Data Warehousing

1 Characteristics a data warehouse

A data warehouse is an environment where a user can find strategic information. However, it put data such a way that a user can find strategic information to make better decisions. It is not a product containing a single software or hardware to provide strategic information. The basic characterise of a data warehouse are:

- Fully user driven not event driven
- Data store for analysis and decision support
- Capable to provide solutions for very complex and unpredictable query
- Flexible, responsive, and answer in an interactive way (for example, Subway management want to know "which sandwich is most profitable?" Then the next question may be "which store sells most profitable sandwich?" Then the next question may be" which classes of customer are having most profitable sandwiches?" a data warehouse environment can answer the above queries interactively.)

2 Defining a data warehouse

Bill Inmon, father of data warehouse, provides four characteristics of data warehouse: subject oriented, integrated, time variant, and non-volatile. A data warehouse is build to support management's decisions.

2.1 Subject-oriented Data

In operational system, data is application processing oriented. For example, order processing application need data for entering order, checking availability, authenticate the customer's payment, and arranging shipment the order to the customer.

On the other hand, a data warehouse stores data by subject. Data store by business subjects which may vary organizations to organizations. For example, a manufacturing company's critical business subjects are sales, inventory, and delivery of a shipment. Subway critical business subject is sales at the point of sales terminal (POST).

2.2 Integrated data

In a data warehouse, data comes from various operational systems. Venda Ltd has two subways. Both stores have different operational systems. One store is using SUBSHOP 08 and another one is using SUBSHOP 05. So the inconsistencies should be removed from both operational systems before data stored in the data warehouse.

2.3 Time-Variant Data

In an operational system, data contains current values. For example, an account receives system store information about the outstanding balance in the customer balance. On the other hand, data warehouse store historical data for analysis and decision making. For example, if Subway management wants to discover the hidden patterns of a customer spending, they need data from both current purchase and past purchase. The time variant of data will help Subway management to analyse past data by relating current information to forecasts the future.

2.4 Non-volatile data

Data from operational systems are transformed, integrated, and store in a data warehouse for analysis purpose not to run day to day business operations. For example, when an order comes from a customer, a data warehouse will be used to know the current status of the stock. The operational system will be used to check the status of the product. Usually data in a data warehouse is not update. Data read form a data warehouse for query and analysis.

3 Dimensional Modelling

Dimensional Modelling is a logical design technique deriving its name from business dimensions. This model is suitable for queries and analysis. The characteristics of the dimensional modelling are:

- ✓ It provide the way to access the data
- ✓ It is a query centric data model.
- ✓ It shows the interactions among dimensions and fact tables.
- ✓ It is flexible for drilling down, rolling up along dimension hierarchy

One of the major problems in the OLAP is that usage of information is totally unpredictable. Users cannot define their requirements clearly. Even they do not know how they would like to use the information or process it On the other hand, in OLTP precise functions are specified by end-users.

3.1 Star Schema

Star Schema also known as dimensional model. It forms ‘star-like’ structure, which is called a star schema or star join. Every dimensional model (DM) is composed of one (or more) fact tables, and a number of dimension tables. Fact table contains numeric data value and dimension table contain description of the fact table.

For example, what is the sale amount in Consumer Product category, for young customers in April 2008? Here sales amount is in the fact table and dimension tables are customer and time. Basically fact tables are narrow, big (many rows), numeric, growing over time. On the other hand, dimension tables are wide, small (few rows), descriptive, static.

