Data Warehousing and OLAP

Characteristics of Data warehouse:

1. Subject-oriented –
A data warehouse is always a subject oriented as it delivers information about a theme instead of organization's current operations. It can be achieved on specific theme. That means the data warehousing process is proposed to handle with a specific theme which is more defined. These themes can be sales, distributions, marketing etc.

A data warehouse never put emphasis only current operations. Instead, it focuses on demonstrating and analysis of data to make various decision. It also delivers an easy and precise demonstration around particular theme by eliminating data which is not required to make the decisions.

2. Integrated –
It is somewhere same as subject orientation which is made in a reliable format. Integration means founding a shared entity to scale the all similar data from the different databases. The data also required to be resided into various data warehouse in shared and generally granted manner.

A data warehouse is built by integrating data from various sources of data such that a mainframe and a relational database. In addition, it must have reliable naming conventions, format and codes. Integration of data warehouse benefits in effective analysis of data. Reliability in naming conventions, column scaling, encoding structure etc. should be confirmed. Integration of data warehouse handles various subject related warehouse.

3. Time-Variant –
In this data is maintained via different intervals of time such as weekly, monthly, or annually etc. It founds various time limit which are structured between the large datasets and are held in online transaction process (OLTP). The time limits for data warehouse is wide-ranged than that of operational systems. The data resided in data warehouse is predictable with a specific interval of time and delivers information from the historical perspective. It comprises elements of time explicitly or implicitly. Another feature of time-variance is that once data is stored in the data warehouse then it cannot be modified, alter, or updated.

4. Non-Volatile –
As the name defines the data resided in data warehouse is permanent. It also means that data is not erased or deleted when new data is inserted. It includes the mammoth quantity of data that is inserted into modification between the selected quantity on logical business. It evaluates the analysis within the technologies of warehouse.

In this, data is read-only and refreshed at particular intervals. This is beneficial in analysing historical data and in comprehension the functionality. It does not need transaction process, recapture and concurrency control mechanism. Functionalities such as delete, update, and insert that are done in an operational application are lost in data warehouse environment.
Two types of data operations done in the data warehouse are:

- Data Loading
- Data Access

OLAP vs OLTP

Category	OLTP	OLAP
Users and system orientation	It is customer-oriented and is used for transaction and query processing by clerks, clients, and information technology professionals.	It is market-oriented and is used for data analysis by knowledge workers, including managers, executives, and analysts.
Data contents	It manages current data that, typically, are too detailed to be easily used for decision making.	It manages large amounts of historic data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity.
Database design	It usually adopts an entity-relationship (ER) data model and an application-oriented database design.	It typically adopts either a *star* or a *snowflake* model and a subject-oriented database design.
View	It focuses mainly on the current data within an enterprise or department, without referring to historic data or data in different organizations.	An OLAP system often spans multiple versions of a database schema, due to the evolutionary process of an organization. OLAP systems also deal with informa- tion that originates from different organizations, integrating information from many data stores.
Access patterns	The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery mech-anisms.	The access to OLAP systems are mostly read-only operations (because most data warehouses store historic rather than up-to-date information), although many could be complex queries.

Feature	*OLTP*	*OLAP*
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long-term informational requirements decision support
DB design	ER-based, application-oriented	star/snowflake, subject-oriented
Data	current, guaranteed up-to-date	historic, accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
Number of records accessed	tens	millions
Number of users	thousands	hundreds
DB size	GB to high-order GB	≥ TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

