Data Warehousing and Mining

ECE 356
University of Waterloo
Prof. Wojciech Golab

Acknowledgment: slides derived from materials provided by Silberschatz, Korth, and Sudarshan, copyright 2010
(source: www.db-book.com)

Learning Outcomes

- Develop a conceptual understanding of:
  - decision support systems
  - data warehousing
- Develop a working knowledge of fundamental data mining techniques:
  - classification
  - association rule mining
  - clustering

- Textbook sections (6th ed.): chapter 20

DECISION SUPPORT

Decision Support Systems

- **Decision-support systems** are used to make business decisions, often based on data collected using OLTP systems.
- Examples of business decisions:
  - What items to stock?
  - What insurance premium to charge?
  - To whom to send advertisements?
- Examples of data used for making decisions:
  - Retail sales transaction details
  - Customer profiles (income, age, gender, "friends with", etc.)

Decision Support Systems

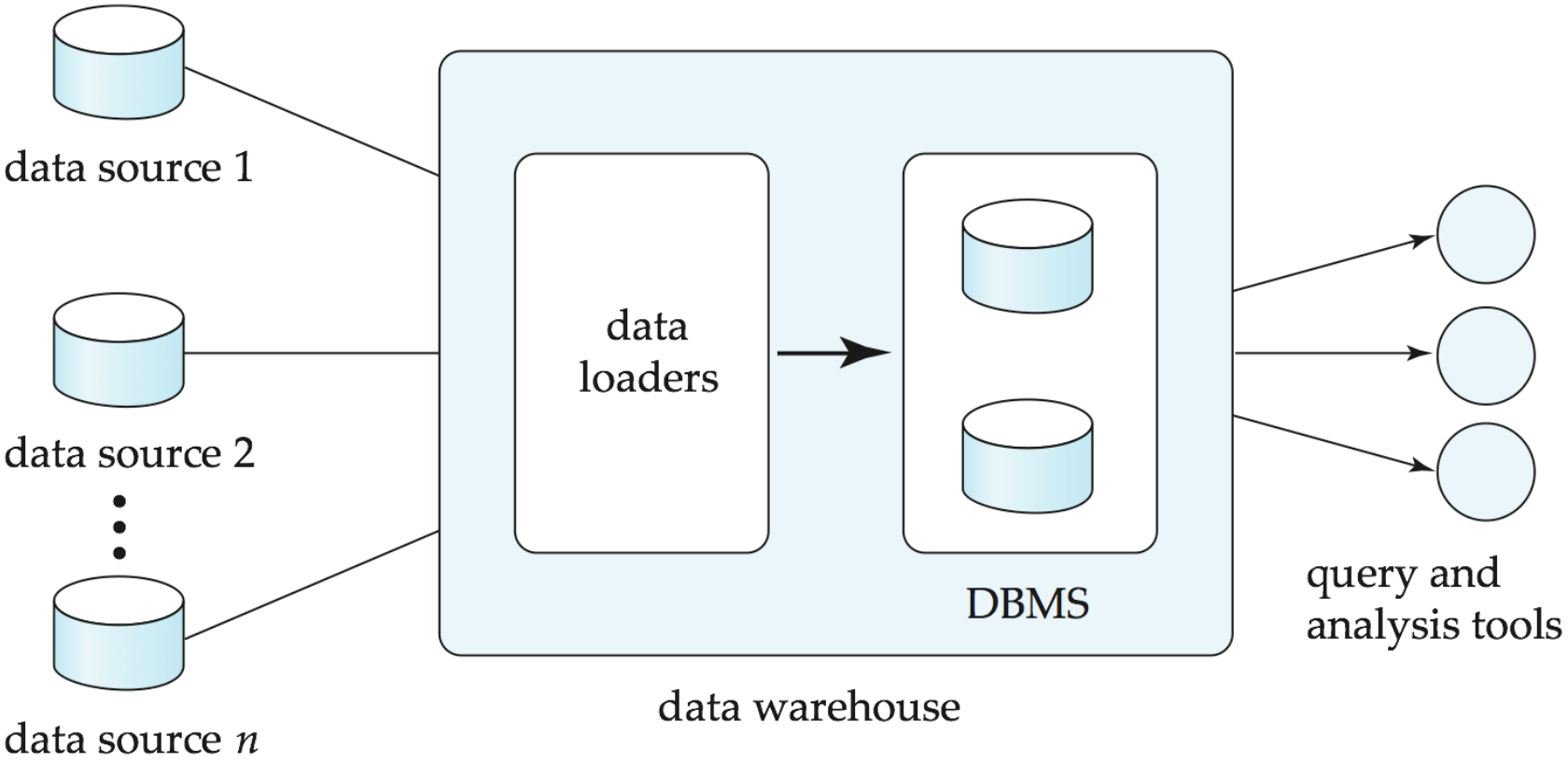
- **Data analysis** tasks are simplified by specialized tools and SQL extensions. Example tasks:
  - for each product category and each region, what were the total sales in the last quarter and how do they compare with the same quarter last year
  - as above, for each product category and each customer category
- **Statistical analysis** packages (e.g., S++) can be interfaced with databases. (Outside the scope of the lecture.)
- **Data mining** seeks to discover knowledge automatically (or semi-automatically) in the form of statistical rules and patterns from large databases.