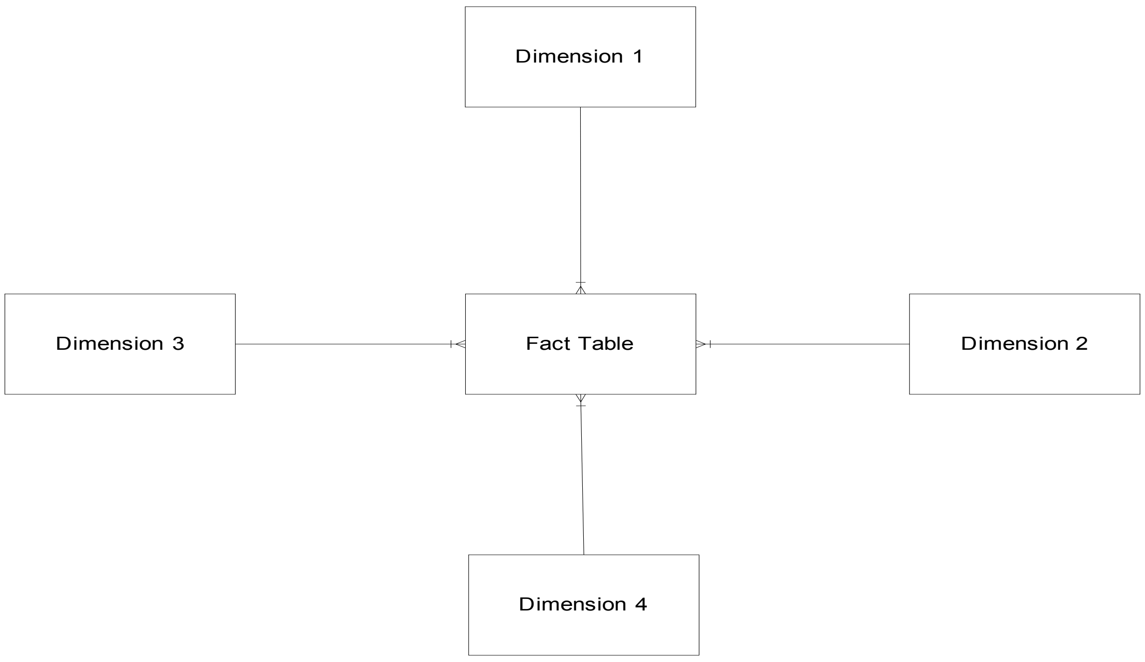

Figure: Star Schema

3.2 Snowflake Schema

In order to keep the data warehouse simple and easy to understand, dimension tables are not in normal form. They contain huge redundant information about hierarchies. Normalizing dimension tables leads to snowflake schema. In reality, snow flaking not recommended in most cases. Because more tables introduce more complex design, more joins and make the queries slower.