ROLAP vs MOLAP vs HOLAP

Category	ROLAP	MOLAP	HOLAP
Definition	A ROLAP technique for implementing a multiple dimensional view consists of intermediate servers that stand in between a relational back-end server and client front-end tools, thereby using a relational or extended-relational DBMS to store and manage warehouse data, and OLAP middleware to support missing pieces.	A MOLAP implementation technique consists of servers, which support multidimensional views of data through array-based multidimensional storage engines that map multidimensional views directly to data cube array structures.	A HOLAP implementation approach combines ROLAP and MOLAP technology, which means that large volumes of detailed data and some very low level aggregations can be stored in a relational database, while some high level aggregations are kept in a separate MOLAP store.
Generation of Data warehouse	Using a ROLAP server, the generation of a data warehouse can be implemented by a relational or extended-relational DBMS using summary fact tables. The fact tables can store aggregated data and the data at the abstraction levels indicated by the join keys in the schema for the given data cube.	In generating a data warehouse, the MOLAP technique uses multidimensional array structures to store data and multiway array aggregation to compute the data cubes.	The HOLAP technique typically uses a relational database to store the data and some low level aggregations, and then uses a MOLAP to store higher-level aggregations.
Roll-Up	To roll-up on a dimension using the summary fact table, we look for the record in the table that contains a generalization on the desired dimension.	To perform a roll-up in a data cube, simply climb up the concept hierarchy for the desired dimension.	The roll-up using the HOLAP technique will be similar to either ROLAP or MOLAP, depending on the techniques used in the implementation of the corresponding dimensions.

Category	ROLAP	MOLAP	HOLAP
Drill Down	To drill-down on a dimension using the summary fact table, we look for the record in the table that contains a generalization on the desired dimension.	To perform a drill-down in a data cube, simply step down the concept hierarchy for the desired dimension.	The drill-down using the HOLAP technique is similar either to ROLAP or MOLAP depending on the techniques used in the implementation of the corresponding dimensions.
Incremental Updating	To perform incremental updating, check whether the corresponding tuple is in the summary fact table. If not, insert it into the summary table and propagate the result up. Otherwise, update the value and propagate the result up.	To perform incremental updating, check whether the corresponding cell is in the MOLAP cuboid. If not, insert it into the cuboid and propagate the result up. Otherwise, update the value and propagate the result up.	similar either to ROLAP or MOLAP depending on the techniques used in the implementation of the corresponding dimensions.
Which is preferred	HOLAP is often preferred since it integrates the strength of both ROLAP and MOLAP methods and avoids their shortcomings—if the cube is quite dense, MOLAP is often preferred. Also, if the data are sparse and the dimensionality is high, there will be too many cells (due to exponential growth) and, in this case, it is often desirable to compute iceberg cubes instead of materializing the complete cubes.		

Typical OLAP Operations

Operation Type	Operation Definition
Roll Up (Drill Up)	The roll-up operation (also called the drill-up operation by some vendors) performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction. When roll-up is performed by dimension reduction, one or more dimensions are removed from the given cube.
Roll Down (Drill Down)	Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions. As a drill-down adds more detail to the given data, it can also be performed by adding new dimensions to a cube.
Slice and Dice	The slice operation performs a selection on one dimension of the given cube, resulting in a sub cube.
Pivot (Rotate)	Pivot (also called rotate) is a visualization operation that rotates the data axes in view to provide an alternative data presentation. Examples include rotating the axes in a 3-D cube, or transforming a 3-D cube into a series of 2-D planes.

Type of Schemas and DW vs DM

Schema Name	Definition
Star Schema	The most common modeling paradigm is the star schema, in which the data warehouse contains (1) a large central table (fact table) containing the bulk of the data, with no redundancy, and (2) a set of smaller attendant tables (dimension tables), one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.
Snowflakes Schema	The snowflake schema is a variant of the star schema model, where some dimension tables are *normalized*, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake. The major difference between the snowflake and star schema models is that the dimension tables of the snowflake model may be kept in normalized form to reduce redundancies. Such a table is easy to maintain and saves storage space. However, this space savings is negligible in comparison to the typical magnitude of the fact table. Fur- thermore, the snowflake structure can reduce the effectiveness of browsing, since more joins will be needed to execute a query. Consequently, the system performance may be adversely impacted. Hence, although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.
Fact Constellations Schema	Sophisticated applications may require multiple fact tables to share dimension tables. This kind of schema can be viewed as a collection of stars, and hence is called a galaxy schema or a fact constellation.

In data warehousing, there is a distinction between a data warehouse and a data mart. A data warehouse collects information about subjects that span the entire organization, such as customers, items, sales, assets, and personnel, and thus its scope is enterprise-wide. For data warehouses, the fact constellation schema is commonly used, since it can model multiple, interrelated subjects. A data mart, on the other hand, is a department subset of the data warehouse that focuses on selected subjects, and thus its scope is department-wide. For data marts, the star or snowflake schema is commonly used, since both are geared toward modeling single subjects, although the star schema is more popular and efficient.