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** archives information gathered from multiple sources, and stores it under a unified schema, at a single site.
  - important for large businesses that generate data from multiple divisions, possibly at multiple sites (data may also be purchased externally)
  - simplifies querying and permits study of historical trends
  - shifts decision support query load away from transaction processing systems and into other tools

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.
  - Keeping warehouse exactly synchronized with data sources (e.g., using two-phase commit) is too expensive. It is often OK to have slightly out-of-date data at warehouse.
- *What schema to use*
  - schema integration

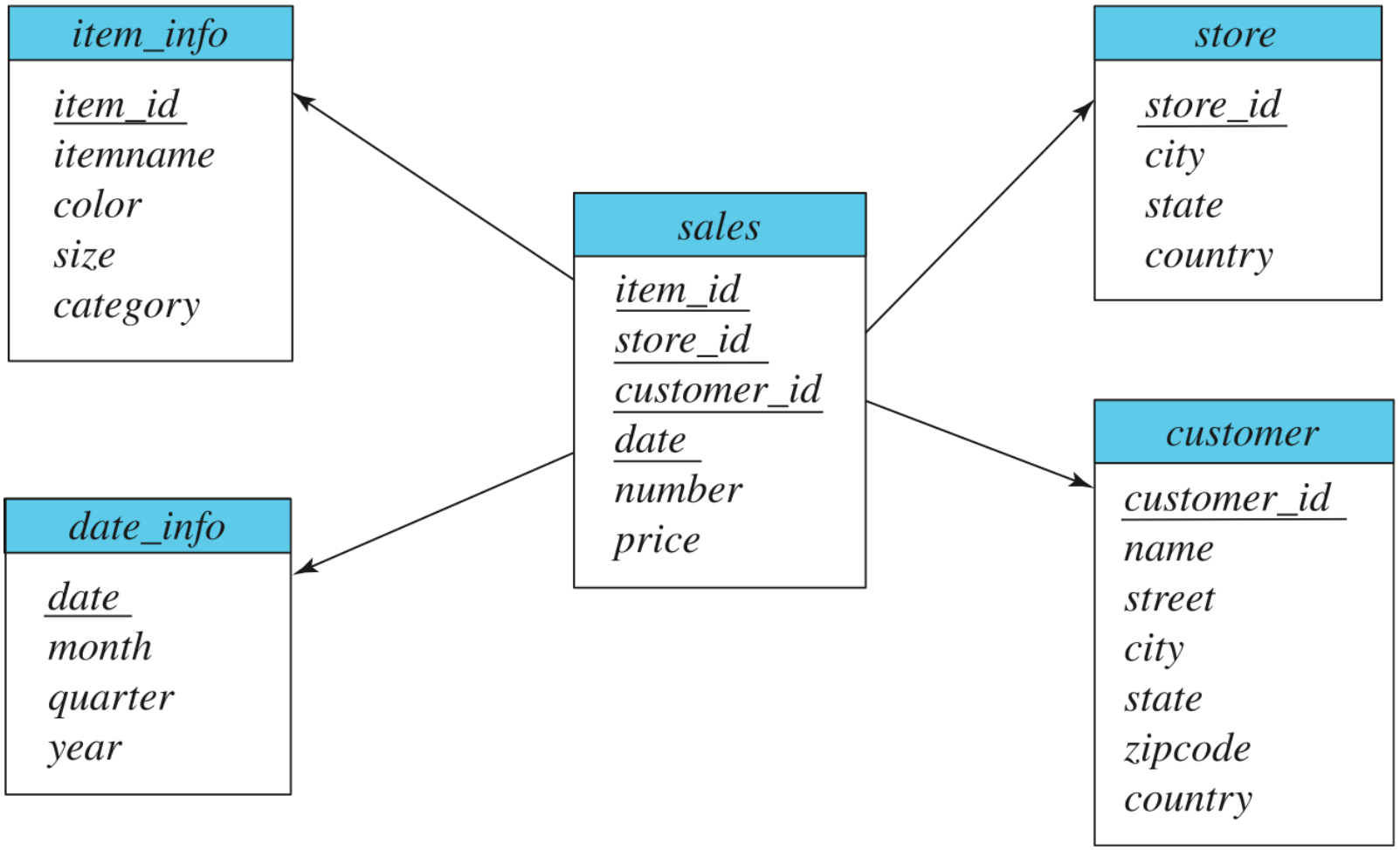
More Design Issues

- *Data cleansing*
  - e.g., **correct** mistakes in addresses (misspellings, zip code errors)
  - or **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
- *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

Warehouse Schemas

- Warehouses generally organize data into **fact tables** and **dimension tables**.
- Fact tables describe specific events (e.g., transactions), and contain mostly small numeric values as well as foreign keys pointing to rows of dimension tables. Fact tables can be very large.
- Dimension tables store descriptive data, for example referring to a time, a place, a product, or a person. They tend to record a larger number of attributes, including both numeric and text. Dimension tables tend to contain fewer rows than fact tables.
- Resultant schema is called a **star schema**. More elaborate schema structures are possible:
  - **Snowflake schema**: multiple levels of dimension tables
  - **Constellation**: multiple fact tables