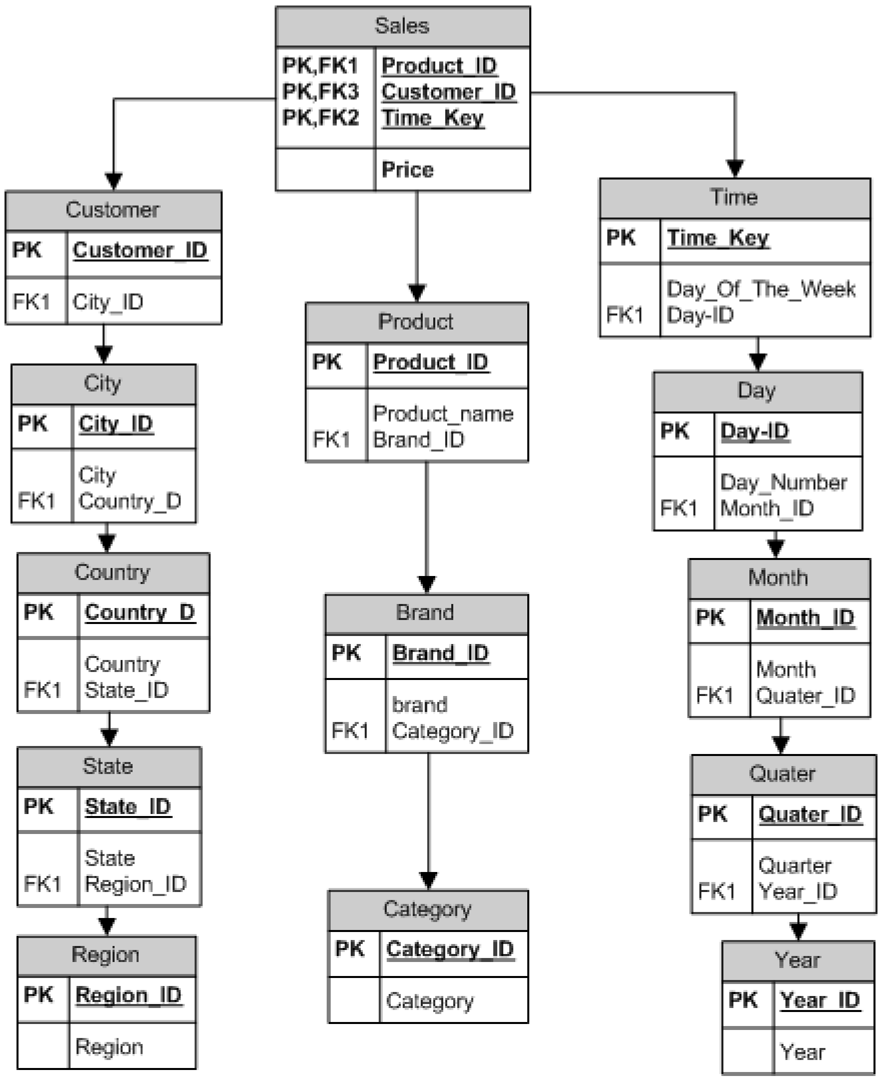

Figure 2.4 snowflake schema

3.3 Star Flake Schema

The **star flake schema** is a hybrid schema derived from star and snowflake schema. The star flake schema contains a fact table and a set of demoralised and normalised dimension tables.