General Definitions

Name	Definition
Data Cube	A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts.
Dimensions	In general terms, dimensions are the perspectives or entities with respect to which an organization wants to keep records. Each dimension may have a table associated with it, called a dimen- sion table, which further describes the dimension.
Facts	A multidimensional data model is typically organized around a central theme, such as *sales*. This theme is represented by a fact table. Facts are numeric measures. Think of them as the quantities by which we want to analyze relationships between dimensions. The fact table contains the names of the *facts*, or measures, as well as keys to each of the related dimension tables.
Hierarchy	A dimensional hierarchy denotes how data is organized at various levels of aggregation. An analyst uses a dimensional hierarchy to identify various trends at one level, drill down to lower levels to detect causes for these trends, and roll up to higher levels to see the effects the trends have on the whole business.
Measure	In data warehouse, measure is a property on which calculations (e.g sum, count, average) can be made.

Methods/Algorithms	Definition
Multiway Array Aggregation	The multiway array aggregation (or simply MultiWay) method computes a full data cube by using a multidimensional array as its basic data structure. It is a typical MOLAP approach that uses direct array addressing, where dimension values are accessed via the position or index of their corresponding array locations. Hence, MultiWay cannot per- form any value-based reordering as an optimization technique.
BUC Algorithm	BUC is an algorithm for the computation of sparse and iceberg cubes. Unlike MultiWay, BUC constructs the cube from the apex cuboid toward the base cuboid. This allows BUC to share data partitioning costs. This processing order also allows BUC to prune during construction, using the Apriori property. compute cube *iceberg cube* as select A, B, C, D, count(*) from R cube by A, B, C, D having count(*) >= 3
Star Cubing	Star-Cubing combines the strengths of the other methods we have studied up to this point. It integrates top-down and bottom-up cube computation and explores both multidimensional aggregation (similar to MultiWay) and Apriori like pruning (similar to BUC). It operates from a data structure called a star-tree, which performs lossless data compression, thereby reducing the computation time and memory requirements. The Star-Cubing algorithm explores both the bottom-up and top-down computation models as follows: On the global computation order, it uses the bottom-up model. However, it has a sublayer underneath based on the top-down model, which explores the notion of *shared dimensions*, as we shall see in the following. This integration allows the algorithm to aggregate on multiple dimensions while still partitioning parent group-by's and pruning child group-by's that do not satisfy the iceberg condition.

Classification And Prediction

Basic Concepts

Name	Description
Classification	Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels. For example, we can build a classification model to categorize bank loan applications as either safe or risky. Such analysis can help provide us with a better understanding of the data at large. Many classification methods have been proposed by researchers in machine learn- ing, pattern recognition, and statistics. Most algorithms are memory resident, typically assuming a small data size. Recent data mining research has built on such work, developing scalable classification and prediction techniques capable of handling large amounts of disk-resident data. Classification has numerous applications, including fraud detection, target marketing, performance prediction, manufacturing, and medical diagnosis.
Overfitting	Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.
Underfitting	Underfitting refers to a model that can neither model the training data nor generalize to new data. An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data. Underfitting is often not discussed as it is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms. Nevertheless, it does provide a good contrast to the problem of overfitting.

Name	Description
K fold Cross Validation	In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or "folds," $D_1, D_2,..., D_k$, each of approximately equal size. Training and testing is performed k times. In iteration i, partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model. That is, in the first iteration, subsets $D_2,..., D_k$ collectively serve as the training set to obtain a first model, which is tested on D_1; the second iteration is trained on subsets $D_1, D_3,..., D_k$ and tested on D_2; and so on. Unlike the holdout and random subsampling methods, here each sample is used the same number of times for training and once for testing. For classification, the accuracy estimate is the overall number of correct classifications from the k iterations, divided by the total number of tuples in the initial data.