Example of Star Schema

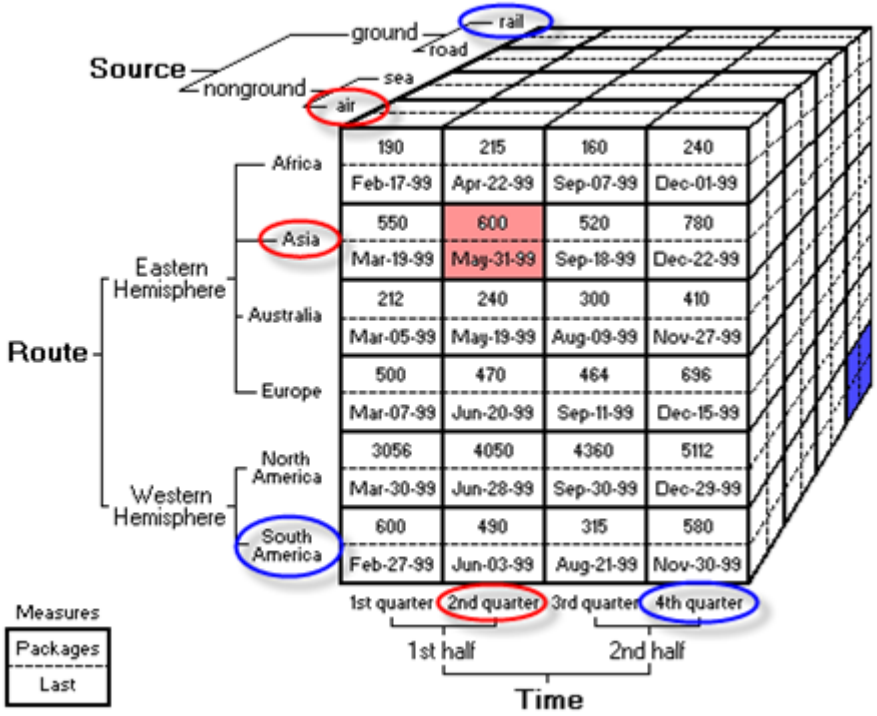


Data Cubes

- A star schema represents multidimensional data, which is difficult to visualize.
- OLAP (on-line analytical processing) systems enable data summarization and interactive exploration of a multidimensional data set using the **data cube** abstraction.
- Each cell of the data cube holds a data item corresponding to an intersection of dimensions (e.g., transaction for some product in some city at some time).
- A **slicer** is a dimension that is held constant so that the cube can be collapsed onto fewer dimensions.

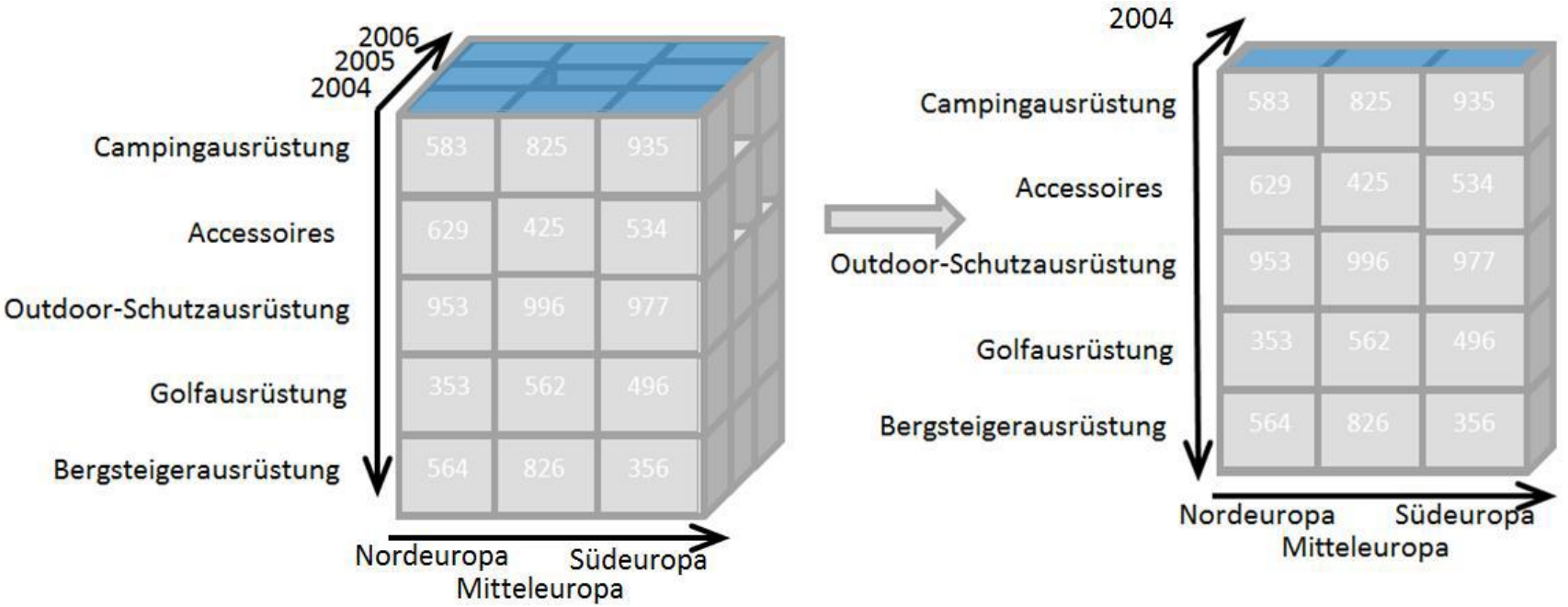
Data cubes

Example 3D OLAP cube over shipping data.