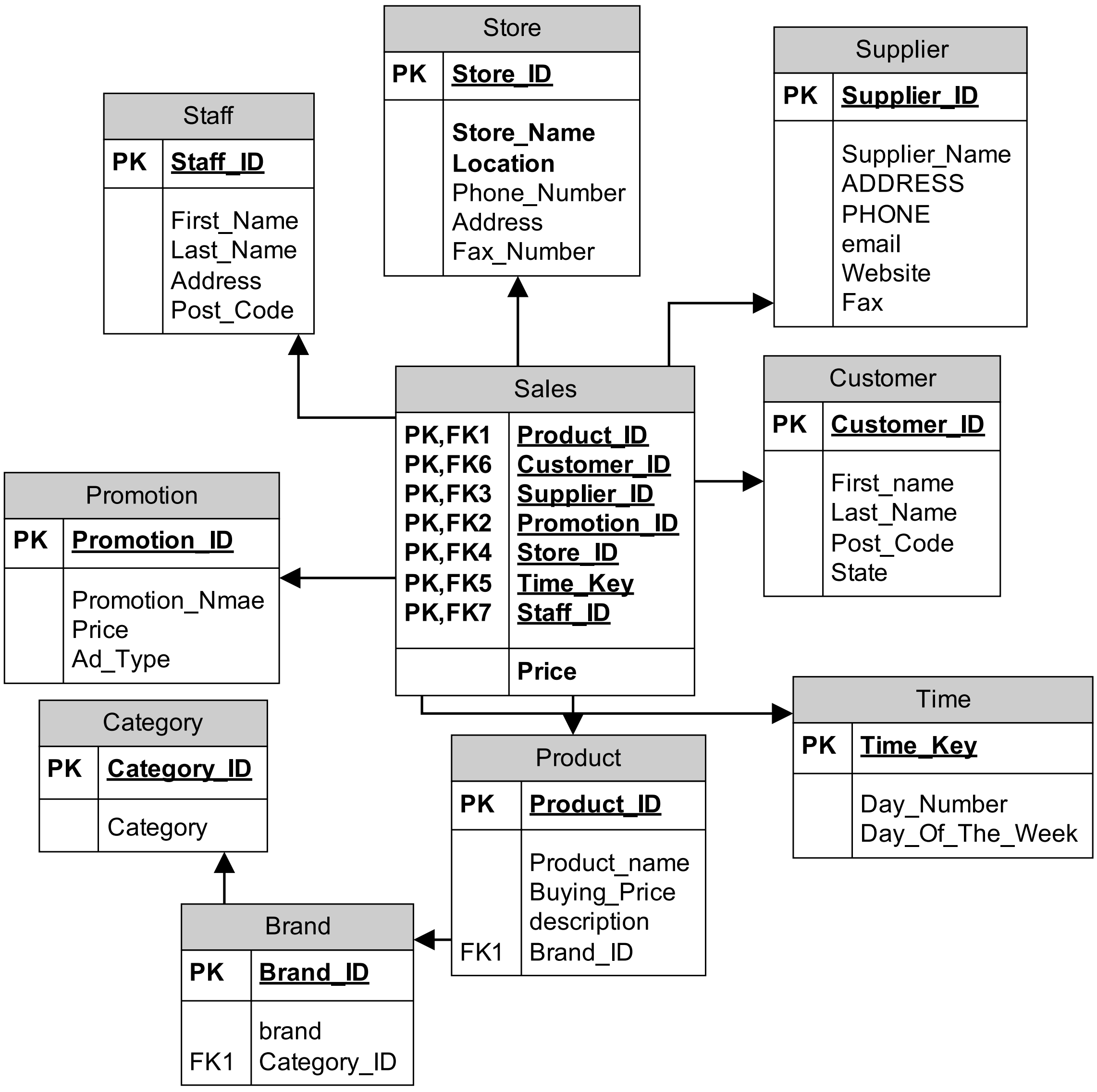

Figure: star slake schema

3.4 OLTP vs. OLAP

	OLTP	OLAP
Source of data	Data comes from day to day transaction	Data comes from various operational systems
Data content	Data contains current values	Data is historical, derived, summarized
Purpose of data	To run day to day operation	To make decisions
Access Type	read, write, delete, update, add	Data is used for query purpose
Usage	Users can define almost all the requirements at the early stage	Unpredictable
Response time	Very fast (Sub-second)	Depends on the complexity and data involve in the query (Several

		second to minutes).
Users	Depends on the system (thousands).	The number of users is less than OPTP (hundreds).
Database Design	Entity Relationship Modeling is used to develop the system. Data is highly normalized	Data is demoralized. Dimensional modeling is used for the OLAP design.
Space Requirement	OLTP takes less space than OLAP because is archived regularly (100 MB to GB)	OLAP takes much more space than OLTP because this the place where data is archived from operational system (100 GB to TB)
Queries	Queries are relatively simple and need retrieval of few records	Queries are complex, often needs aggregations, multi dimensional views of data
Access Frequency	Access frequency is very high as it is used for day to day operation	Access frequency is low as it is used for query.
What the data reveals	OLTP provides information about business processes	OLAP provides multidimensional views of business activities

Table: OLTP vs. OLAP

Data Mining

The term **data mining** refers analyzing large databases to find useful patterns. Like knowledge discovery in artificial intelligence (also called machine learning), or statistical analysis, data mining attempts to discover rules and patterns from data. However, data mining differs from machine learning and statistics in that it deals with large volumes of data, stored primarily on disk. That is, data mining deals with "knowledge discovery in databases."
Some types of knowledge discovered from a database can be represented by a set of **rules**. The following is an example of a rule, stated informally: "Young women with annual incomes greater than $50,000 are the most likely people to buy small sports cars." Of course such rules are not universally true, and have degrees of "support" and "confidence," as we shall see. Other types of knowledge are represented by equations relating different variables to each other, or by other mechanisms for predicting outcomes when the values of some variables are known.

3.5 Applications of Data Mining

The discovered knowledge has numerous applications. The most widely used applications are those that require some sort of **prediction**. For instance, when a person applies for a credit card, the credit-card company wants to predict if the person is a good credit risk. The prediction is to be based on known attributes of the person, such as age, income, debts, and past debt repayment history. Rules for making the prediction are derived from the same attributes of past and current credit card holders, along with their observed behavior, such as whether they defaulted on their credit card dues. Other types of prediction include predicting which customers may switch over to a competitor (these customers may be offered special discounts to tempt them not to switch), predicting which people are likely to respond to promotional mail ("junk mail"), or predicting what types of phone calling card usage are likely to be fraudulent.

Another class of applications looks for **associations**, for instance, books that tend to be bought together. If a customer buys a book, an online bookstore may suggest other associated books. If a person buys a camera, the system may suggest accessories that tend to be bought along with cameras. A good salesperson is aware of such patterns and exploits them to make additional sales. The challenge is to automate the process. Other types of associations may lead to discovery of causation. For instance, discovery of unexpected associations 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.

Associations are an example of **descriptive patterns**. **Clusters** are another example of such patterns. For example, over a century ago a cluster of typhoid cases was found around a well, which led to the discovery that the water in the well was contaminated and was spreading typhoid. Detection of clusters of disease remains important even today.