Classification vs Prediction vs Clustering

Classification	Clustering
used for supervised learning	used for unsupervised learning
process of classifying the input instances based on their corresponding class labels	grouping the instances based on their similarity without the help of class labels
it has labels so there is need of training and testing dataset for verifying the model created	there is no need of training and testing dataset
more complex as compared to clustering	less complex as compared to classification
Logistic regression, Naive Bayes classifier, Support vector machines etc.	k-means clustering algorithm, Fuzzy c-means clustering algorithm, Gaussian (EM) clustering algorithm etc.

Classification	Prediction
Classification is the method of recognizing to which group; a new process belongs to a background of a training data set containing a new process of observing whose group membership is familiar.	Predication is the method of recognizing the missing or not available numerical data for a new process of observing.
A classifier is built to detect explicit labels.	A predictor will be build that predicts a current valued job or command value.
In classification, authenticity depends on detecting the class label correctly.	In predication, the authenticity depends on how well a given predictor can guess the value of a predicated attribute for new data.
In classification, the sample can be called the classifier.	In predication, the sample can be called the predictor.

Eager Vs Lazy Learning

Eager	Lazy
It is faster at classification than lazy classification because it constructs a generalization model before receiving any new tuples to classify.	It is slower at classification because classifiers are not built until new tuples need to be classified.
It must commit to a single hypothesis that covers the entire instance space, which can decrease classification, and more time is needed for training.	It uses a richer hypothesis space, which can improve classification accuracy. It requires less time for training than eager classification.
Weights can be assigned to attributes, which can improve classification accuracy.	Attributes are all equally weighted, which can decrease classification accuracy.
Examples include Bayesian Classification, Decision Trees, Neural Networks	Examples include k-nearest neighbour, case based reasoning.

ID3	CART
During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). This work expanded on earlier work on concept learning systems, described by E. B. Hunt, J. Marin, and P. T. Stone.	In 1984, a group of statisticians (L. Breiman, J. Friedman, R. Olshen, and C. Stone) published the book Classification and Regression Trees (CART), which described the generation of binary decision trees.
ID3 and CART were invented independently of one another at around the same time, yet follow a similar approach for learning decision trees from training tuples. These two cornerstone algorithms spawned a flurry of work on decision tree induction. ID3 and CART adopt a greedy (i.e., non backtracking) approach in which deci- sion trees are constructed in a top-down recursive divide-and-conquer manner. Most algorithms for decision tree induction also follow a top-down approach, which starts with a training set of tuples and their associated class labels. The training set is recur- sively partitioned into smaller subsets as the tree is being built.	
Attribute Selection Methods	
ID3 uses **information gain** as its attribute selection measure. This measure is based on pioneering work by Claude Shannon on information theory, which studied the value or "information content" of messages.	The Gini index is used in CART. Using the notation previously described, the Gini index measures the impurity of D, a data partition or set of training tuples. The Gini index considers a binary split for each attribute.
$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$ where p_i is the nonzero probability that an arbitrary tuple in D belongs to class C_i and is estimated by $\|C_{i,D}\|/\|D\|$. A log function to the base 2 is used, because the information is encoded in bits. $Info(D)$ is just the average amount of information needed to identify the class label of a tuple in D. Note that, at this point, the information we have is based solely on the proportions of tuples of each class. $Info(D)$ is also known as the entropy of D.	$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$ where p_i is the probability that a tuple in D belongs to class C_i and is estimated by $\|C_{i,D}\|/\|D\|$. The sum is computed over m classes.

ID3	CART
This is equivalent to saying that we want to partition on the attribute A that would do the "best classification," so that the amount of information still required to finish classifying the tuples is minimal (i.e., minimum $Info_A(D)$).	The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute. This attribute and either its splitting subset (for a discrete-valued splitting attribute) or split-point (for a continuous-valued splitting attribute) together form the splitting criterion.

Pruning Decision Trees

When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers. Tree pruning methods address this problem of *overfitting* the data. Such methods typically use statistical measures to remove the least-reliable branches.

Pruned trees tend to be smaller and less complex and, thus, easier to comprehend. They are usually faster and better at correctly classifying independent test data (i.e., of previously unseen tuples) than unpruned trees.