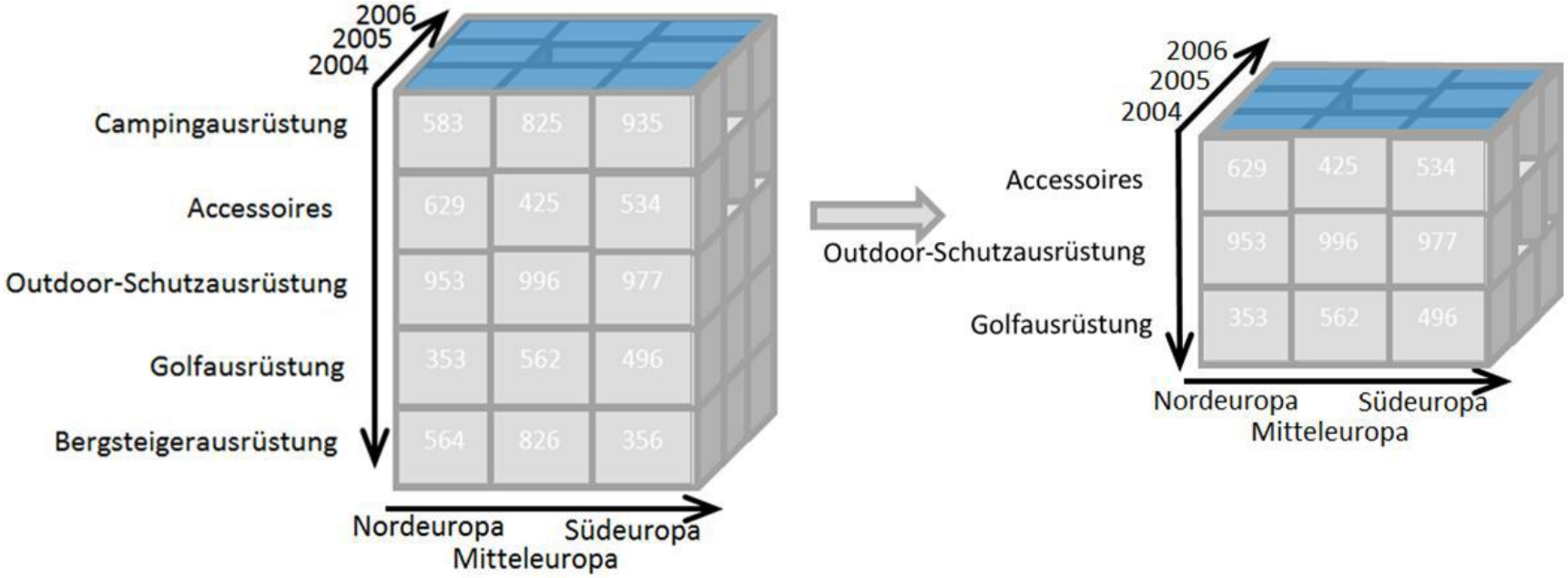
Source: http://blog.xlcubed.com/wp-content/uploads/2010/06/cube.png

Data cubes: slicing



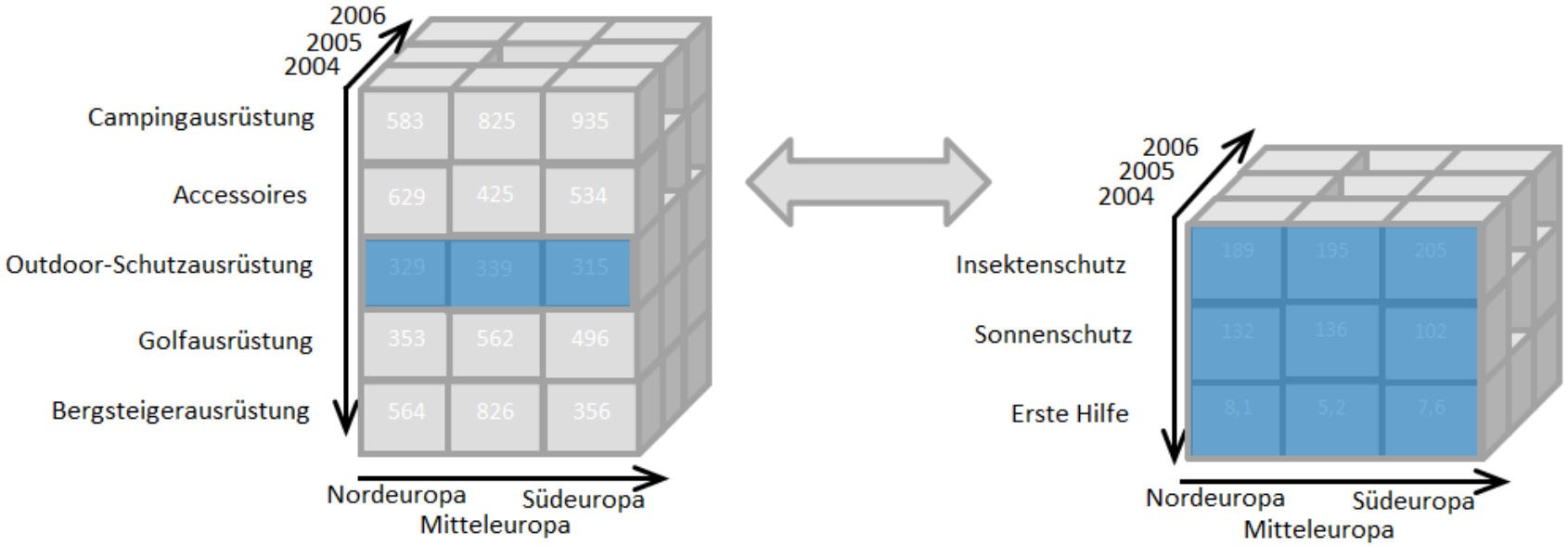
Source:
http://en.wikipedia.org/wiki/OLAP_cube#mediaviewer/File:OLAP_slicing.png