3.6 Classification

Prediction is one of the most important types of data mining. We outline what is classification, study techniques for building one type of classifiers, called decision tree classifiers, and then study other prediction techniques.

Abstractly, the **classification** problem is this: Given that items belong to one of 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 can be done by finding rules that partition the given data into disjoint groups. For instance, suppose that a credit-card company wants to decide whether or not to give a credit card to an applicant. The company has a variety of information about the person, such as her age, educational background, annual income, and current debts that it can use for making a decision.

Some of this information could be relevant to the credit worthiness of the applicant, whereas some may not be. To make the decision, the company assigns a creditworthiness level of excellent, good, average, or bad to each of a sample set of *current* customers according to each customer's payment history. Then, the company attempts to find rules that classify its current customers into excellent, good, average, or bad, on the basis of the information about the person, other than the actual payment history (which is unavailable for new customers).

Let us consider just two attributes: education level (highest degree earned) and income. The rules may be of

the following form:

$$\textit{person P, P.degree} = \textit{masters}\ \textbf{and}\ \textit{P.income} > 75,000$$
$$\textit{P.credit} = \textit{excellent}$$
$$\textit{person P, P.degree} = \textit{bachelors}\ \textbf{or}$$
$$(\textit{P.income} \geq 25,000\ \textbf{and}\ \textit{P.income} \leq 75,000) \quad \textit{P.credit} = \textit{good}$$

Similar rules would also be present for the other credit worthiness levels (average and bad).
The process of building a classifier starts from a sample of data, called a **training set**. For each tuple in the training set, the class to which the tuple belongs is already known. For instance, the training set for a credit-card application may be the existing customers, with their credit worthiness determined from their payment history. The actual data, or population, may consist of all people, including those who are not existing customers.

3.7 Decision Tree Classifiers

The decision tree classifier is a widely used technique for classification. As the name suggests, **decision tree classifiers** use a tree; each leaf node has an associated class, and each internal node has a predicate (or more generally, a function) associated with it. Figure 22.6 shows an example of a decision tree.
To classify a new instance, we start at the root, and traverse the tree to reach a leaf; at an internal node we evaluate the predicate (or function) on the data instance,

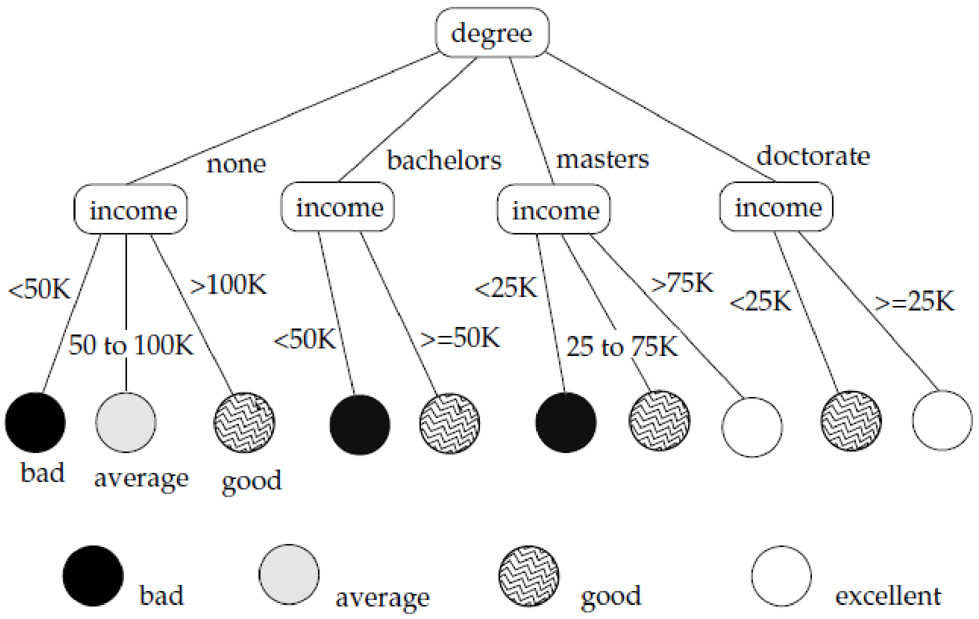

Figure 22.6 Classification tree.

to find which child to go to. The process continues till we reach a leaf node. For example, if the degree level of a person is masters, and the persons income is 40K, starting from the root we follow the edge labeled "masters," and from there the edge labeled "25K to 75K," to reach a leaf. The class at the leaf is "good," so we predict that the credit risk of that person is good.