Prepruning	Postpruning
In the prepruning approach, a tree is "pruned" by halting its construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node). Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.	The second and more common approach is postpruning, which removes subtrees from a "fully grown" tree.
When constructing a tree, measures such as statistical significance, information gain, Gini index, and so on, can be used to assess the goodness of a split. If partitioning the tuples at a node would result in a split that falls below a prespecified threshold, then fur- ther partitioning of the given subset is halted. There are difficulties, however, in choosing an appropriate threshold. High thresholds could result in oversimplified trees, whereas low thresholds could result in very little simplification.	A subtree at a given node is pruned by removing its branches and replacing it with a leaf. The leaf is labeled with the most frequent class among the subtree being replaced.
Requires less computation	Requires more computation
Gives less reliable trees	Gives more reliables trees

The cost complexity pruning algorithm used in CART is an example of the postprun- ing approach. This approach considers the cost complexity of a tree to be a function of the number of leaves in the tree and the error rate of the tree (where the error rate is the percentage of tuples misclassified by the tree). It starts from the bottom of the tree. For each internal node, N , it computes the cost complexity of the subtree at N , and the cost complexity of the subtree at N if it were to be pruned (i.e., replaced by a leaf node). The two values are compared. If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned. Otherwise, it is kept. A pruning set of class-

labeled tuples is used to estimate cost complexity. This set is independent of the training set used to build the unpruned tree and of any test set used for accuracy estimation. The algorithm generates a set of progressively pruned trees. In general, the smallest decision tree that minimizes the cost complexity is preferred.

Rule Extraction from Decision trees

To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each splitting criterion along a given path is logically ANDed to form the rule antecedent ("IF" part). The leaf node holds the class prediction, forming the rule consequent ("THEN" part).

Eg:

*R*1: **IF** *age = youth* **AND** *student = no* **THEN** *buys computer = no*
*R*2: **IF** *age = youth* **AND** *student = yes* **THEN** *buys computer = yes*
*R*3: **IF** *age = middle aged* **THEN** *buys computer = yes*
*R*4: **IF** *age = senior* **AND** *credit rating = excellent* **THEN** *buys computer = yes*
*R*5: **IF** *age = senior* **AND** *credit rating = fair* **THEN** *buys computer = no*

Näive Bayes Classifier

Bayesian classifiers are statistical classifiers. They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

Bayesian classification is based on Bayes' theorem. Studies comparing classification algorithms have found a simple Bayesian classifier known as the näive *Bayesian classifier* to be comparable in performance with decision tree and selected neural network classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases.

Näive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called *class-conditional independence*. It is made to simplify the computations involved and, in this sense is considered "näive"

Bayes' theorem is useful in that it provides a way of calculating the posterior probability, $P(H|X)$, from $P(H)$, $P(X|H)$, and $P(X)$. Bayes' theorem is

$$P(H \mid X) = \frac{P(X \mid H) * P(H)}{P(X)}$$

This technique for probability estimation is known as the **Laplacian correction** or **Laplace estimator**, named after Pierre Laplace, a French mathematician who lived from 1749 to 1827. If we have, say, q counts to which we each add one, then we must remember to add q to the corresponding denominator used in the probability calculation.

Laplace Smoothing

Laplace smoothing is a smoothing technique that handles the problem of zero probability in Naïve Bayes. Using Laplace smoothing, we can represent P(w'|positive) as

$$P(w'|positive) = \frac{\text{number of reviews with w' and y = positive } + \ \alpha}{\text{N } + \ \alpha * K}$$

Here,

alpha represents the smoothing parameter,

K represents the number of dimensions (features) in the data, and

N represents the number of reviews with y=positive

If we choose a value of alpha!=0 (not equal to 0), the probability will no longer be zero even if a word is not present in the training dataset.

Advantages	Easy to Implement
	Good Results obtained in most cases
Disadvantages	Class conditional independence assumption causes loss of accuracy
	Practically inter dependencies among variables cannot be modelled by Näive Bayes

Näive Bayes for Text Classification

Multinomial Näive Bayes	Bernoulli Näive Bayes
Its is used when we have discrete data (e.g. movie ratings ranging 1 and 5 as each rating will have certain frequency to represent). In text learning we have the count of each word to predict the class or label.	It assumes that all our features are binary such that they take only two values. Means 0s can represent “word does not occur in the document” and 1s as "word occurs in the document" .
Multinomial NB will classify a document based on the counts it finds of multiple keywords	Bernoulli NB can only focus on a single keyword, but will also count how many times that keyword does not occur in the document.
More Accurate	Less Accurate

Support Vector Machines (SVM)

In a nutshell, an SVM is an algorithm that works as follows. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (i.e., a “decision boundary” separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM finds this hyperplane using *support vectors* (“essential” training tuples) and *margins* (defined by the support vectors).

