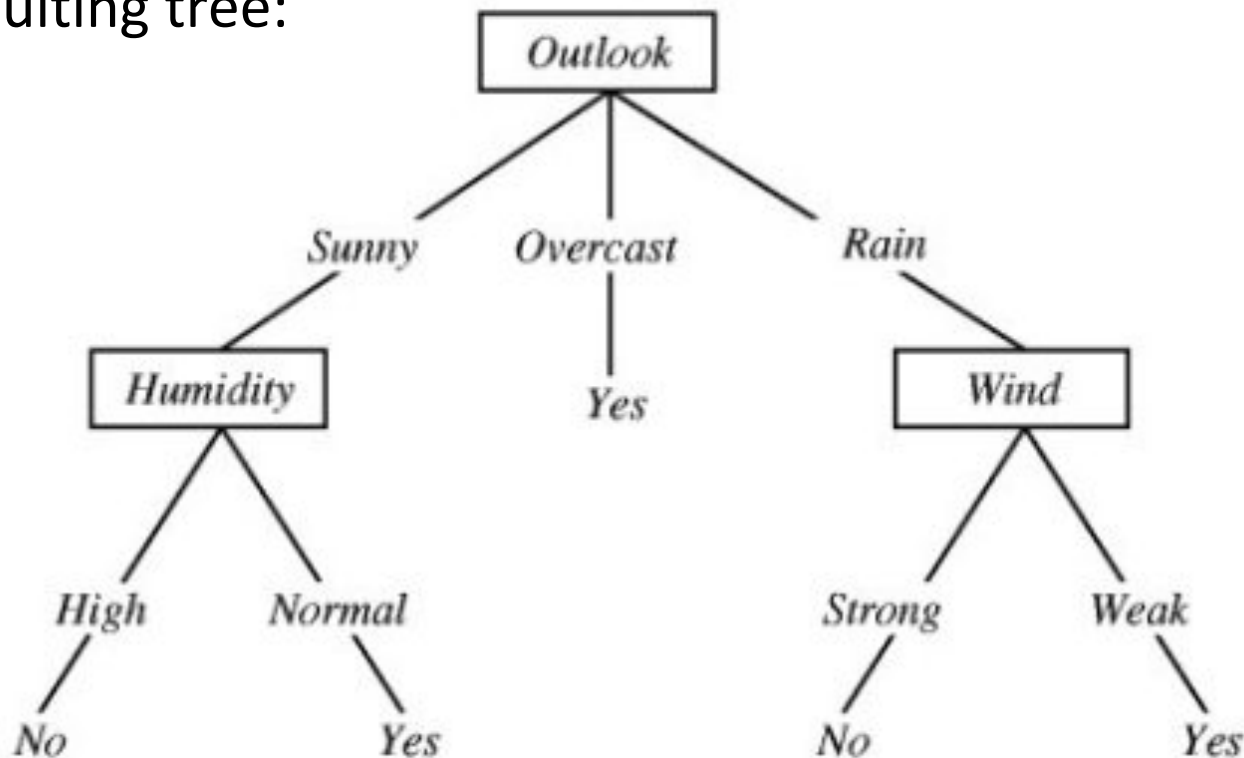
In this example,

$$Entropy(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right) = 0.9450 \quad \text{Eq.3}$$



# Decision Tree for Playing Tennis

- ❑ Based on ID3 (Iterative Dichotomiser) algorithm invented by Ross Quinlan (1986).
- ❑ Uses Entropy based Information Gain
- ❑ Information Gain of Outlook, Humidity, Windy and Temperature are 0.246, 0.152, 0.048, 0.029 respectively
- ❑ Resulting tree:



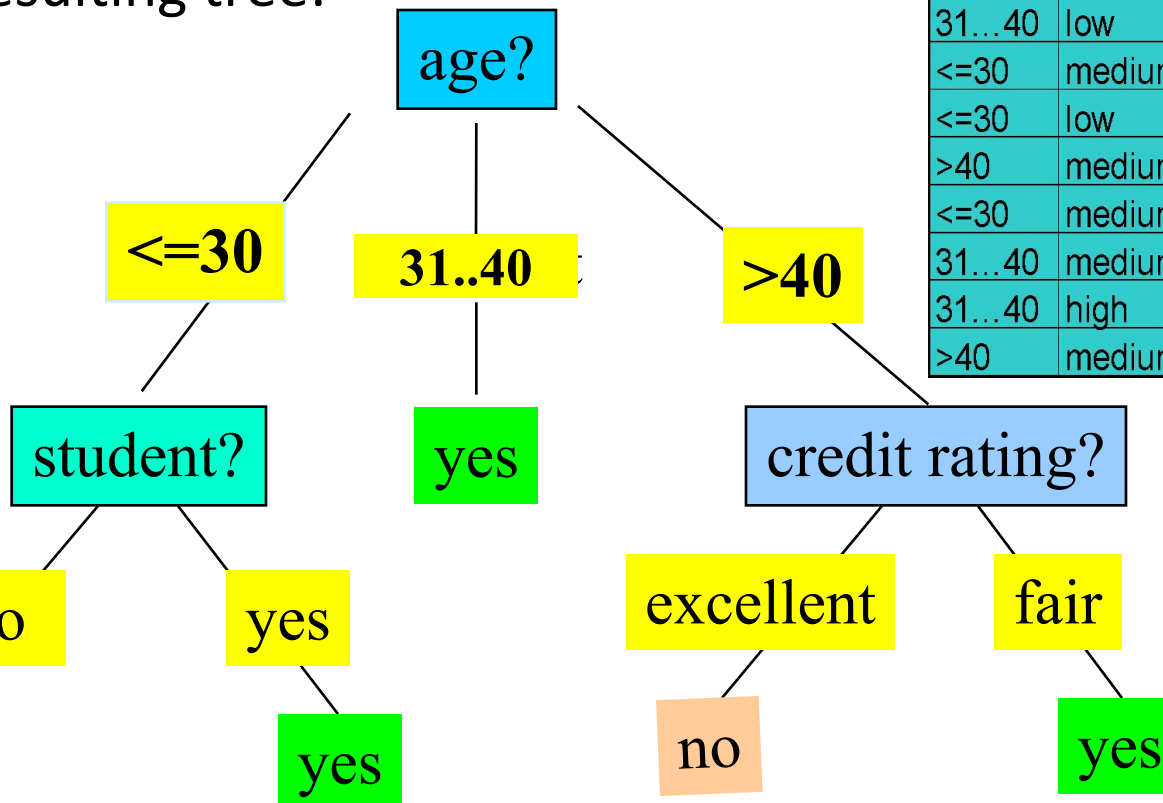




# Decision Tree Induction: An Example

- ❑ Training data set: Buys\_computer
- ❑ The data set follows an example of Quinlan's ID3 (Playing Tennis)
- ❑ Resulting tree:

| age     | income | student | credit rating | buys_computer |
|---------|--------|---------|---------------|---------------|
| <=30    | high   | no      | fair          | no            |
| <=30    | high   | no      | excellent     | no            |
| 31...40 | high   | no      | fair          | yes           |
| >40     | medium | no      | fair          | yes           |
| >40     | low    | yes     | fair          | yes           |
| >40     | low    | yes     | excellent     | no            |
| 31...40 | low    | yes     | excellent     | yes           |
| <=30    | medium | no      | fair          | no            |
| <=30    | low    | yes     | fair          | yes           |
| >40     | medium | yes     | fair          | yes           |
| <=30    | medium | yes     | excellent     | yes           |
| 31...40 | medium | no      | excellent     | yes           |
| 31...40 | high   | yes     | fair          | yes           |
| >40     | medium | no      | excellent     | no            |





# Construction of Decision Trees

- **Training set**: a data sample in which the classification is already known.
- **Greedy** top down generation of decision trees.
  - Each internal node of the tree partitions the data into groups based on a **partitioning attribute**, and a **partitioning condition** for the node
  - **Leaf** node:
    - all (or most) of the items at the node belong to the same class, or
    - all attributes have been considered, and no further partitioning is possible.
- Traverse tree from top to make a prediction
- Number of techniques for constructing decision tree classifiers



# Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a **top-down recursive divide-and-conquer manner**
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., **information gain**)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – **majority voting** is employed for classifying the leaf
  - There are no samples left