Data cubes: dicing



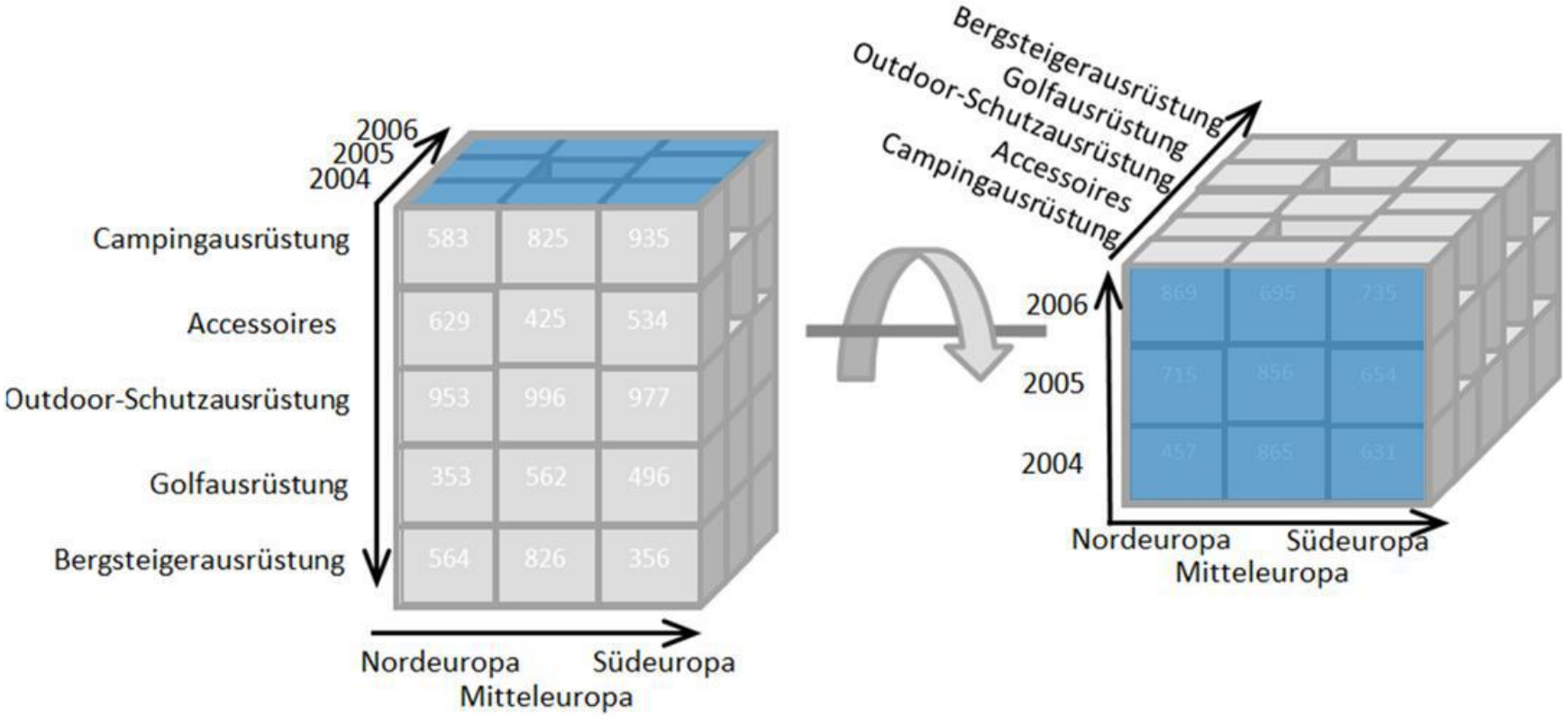
Source:
http://en.wikipedia.org/wiki/OLAP_cube#mediaviewer/File:OLAP_dicing.png

Data cubes: drill-up/drill-down



Source:
http://en.wikipedia.org/wiki/OLAP_cube#mediaviewer/File:OLAP_drill_up%26down.png

Data cubes: pivoting



Source:
http://en.wikipedia.org/wiki/OLAP_cube#mediaviewer/File:OLAP_pivoting.png

DATA MINING

Data Mining

- Data mining is the process of **semi-automatically** analyzing large databases to find **useful patterns**.
- Data mining is often used for **prediction**:
  - e.g., predict if a credit card applicant poses a good credit risk, based on some attributes (income, job type, age, ..) and past history
  - e.g., predict if a pattern of phone calling card usage is likely to be fraudulent
- Examples of prediction mechanisms:
  - **Classification:** given a new item whose class is unknown, predict to which class it belongs.
    - e.g., classify a blog post as either positive or negative
  - **Regression:** given a set of mappings for an unknown function, predict the function result for a new parameter value.
    - e.g., given the outdoor temperature measurements for the last week, predict tomorrow's outdoor temperature

Data Mining (Cont.)