Although the training time of even the fastest SVMs can be extremely slow, they are highly accurate, owing to their ability to model complex nonlinear decision boundaries. They are much less prone to overfitting than other methods. The support vectors found also provide a compact description of the learned model. SVMs can be used for numeric prediction as well as classification. They have been applied to a number of areas, including handwritten digit recognition, object recognition, and speaker identification, as well as benchmark time-series prediction tests.

Clustering

Cluster Analysis

Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a clustering. In this context, different clustering methods may generate different clusterings on the same data set. The partitioning is not performed by humans, but by the clustering algorithm. Hence, clustering is useful in that it can lead to the discovery of previously unknown groups within the data.

Example	Description
Example 1	In image recognition, clustering can be used to discover clusters or "subclasses" in handwritten character recognition systems. Suppose we have a data set of handwritten digits, where each digit is labeled as either 1, 2, 3, and so on. Note that there can be a large variance in the way in which people write the same digit. Take the number 2, for example. Some people may write it with a small circle at the left bottom part, while some others may not. We can use clustering to determine subclasses for "2," each of which represents a variation on the way in which 2 can be written. Using multiple models based on the subclasses can improve overall recognition accuracy.
Example 2	Clustering has also found many applications in Web search. For example, a keyword search may often return a very large number of hits (i.e., pages relevant to the search) due to the extremely large number of web pages. Clustering can be used to organize the search results into groups and present the results in a concise and easily accessible way. Moreover, clustering techniques have been developed to cluster documents into topics, which are commonly used in information retrieval practice.
Example 3	Clustering can also be used for outlier detection, where outliers (values that are "far away" from any cluster) may be more interesting than common cases. Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce. For example, exceptional cases in credit card transactions, such as very expensive and infrequent purchases, may be of interest as possible fraudulent activities

Requirements for Clustering

Requirement	Definition
Scalability	Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions or even billions of objects, particularly in Web search scenarios. Clustering on only a sample of a given large data set may lead to biased results. Therefore, highly scalable clustering algorithms are needed.
Ability to deal with different types of attributes	Many algorithms are designed to cluster numeric (interval-based) data. However, applications may require clustering other data types, such as binary, nominal (categorical), and ordinal data, or mixtures of these data types. Recently, more and more applications need clustering techniques for complex data types such as graphs, sequences, images, and documents.
Discovery of clusters with arbitrary shape	Algorithms based on such distance measures tend to find spherical clusters with similar size and density. However, a cluster could be of any shape. Consider sensors, for example, which are often deployed for environment surveillance. Cluster analysis on sensor readings can detect interesting phenomena. We may want to use clustering to find the frontier of a running forest fire, which is often not spherical. It is important to develop algorithms that can detect clusters of arbitrary shape.
Requirements for domain knowledge to determine input parameters	Many clus- tering algorithms require users to provide domain knowledge in the form of input parameters such as the desired number of clusters. Consequently, the clustering results may be sensitive to such parameters. Parameters are often hard to determine, especially for high-dimensionality data sets and where users have yet to grasp a deep understanding of their data. Requiring the specification of domain knowledge not only burdens users, but also makes the quality of clustering difficult to control.
Ability to deal with noisy data	Most real-world data sets contain outliers and/or missing, unknown, or erroneous data. Sensor readings, for example, are often noisy—some readings may be inaccurate due to the sensing mechanisms, and some readings may be erroneous due to interferences from surrounding transient objects. Clustering algorithms can be sensitive to such noise and may produce poor-quality clusters. Therefore, we need clustering methods that are robust to noise.