# Bayesian Classifiers

- **Bayesian classifiers** find the distribution of attribute values for each class in the training data; when given a new instance  $d$ , they use the distribution information to estimate, for each class  $c_j$ , the probability that instance  $d$  belongs to class  $c_j$ , denoted by  $p(c_j|d)$ , in a manner outlined here.
- The class with maximum probability becomes the predicted class for instance  $d$
- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j | d) = \frac{p(d | c_j) p(c_j)}{p(d)}$$

where

$p(c_j | d)$  = probability of instance  $d$  being in class  $c_j$ ,

$p(d | c_j)$  = probability of generating instance  $d$  given class  $c_j$ ,

$p(c_j)$  = probability of occurrence of class  $c_j$ , and

$p(d)$  = probability of instance  $d$  occurring



# Naïve Bayesian Classifiers

- Bayesian classifiers require
  - computation of  $p(d | c_j)$
  - precomputation of  $p(c_j)$
  - $p(d)$  can be ignored since it is the same for all classes
- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

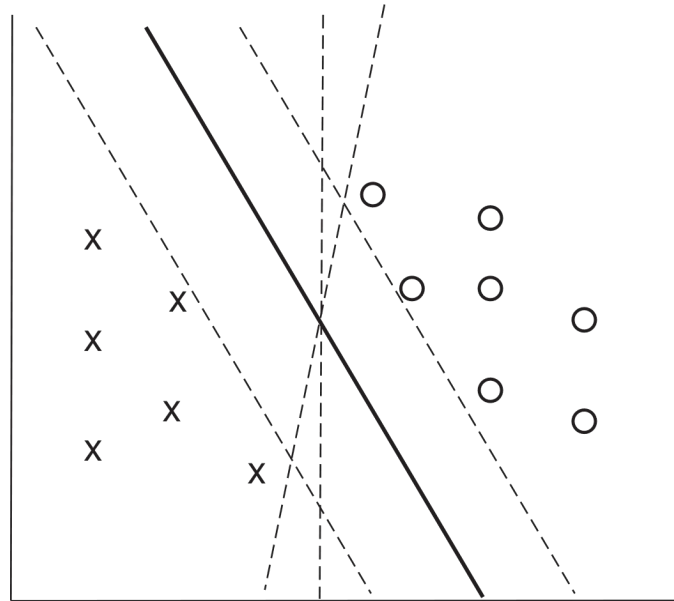
$$p(d | c_j) = p(d_1 | c_j) * p(d_2 | c_j) * \dots * (p(d_n | c_j))$$

- Each of the  $p(d_i | c_j)$  can be estimated from a histogram on  $d_i$  values for each class  $c_j$ 
  - the histogram is computed from the training instances
- Histograms on multiple attributes are more expensive to compute and store



# Support Vector Machine Classifiers

- Simple 2-dimensional example:
  - Points are in two classes
  - Find a line (**maximum margin line**) s.t. line divides two classes, and distance from nearest point in either class is maximum





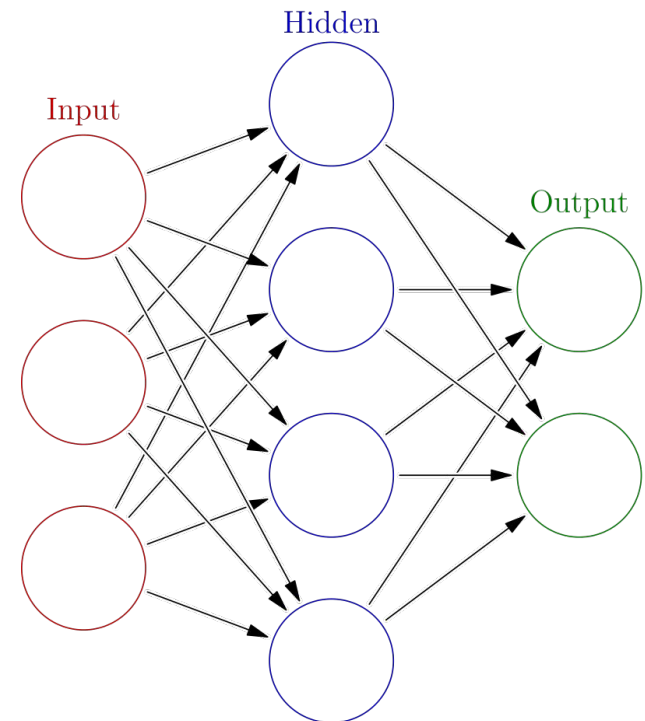
# Support Vector Machine

- In  $n$ -dimensions points are divided by a plane, instead of a line
- SVMs can be used separators that are curve, not necessarily linear, by transforming points before classification
  - Transformation functions may be non-linear and are called kernel functions
  - Separator is a plane in the transformed space, but maps to curve in original space
- There may not be an exact planar separator for a given set of points
  - Choose plane that best separates points
- $N$ -ary classification can be done by  $N$  binary classifications
  - In class  $i$  vs. not in class  $i$ .