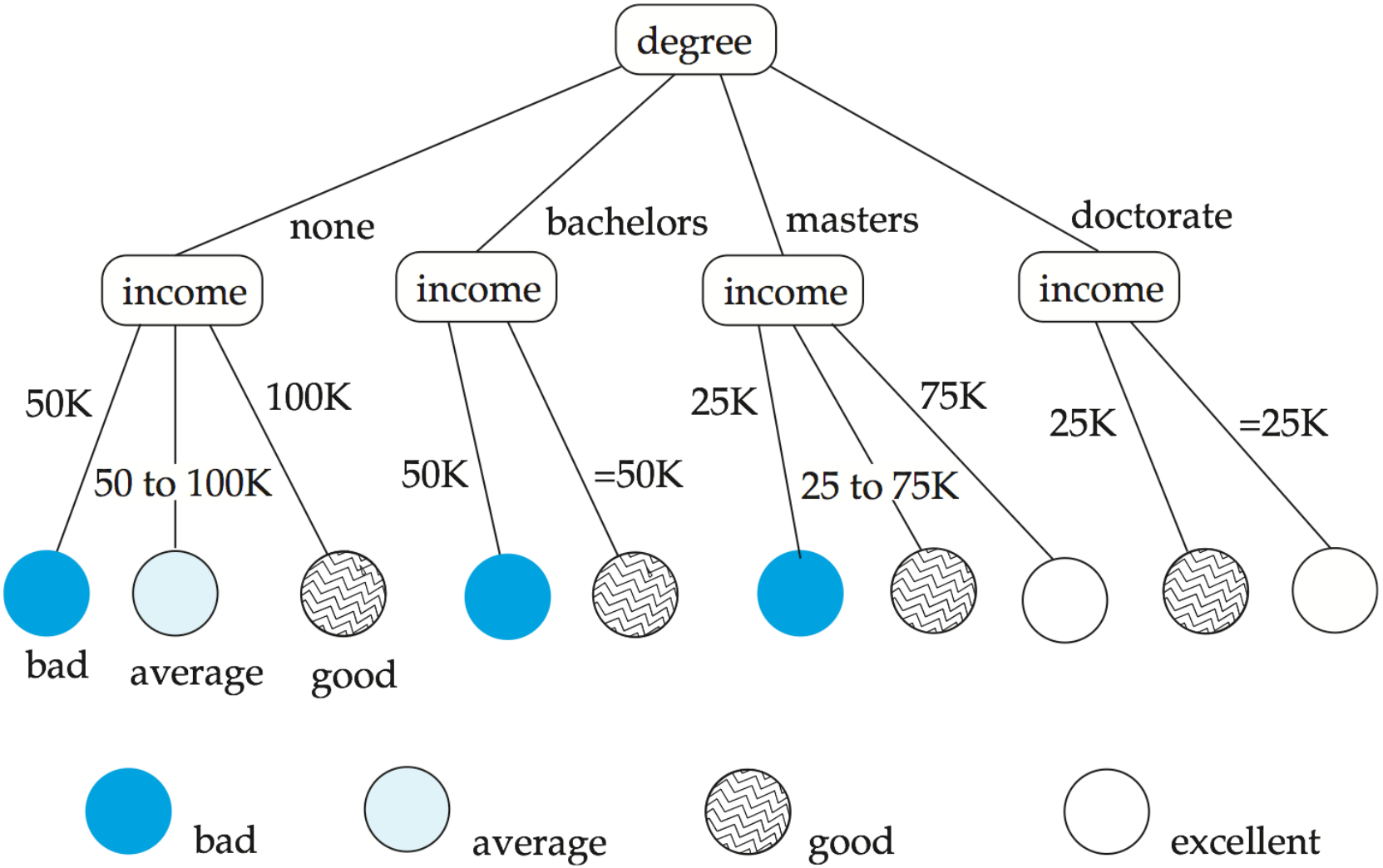
- Other applications of data mining aim to **identify descriptive patterns** in existing data.
- Examples of descriptive patterns:
  - **Associations:** "if-then" patterns.
    - e.g., if a customer C purchases a book B, then they are likely to enjoy book B' because other customers "similar" to C have purchased both B and B' (… so recommend B' to C)
    - e.g., if exposure to a chemical C in a population tends to co-occur with cancer, then suspect that C may be a carcinogen (… so impose a ban on C)
  - **Clustering:** discover groups of similar objects
    - e.g., mobile network users are clustered in certain areas, suggesting where cellular towers should be placed
    - e.g., typhoid cases were clustered in an area surrounding a particular well, suggesting that the well was contaminated

DATA MINING: CLASSIFICATION

Classification Rules

- Classification rules help assign new objects to classes.
  - e.g., given a new credit card application, predict the credit risk of the applicant
- Classification rules for the 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 represented compactly as a decision tree.

Decision Tree



Construction of Decision Trees

- **Training set**: a sample of instances (i.e., inputs) for which the classification is already known.
- The decision tree is generated from the training set using a **greedy** top-down approach:
  - Each internal node of the tree partitions the data into groups based on a **partitioning attribute**, and a **partitioning condition** for the node.
  - At each **leaf** node, either all (or most) of the items at the node belong to the same class, or else all attributes have been considered and no further partitioning is possible.

Best Splits

- A traversal of the decision tree begins with “impure” data (instances from many classes) at the root and terminates with “pure” data (instances from one class only) at the leaf level.
- The main **goal in building a decision tree** is to pick the best attributes and conditions on which to partition at each level so as the reduce the “impurity”.
- Several quantitative measures of **impurity** have been proposed over a set *S* of **training instances**. (See next slide.)
- Notation:

  k = number of classes

  $|S|$ = number of instances

  p_i = fraction of instances in class i.

Impurity Measures: Gini

- The **Gini** measure of impurity is defined as

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

- If all instances are in a single class (i.e., maximum purity), the Gini value is 0.
- If each class has the same number of instances (i.e., minimum purity) the value is $1 - 1/k$.

Impurity Measures: Entropy

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

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

- If all instances are in a single class (i.e., maximum purity), the entropy value is 0.
  - note: $p_i \log_2 p_i$ is defined as 0 for $p_i = 0$
- If each class has the same number of instances (i.e., minimum purity) the value is $-\log_2 1/k$.

Information Gain

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

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

- The **information gain** due to a particular split of S into S_i, i = 1, 2, …., r is defined as follows:

$$\text{Information-gain}(S, \{S_1, S_2, \ldots, S_r\}) = \text{Impurity}(S) - \text{Impurity}(S_1, S_2, \ldots S_r)$$

- A good split always achieves a positive information gain!

Information Gain (cont.)

- Measure of "cost" of a split:

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

- Measure of "goodness" or "bang-for-buck" of a split:

$$\textbf{Information gain ratio} = \frac{\text{Information-gain } (S, \{S_1, S_2, \ldots\ldots, S_r\})}{\text{Information-content } (S, \{S_1, S_2, \ldots\ldots, S_r\})}$$

- The best split (i.e., one that tends to yield the simplest and most meaningful decision tree) is the one that produces the **maximum information gain ratio**.

Finding Best Splits

- Categorical attributes (with no meaningful order):
  - binary split – try all possible ways to partition the values into two disjoint sets, and pick the best
  - multi-way split – one child for each value
- Continuous-valued attributes (with meaningful order):
  - binary split – sort values, try each as a split point
    - e.g., if values are 1, 10, 15, 25, split at ≤ 1, ≤ 10, ≤ 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

Procedure *GrowTree* (*S*)
    Partition (*S*);

Procedure *Partition* (*S*)
    if (Impurity (*S*) < δ_p or |*S*| < δ_s) **then**
        return;
    for each attribute *A*
        evaluate splits on attribute *A*;
    use best split found (across all attributes) to
    partition *S* into $S_1, S_2, \ldots, S_r$;
    for *i* = 1, 2, ….., *r*
        Partition (S_i);

Note 1: δ_p and δ_s are user-defined thresholds.