Requirement	Definition
Incremental clustering and insensitivity to input order	In many applications, incremental updates (representing newer data) may arrive at any time. Some clus- tering algorithms cannot incorporate incremental updates into existing clustering structures and, instead, have to recompute a new clustering from scratch. Cluster- ing algorithms may also be sensitive to the input data order. That is, given a set of data objects, clustering algorithms may return dramatically different clusterings depending on the order in which the objects are presented. Incremental clustering algorithms and algorithms that are insensitive to the input order are needed.
Capability of clustering high-dimensionality data	data set can contain numerous dimensions or attributes. When clustering documents, for example, each keyword can be regarded as a dimension, and there are often thousands of keywords. Most clustering algorithms are good at handling low-dimensional data such as data sets involving only two or three dimensions. Finding clusters of data objects in a high- dimensional space is challenging, especially considering that such data can be very sparse and highly skewed.
Constraint-based clustering	Real-world applications may need to perform clus- tering under various kinds of constraints. Suppose that your job is to choose the locations for a given number of new automatic teller machines (ATMs) in a city. To decide upon this, you may cluster households while considering constraints such as the city's rivers and highway networks and the types and number of customers per cluster. A challenging task is to find data groups with good clustering behavior that satisfy specified constraints.
Interpretability and usability	Users want clustering results to be interpretable, comprehensible, and usable. That is, clustering may need to be tied in with spe- cific semantic interpretations and applications. It is important to study how an application goal may influence the selection of clustering features and clustering methods.

Clustering Algorithms

Partition-based Clustering	Hierarchical Clustering	Graph-based Clustering
The simplest and most fundamental version of cluster analysis is partitioning, which organizes the objects of a set into several exclusive groups or clusters. To keep the problem specification concise, we can assume that the number of clusters is given as background knowledge. This parameter is the starting point for partitioning methods.	While partitioning methods meet the basic clustering requirement of organizing a set of objects into a number of exclusive groups, in some situations we may want to partition our data into groups at different levels such as in a hierarchy. A hierarchical clustering method works by grouping data objects into a hierarchy or “tree” of clusters.	
Formally, given a data set, D, of n objects, and k, the number of clusters to form, a partitioning algorithm organizes the objects into k partitions ($k \leq n$), where each par- tition represents a cluster. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are “similar” to one another and “dissimilar” to objects in other clusters in terms of the data set attributes.	Consider handwritten character recognition as another example. A set of handwrit- ing samples may be first partitioned into general groups where each group corresponds to a unique character. Some groups can be further partitioned into subgroups since a character may be written in multiple substantially different ways. If necessary, the hierarchical partitioning can be continued recursively until a desired granularity is reached.	
Eg: k-means, k-mediods	Eg: Agglomerative Clustering	

K-means Clustering

k means Clustering	
Definition	Suppose a data set, D, contains n objects in Euclidean space. Partitioning methods dis- tribute the objects in D into k clusters, $C_1,...,C_k$, that is, $C_i \subset D$ and $C_i \cap C_j = \emptyset$ for $(1 \leq i, j \leq k)$. An objective function is used to assess the partitioning quality so that objects within a cluster are similar to one another but dissimilar to objects in other clusters. This is, the objective function aims for high intracluster similarity and low intercluster similarity.
Advantages	Relatively simple to implement.
	Scales to large data sets.
	Guarantees convergence.
	Can warm-start the positions of centroids.
	Easily adapts to new examples.
	Generalizes to clusters of different shapes and sizes, such as elliptical clusters.
Disadvantages	Choosing K Manually
	For a low k, you can mitigate this dependence by running k-means several times with different initial values and picking the best result. As k increases, you need advanced versions of k-means to pick better values of the initial centroids (called k-means seeding)
	k-means has trouble clustering data where clusters are of varying sizes and density
	Centroids can be dragged by outliers, or outliers might get their own cluster instead of being ignored.
	Scaling with number of dimensions. As the number of dimensions increases, a distance-based similarity measure converges to a constant value between any given examples