# Neural Network Classifiers

- Neural network has multiple layers
  - Each layer acts as input to next later
- First layer has input nodes, which are assigned values from input attributes
- Each node combines values of its inputs using some weight function to compute its value
  - Weights are associated with edges
- For classification, each output value indicates likelihood of the input instance belonging to that class
  - Pick class with maximum likelihood
- Weights of edges are key to classification
- Edge weights are learnt during training phase

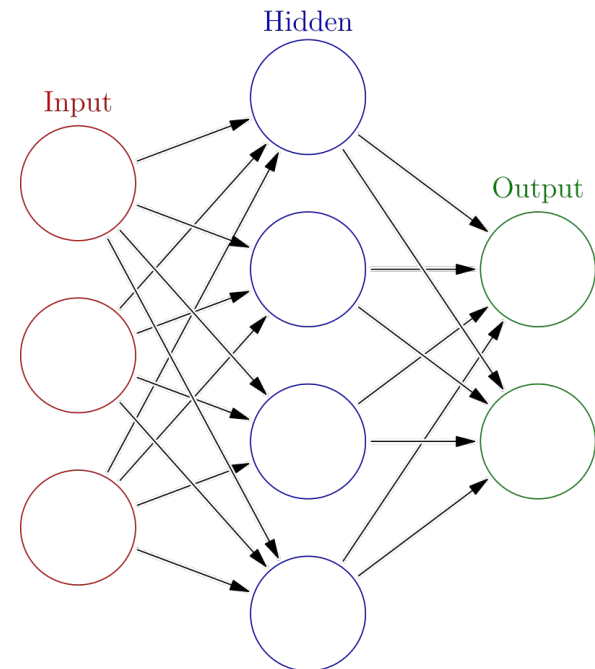
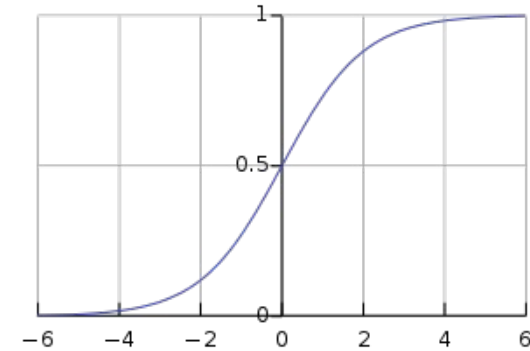






# Neural Network Classifiers

- Value of a node may be linear combination of inputs, or may be a non-linear function
  - E.g., sigmoid function
- **Backpropagation algorithm** works as follows
  - Weights are set randomly initially
  - Training instances are processed one at a time
    - Output is computed using current weights
    - If classification is wrong, weights are adjusted to get a higher score for the correct class





# Neural Networks (Cont.)

- **Deep neural networks** have a large number of layers with large number of nodes in each layer
- **Deep learning** refers to training of deep neural network on very large numbers of training instances
- Each layer may be connected to previous layers in different ways
  - Convolutional networks used for image processing
  - Recurrent neural network, LSTM (Long-Short Term Memory)
  - More complex architectures used for text processing, and machine translation, speech recognition, etc.
- Neural networks are a large area in themselves
  - Further details beyond scope of this chapter



# Regression

- Regression deals with the prediction of a value, rather than a class.
  - Given values for a set of variables,  $X_1, X_2, \dots, X_n$ , we wish to predict the value of a variable  $Y$ .
- One way is to infer coefficients  $a_0, a_1, a_1, \dots, a_n$  such that
$$Y = a_0 + a_1 * X_1 + a_2 * X_2 + \dots + a_n * X_n$$
- Finding such a linear polynomial is called **linear regression**.
  - In general, the process of finding a curve that fits the data is also called **curve fitting**.
- The fit may only be approximate
  - because of noise in the data, or
  - because the relationship is not exactly a polynomial
- Regression aims to find coefficients that give the best possible fit.