Note 2: In addition to computing the structure of the decision tree, we must assign a decision for each leaf node, for example based on the most common class in the corresponding set S_i of training instances.

Other Types of Classifiers

- Neural net classifiers are studied in artificial intelligence and are not covered in this lecture.
- Bayesian classifiers use **Bayes theorem**, which states:

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

where

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

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

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

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

Naïve Bayesian Classifiers

- Bayesian classifiers require
  - computation of $p(d \mid c_j)$
  - pre-computation 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

  $p(d \mid c_j) = p(d_1 \mid c_j) \cdot p(d_2 \mid c_j) \cdot \ldots \cdot p(d_n \mid c_j)$

  - Each of the $p(d_i \mid c_j)$ can be estimated from a histogram on d_i values for each class c_j.
  - Histograms are computed from training instances.

Validating a Classifier

- The quality of a classifier can be quantified along several dimensions.
- For a Boolean classifier, outcomes fall into four categories:
  - **true positive (TP):** prediction was positive, instance is positive
  - **false positive (FP):** prediction was positive, instance is negative
  - **true negative (TN):** prediction was negative, instance is negative
  - **false negative (FN):** prediction was negative, instance is positive
- The quality of a classifier can be described in several ways:
  - **accuracy:** fraction of instances where the classifier is correct
    (#TP + #TN)/(#TP + #FP + #TN + #FN)
  - **recall:** fraction of positive instances correctly classified
    (#TP)/(#TP + #FN) [a.k.a. true positive rate]
  - **precision:** fraction of correct positive predictions
    (#TP)/(#TP + #FP)
  - **specificity:** fraction of negative instances correctly classified
    (#TN)/(#FP + #TN) [a.k.a. true negative rate]

Regression

- Regression deals with the prediction of a value, rather than a class. Given values for a set of variables, X_1, X_2, …, X_n, we wish to predict the value of a variable Y.
- One way is to infer coefficients a_0, a_1, a_1, …, a_n such that
  $$Y = a_0 + a_1 \cdot X_1 + a_2 \cdot X_2 + \ldots + a_n \cdot X_n$$
- 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**.
- Regression aims to find coefficients that give the best possible fit (e.g., minimizes sum of squared residuals).
- The fit may only be approximate because of noise in the data, or because the relationship does not follow exactly the type of curve being fitted (e.g., polynomial).

DATA MINING:
ASSOCIATION RULE MINING

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*.
- **Association rules:**

  $bread \Rightarrow milk$

  $DB\text{-}Concepts, OS\text{-}Concepts \Rightarrow Networks$

  - Left side is the **antecedent**, right side is the **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 that only 0.001% of all purchases include both milk and screwdrivers. Then the degree of support for the rule *milk* $\Rightarrow$ *screwdrivers* is low.
- **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% if 80% of the purchases that include bread also include milk.

Finding Association Rules

- We are generally only interested in association rules with reasonably high (e.g., few %) support.
- Naïve algorithm:
  1. Consider all possible sets of relevant items.
  2. Compute the support for each set and identify sets with sufficiently high support (e.g., based on a threshold).
  3. Select **large itemsets** with sufficiently high support.
  4. Use these large itemsets to generate association rules. From itemset A and each $b \in A$ generate the rule $A - \{b\} \Rightarrow b$
     - support of rule = support(A)
     - confidence of rule = support (A) / support ($A - \{b\}$)
- **Question:** How do we find large itemsets? (See next slide.)

Apriori Algorithm

- Algorithm computes large itemsets given a set T of transactions over a set of items, and a support threshold ε.
- Apriori (T, ε)

1. L_1 := set of 1-itemsets with support at least ε in T
2. $k := 2$
3. **while** L_{k-1} is not empty
4. C_k := set of k-itemsets such that every subset of size k-1 is an element of L_{k-1}
5. L_k := subset of k-itemsets in C_k with support at least ε in T
6. $k := k + 1$
7. **return** the union of $L_1, L_2, \ldots, L_{k-2}$

Apriori: Example

$\varepsilon = 3/7$

Transactions
{1,2,3,4}
{1,2,4}
{1,2}
{2,3,4}
{2,3}
{3,4}
{2,4}

L_1

Itemset	Support
{1}	3/7
{2}	6/7
{3}	4/7
{4}	5/7

C_2

Itemset	Support
{1,2}	3/7
{1,3}	1/7
{1,4}	2/7
{2,3}	3/7
{2,4}	4/7
{3,4}	3/7

L_2

C_3

Itemset	Support
{2,3,4}	2/7

(L_3 empty)