Agglomerative Clustering

Agglomerative Clustering	
Definition	An agglomerative hierarchical clustering method uses a bottom-up strategy. It typ- ically starts by letting each object form its own cluster and iteratively merges clusters into larger and larger clusters, until all the objects are in a single cluster or certain termi- nation conditions are satisfied. The single cluster becomes the hierarchy's root. For the merging step, it finds the two clusters that are closest to each other (according to some similarity measure), and combines the two to form one cluster. Because two clusters are merged per iteration, where each cluster contains at least one object, an agglomerative method requires at most n iterations.
Single Link Clustering	When an algorithm uses the *minimum distance*, $d_{min}(C_i,C_j)$, to measure the distance between clusters, it is sometimes called a nearest - neighbour clustering algorithm. Moreover, if the clustering process is terminated when the distance between nearest clusters exceeds a user-defined threshold, it is called a single-linkage algorithm.
	Strength: Can handle non-elliptical shapes
	Limitation: Sensitive to noise and outliers
Complete Link Clustering	When an algorithm uses the *maximum distance*, $d_{max}(C_i, C_j)$, to measure the distance between clusters, it is sometimes called a farthest-neighbour clustering algorithm. If the clustering process is terminated when the maximum distance between nearest clusters exceeds a user-defined threshold, it is called a complete-linkage algorithm.
	Strength: Less susceptible to noise and outliers
	Limitations: Tends to break large clusters and biased towards globular clusters.
Group Average Clustering	Group-average agglomerative clustering or GAAC evaluates cluster quality based on all similarities between documents, thus avoiding the pitfalls of the single-link and complete-link criteria, which equate cluster similarity with the similarity of a single pair of documents. GAAC is also called group-average clustering and average-link clustering . GAAC computes the average similarity SIM-GA of all pairs of documents, including pairs from the same cluster. But self-similarities are not included in the average.
	Strength: Less susceptible to noise and outliers
	Limitations: Biased towards globular clusters.

Agglomerative Clustering	
Limitations	Doesn't Scale well: Time complexity of at least $O(n^2)$, where n is number of total objects
	Can never undo what was previously done.

Spectral Clustering

Spectral Clustering	
Definition	Spectral clustering is effective in high-dimensional applications such as image pro- cessing. Theoretically, it works well when certain conditions apply. Scalability, however, is a challenge. Computing eigenvectors on a large matrix is costly. Spectral clustering can be combined with other clustering methods, such as biclustering.
Advantages	Usually Better Quality than other algorithms
	Can be thought of (non-linear) dimensionality reduction or embedding.
	Freedom to construct a (sparse) G to preserve local similarity/ connectivity.
	Only requires some similarity measure.
	Could be more efficient than k-means for high-dimensional sparse vectors (esp. if k-means is not fully optimized for such case).
Disadvantages	Still need to determine k
	Assumes clusters are of similar sizes.
	Does not scale well with large datasets; but more scalable variants exist.
	One of the relaxation of the original NP-hard problem – may not be the tightest relaxation.

Association Rule Mining

Apriori Algorithm

Apriori Background	Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for min- ing frequent itemsets for Boolean association rules [AS94b]. The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset prop- erties
Apriori Property	**Apriori property:** *All nonempty subsets of a frequent itemset must also be frequent*.
	The Apriori property is based on the following observation. By definition, if an item- set I does not satisfy the minimum support threshold, min sup, then I is not frequent, that is, P(I) < min sup. If an item A is added to the itemset I, then the resulting itemset (i.e., I ∪ A) cannot occur more frequently than I . Therefore, I ∪ A is not frequent either, that is, P(I∪A)<min sup. This property belongs to a special category of properties called antimonotonicity in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. It is called antimonotonicity because the property is monotonic in the context of failing a test.
Apriori Algorithm	Find frequent item sets using an iterative level-wise approach based on candidate generation.
Drawbacks	At each step, candidate sets have to be built.
	To build the candidate sets, the algorithm has to repeatedly scan the database

FP Growth Algorithm

FP Growth Algorithm	
Definition	An interesting method in this attempt is called frequent pattern growth, or simply FP-growth, which adopts a divide-and-conquer strategy as follows. First, it compresses the database representing frequent items into a frequent pattern tree, or FP-tree, which retains the itemset association information. It then divides the compressed database into a set of conditional databases (a special kind of projected database), each associated with one frequent item or “pattern fragment,” and mines each database separately. For each “pattern fragment,” only its associated data sets need to be examined. Therefore, this approach may substantially reduce the size of the data sets to be searched, along with the “growth” of patterns being examined.