# Association Rules

- Retail shops are often interested in associations between different items that people buy.
  - Someone who buys bread is quite likely also to buy milk
  - A person who bought the book *Database System Concepts* is quite likely also to buy the book *Operating System Concepts*.
- Associations information can be used in several ways.
  - E.g. when a customer buys a particular book, an online shop may suggest associated books.
- **Association rules:**  
 $bread \Rightarrow milk$        $DB\text{-}Concepts, OS\text{-}Concepts \Rightarrow Networks$ 
  - Left hand side: **antecedent**,    right hand side: **consequent**
  - An association rule must have an associated **population**; the population consists of a set of **instances**
    - E.g. each transaction (sale) at a shop is an instance, and the set of all transactions is the population



# Association Rules (Cont.)

- Rules have an associated support, as well as an associated confidence.
- **Support** is a measure of what fraction of the population satisfies both the antecedent and the consequent of the rule.
  - E.g., suppose only 0.001 percent of all purchases include milk and screwdrivers. The support for the rule is  $milk \Rightarrow screwdrivers$  is low. Businesses are usually not interested in rules with low support, since they involve few customers.
  - On the other hand, if 50 percent of all purchases involve milk and bread, then support for rules involving those (and no other item) is relatively high and such rules may be worth attention. Exactly what minimum degree of support is considered desirable depends on the application. (*minimum support or minsup*)
- **Confidence** is a measure of how often the consequent is true when the antecedent is true.
  - E.g., the rule  $bread \Rightarrow milk$  has a confidence of 80 percent if 80 percent of the purchases that include bread also include milk. A rule with low confidence is not meaningful. In business applications, rules usually have confidences significantly less than 100 percent, whereas in other domains, such as physics, rules may have high confidences.
  - The confidence of  $bread \Rightarrow milk$  may be very different from the confidence of  $milk \Rightarrow bread$ , although both have the same support.

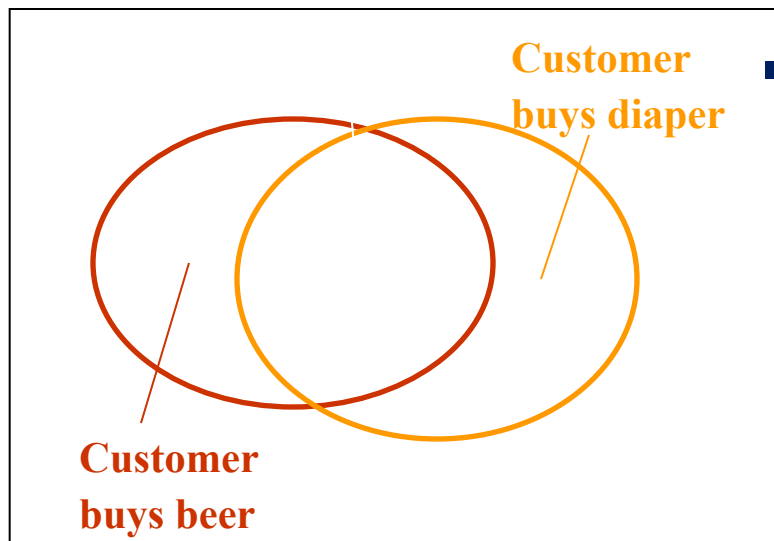


# Finding Association Rules

- We are generally only interested in association rules with reasonably high support (e.g., support of 2% or greater)
- Naïve algorithm (Apriori)
  1. Consider all possible sets of relevant items.
  2. For each set find its support (i.e., count how many transactions purchase all items in the set).
    - **Large itemsets**: sets with sufficiently high support
  3. Use large itemsets to generate association rules.
    1. From itemset  $A$  generate the rule  $A - \{b\} \Rightarrow b$  for each  $b \in A$ .
      - Support of rule = support ( $A$ ).
      - Confidence of rule = support ( $A$ ) / support ( $A - \{b\}$ )
- Finding frequent patterns: FP-tree Algorithm



# Rule Measures: Support and Confidence



Find all the rules  $X \& Y \Rightarrow Z$  with minimum confidence and support

- **support,  $s$ , probability** that a transaction contains  $\{X \cap Y \cap Z\}$
- **confidence,  $c$ , conditional probability** that a transaction having  $\{X \cap Y\}$  also contains  $Z$

| Transaction ID | Items Bought |
|----------------|--------------|
| 2000           | A,B,C        |
| 1000           | A,C          |
| 4000           | A,D          |
| 5000           | B,E,F        |