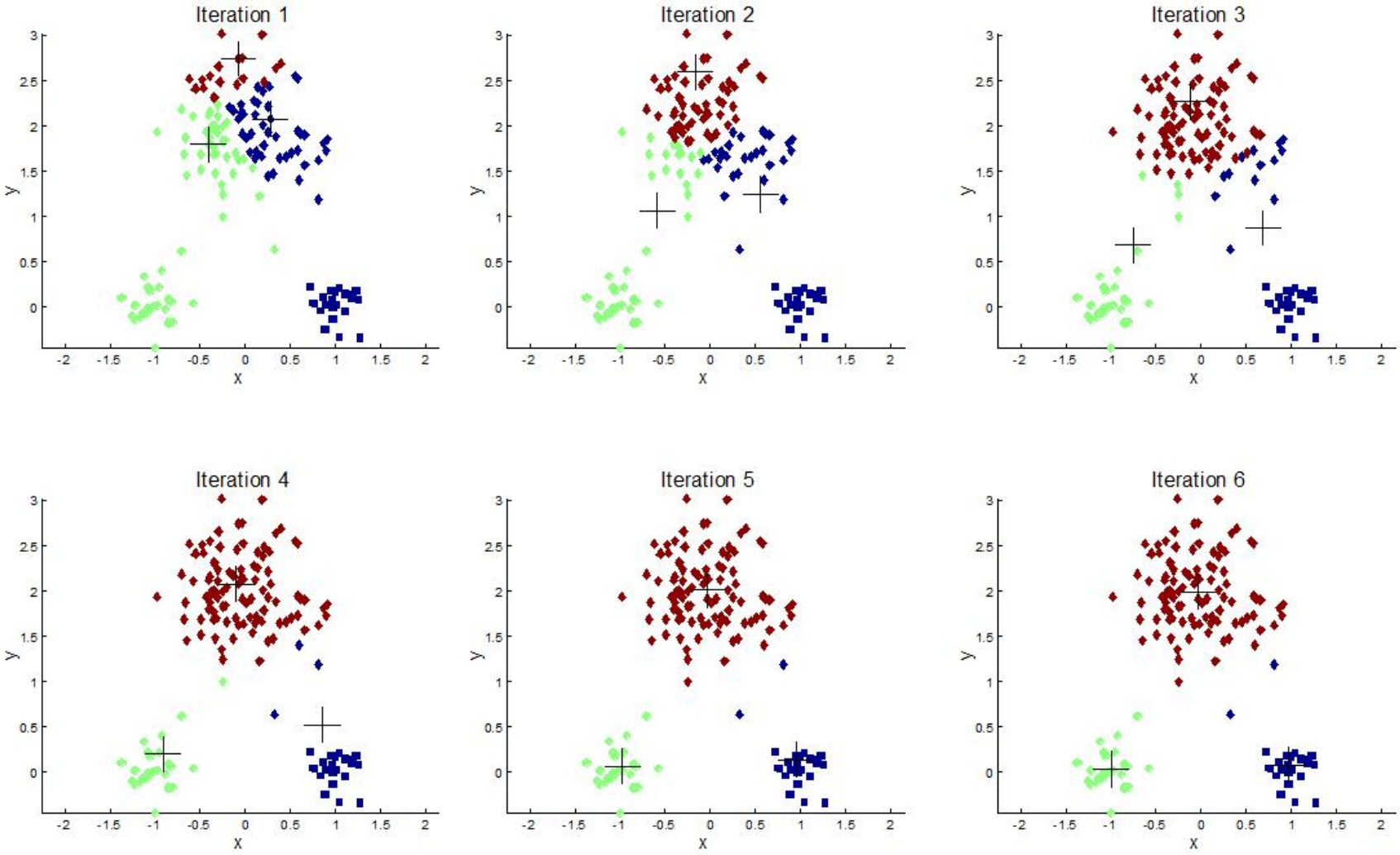
(based on https://en.wikipedia.org/wiki/Apriori_algorithm)

DATA MINING: CLUSTERING

Clustering

- Clustering: intuitively, finding clusters of points in the given data such that similar points lie in the same cluster.
- The problem can be formalized using distance metrics.
- Example: ***k*-means**
  - Group points into *k* sets (for a given *k*) such that the average distance (e.g., Euclidean distance) of points from the centroid of their assigned set is minimized.
- Clustering has been studied extensively in statistics, but on small data sets. Data mining systems instead aim at clustering techniques that can handle very large data sets.

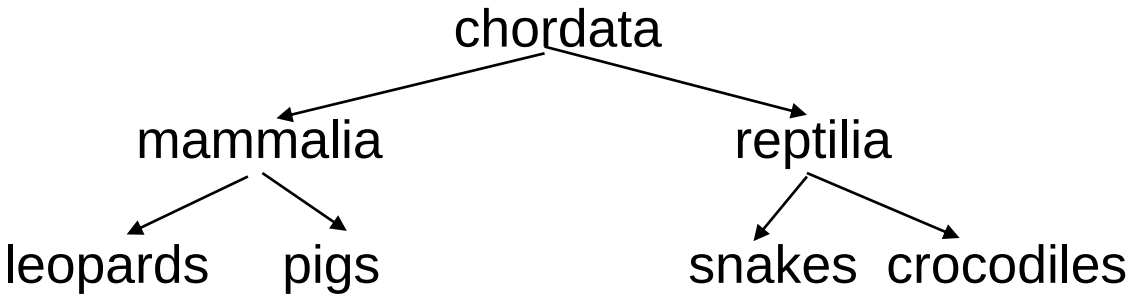
Example of *k*-means clustering



Source: http://apandre.files.wordpress.com/2011/08/kmeansclustering.jpg

Hierarchical Clustering

- Example from biology:



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

Collaborative Filtering

- Goal: predict what movies/books/… a person may be interested in, on the basis of data such as
  - past preferences of the person
  - other people with similar past preferences
- One approach based on repeated clustering:
  1. cluster people on the basis of preferences for movies
  2. then cluster movies on the basis of being liked by the same clusters of people
  3. again cluster people based on their preferences for (the newly created clusters of) movies
  4. repeat until convergence
- Above problem is an instance of **collaborative filtering**, where users collaborate in the task of filtering information to find information of interest. Solution: alternating least squares (ALS) technique.

Other Types of Mining

- **Text mining**: application of data mining to textual documents.
  - e.g., cluster Web pages to find related pages
  - e.g., cluster pages a user has visited to organize their visit history
  - e.g., classify Web pages automatically into a Web directory
- **Data visualization** systems help users examine large volumes of data and detect patterns visually.
  - large amounts of information encoded on a single screen
  - parallel processing power of human brain used to detect patterns visually