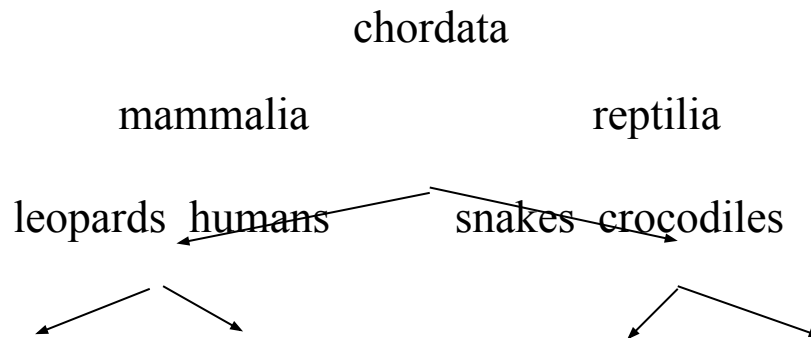
*Let minimum support be 50%, and minimum confidence be 50%, we have*

- $A \Rightarrow C$  (50%, 66.6%)
- $C \Rightarrow A$  (50%, 100%)



# Clustering

- **Clustering:** Intuitively, finding clusters of points in the given data such that similar points lie in the same cluster
- Can be formalized using distance metrics in several ways
  - Group points into  $k$  sets (for a given  $k$ ) such that the average distance of points from the centroid of their assigned group is minimized
    - Centroid: point defined by taking average of coordinates in each dimension.
  - Another metric: minimize average distance between every pair of points in a cluster
- **Hierarchical clustering:** example from biological classification
  - (the word classification here does not mean a prediction mechanism)







# Clustering and Collaborative Filtering

- Goal: predict what movies/books/... a person may be interested in, on the basis of
  - Past preferences of the person
  - Other people with similar past preferences
  - The preferences of such people for a new movie/book/...
- One approach based on repeated clustering
  - Cluster people based on their preferences for movies
  - Then cluster movies on the basis of being liked by the same clusters of people
  - Again cluster people based on their preferences for (the newly created clusters of) movies
  - Repeat above till equilibrium
  - Given new user
    - Find most similar cluster of existing users and
    - Predict movies in movie clusters popular with that user cluster
- Above problem is an instance of **collaborative filtering** , where users collaborate in the task of filtering information to find information of interest



# Other Types of Mining

- **Text mining**: application of data mining to textual documents
  - cluster Web pages to find related pages
  - cluster pages a user has visited to organize their visit history
  - classify Web pages automatically into a Web directory
  - Page Rank (PR) algorithm is used by Google Search Engine
- **Sentiment analysis**
  - E.g., learn to predict if a user review is positive or negative about a product, stock/shares in stock market
- **Information extraction**
  - Create structured information from unstructured textual description or semi-structured data such as tabular displays
- **Entity recognition** and **disambiguation**
  - E.g., given text with name “Michael Jordan” does the name refer to the famous basketball player or the famous ML expert
- **Knowledge graph** (see Section 8.4)
  - Can be constructed by information extraction from different sources, such as



**End of Chapter**



# Best Splits

- Pick best attributes and conditions on which to partition
- The purity of a set  $S$  of training instances can be measured quantitatively in several ways.
  - Notation: number of classes =  $k$ , number of instances =  $|S|$ , fraction of instances in class  $i = p_i$ .
- The **Gini** measure of purity is defined as

[

$$\text{Gini}(S) = 1 - \sum_{i=1}^k p_i^2$$

- When all instances are in a single class, the Gini value is 0
- It reaches its maximum (of  $1 - 1/k$ ) if each class the same number of instances.



## Best Splits (Cont.)

- Another measure of purity is the **entropy** measure, which is defined as

$$\text{entropy}(S) = - \sum_{i=1}^k p_i \log_2 p_i$$

- When a set  $S$  is split into multiple sets  $S_i$ ,  $i=1, 2, \dots, r$ , we can measure the purity of the resultant set of sets as:

$$\text{purity}(S_1, S_2, \dots, S_r) = \sum_{i=1}^r \frac{|S_i|}{|S|} \text{purity}(S_i)$$

- The information gain due to particular split of  $S$  into  $S_i$ ,  $i = 1, 2, \dots, r$   
**Information-gain**  $(S, \{S_1, S_2, \dots, S_r\}) = \text{purity}(S) - \text{purity}(S_1, S_2, \dots, S_r)$



## Best Splits (Cont.)

- Measure of “cost” of a split:

$$\text{Information-content } (S, \{S_1, S_2, \dots, S_r\}) = - \sum_{i=1}^r \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

- Information-gain ratio** = 
$$\frac{\text{Information-gain } (S, \{S_1, S_2, \dots, S_r\})}{\text{Information-content } (S, \{S_1, S_2, \dots, S_r\})}$$
- The best split is the one that gives the maximum information gain ratio



# Finding Best Splits

- Categorical attributes (with no meaningful order):
  - Multi-way split, one child for each value
  - Binary split: try all possible breakup of values into two sets, and pick the best
- Continuous-valued attributes (can be sorted in a meaningful order)
  - Binary split:
    - Sort values, try each as a split point
      - E.g., if values are 1, 10, 15, 25, split at  $\leq 1, \leq 10, \leq 15$
    - Pick the value that gives best split
  - Multi-way split:
    - A series of binary splits on the same attribute has roughly equivalent effect



# Decision-Tree Construction Algorithm

**Procedure** *GrowTree* ( $S$ )

    Partition ( $S$ );

**Procedure** Partition ( $S$ )

**if** ( $\text{purity}(S) > \delta_p$  or  $|S| < \delta_s$ ) **then**

**return**;

**for each** attribute  $A$

        evaluate splits on attribute  $A$ ;

    Use best split found (across all attributes) to partition

$S$  into  $S_1, S_2, \dots, S_r$ ,

**for**  $i = 1, 2, \dots, r$

        Partition ( $S_i$ );





# Finding Association Rules

- We are generally only interested in association rules with reasonably high support (e.g., support of 2% or greater)
- Naïve algorithm
  1. Consider all possible sets of relevant items.
  2. For each set find its support (i.e., count how many transactions purchase all items in the set).
    - **Large itemsets**: sets with sufficiently high support
  3. Use large itemsets to generate association rules.
    - From itemset  $A$  generate the rule  $A - \{b\} \Rightarrow b$  for each  $b \in A$ .
      - Support of rule = support ( $A$ ).
      - Confidence of rule = support ( $A$ ) / support ( $A - \{b\}$ )



# Finding Support

- Determine support of itemsets via a single pass on set of transactions
  - Large itemsets: sets with a high count at the end of the pass
- If memory not enough to hold all counts for all itemsets use multiple passes, considering only some itemsets in each pass.
- Optimization: Once an itemset is eliminated because its count (support) is too small none of its supersets needs to be considered.
- The **a priori** technique to find large itemsets:
  - Pass 1: count support of all sets with just 1 item. Eliminate those items with low support
  - Pass  $i$ : **candidates**: every set of  $i$  items such that all its  $i-1$  item subsets are large
    - Count support of all candidates
    - Stop if there are no candidates



# Other Types of Associations

- Basic association rules have several limitations
- Deviations from the expected probability are more interesting
  - E.g., if many people purchase bread, and many people purchase cereal, quite a few would be expected to purchase both
  - We are interested in **positive** as well as **negative correlations** between sets of items
    - Positive correlation: co-occurrence is higher than predicted
    - Negative correlation: co-occurrence is lower than predicted
- Sequence associations / correlations
  - E.g., whenever bonds go up, stock prices go down in 2 days
- Deviations from temporal patterns
  - E.g., deviation from a steady growth
  - E.g., sales of winter wear go down in summer
    - Not surprising, part of a known pattern.
    - Look for deviation from value predicted using past patterns



# Hierarchical Clustering

- Agglomerative clustering algorithms
  - Build small clusters, then cluster small clusters into bigger clusters, and so on
- Divisive clustering algorithms
  - Start with all items in a single cluster, repeatedly refine (break) clusters into smaller ones



# Clustering Algorithms

- Clustering algorithms have been designed to handle very large datasets
- E.g., the **Birch algorithm**
  - Main idea: use an in-memory R-tree to store points that are being clustered
  - Insert points one at a time into the R-tree, merging a new point with an existing cluster if it is less than some  $\delta$  distance away
  - If there are more leaf nodes than fit in memory, merge existing clusters that are close to each other
  - At the end of first pass we get a large number of clusters at the leaves of the R-tree
    - Merge clusters to reduce the number of clusters