3.8 Association Rules

Retail shops are often interested in **associations** between different items that people buy. Examples of such associations are:

- 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 information can be used in several ways. When a customer buys a particular book, an online shop may suggest associated books. A grocery shop may decide to place bread close to milk, since they are often bought together, to help shoppers finish their task faster. Or the shop may place them at opposite ends of a row, and place other associated items

in between to tempt people to buy those items as well, as the shoppers walk from one end of the row to the other. A shop that offers discounts on one associated item may not offer a discount on the other, since the customer will probably buy the other anyway.
An example of an association rule is

bread milk

In the context of grocery-store purchases, the rule says that customers who buy bread also tend to buy milk with a high probability. An association rule must have an associated **population**: the population consists of a set of **instances**. In the grocery-store example, the population may consist of all grocery store purchases; each purchase is an instance. In the case of a bookstore, the population may consist of all people who made purchases, regardless of when they made a purchase. Each customer is an instance.
Here, the analyst has decided that when a purchase is made is not significant, whereas for the grocery-store example, the analyst may have decided to concentrate on single purchases, ignoring multiple visits by the same customer. Rules have an associated *support*, as well as an associated *confidence*. These are defined in the context of the population:

• **Support** is a measure of what fraction of the population satisfies both the antecedent and the consequent of the rule.

For instance, suppose only 0·001 percent of all purchases include milk and screwdrivers. The support for the rule

milk screwdrivers

is low. The rule may not even be statistically significant—perhaps there was only a single purchase that included both milk and screwdrivers. Businesses are usually not interested in rules that have low support, since they involve few customers, and are not worth bothering about.
On the other hand, if 50 percent of all purchases involve milk and bread, then support for rules involving bread and milk (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.

• **Confidence** is a measure of how often the consequent is true when the antecedent is true. For instance, the rule *bread milk* has a confidence of 80 percent if 80 percent of the purchases that include bread also include milk. A rule with a low confidence is not meaningful. In business
applications, rules usually have confidences significantly less than 100 percent, whereas in other domains, such as in physics, rules may have high confidences.
Note that the confidence of

bread milk

may be very different from the confidence of *milk bread*, although both have the same support.

3.9 Clustering

Intuitively, clustering refers to the problem of finding clusters of points in the given data. The problem of **clustering** can be formalized from distance metrics in several ways. One way is to phrase it as the problem of grouping points into *k* sets (for a given *k*) so that the average distance of points from the *centroid* of their assigned cluster is minimized.5 Another way is to group points so that the average distance between every pair of points in each cluster is

minimized. There are other definitions too; see the bibliographical notes for details. But the intuition behind all these definitions is to group similar points together in a single set.
Another type of clustering appears in classification systems in biology. (Such classification systems do not attempt to *predict* classes, rather they attempt to cluster related items together.) For instance, leopards and humans are clustered under the class mammalia, while crocodiles and snakes are clustered under reptilia. Both mammalian and reptilia come under the common class chordata. The clustering of mammalia has further subclusters, such as carnivora and primates. We thus have **hierarchical clustering**.

Given characteristics of different species, biologists have created a complex hierarchical clustering scheme grouping related species together at different levels of the hierarchy.
Hierarchical clustering is also useful in other domains—for clustering documents, for example. Internet directory systems (such as Yahoo's) cluster related documents in a hierarchical fashion (see Section 22.5.5). Hierarchical clustering algorithms can be classified as **agglomerative clustering** algorithms, which start by building small clusters and then creater higher levels, or **divisive clustering** algorithms, which first create higher levels of the hierarchical clustering, then refine each resulting cluster into lower level clusters.

The statistics community has studied clustering extensively. Database research has provided scalable clustering algorithms that can cluster very large data sets (that may not fit in memory). The Birch clustering algorithm is one such algorithm. Intuitively, data points are inserted into a multidimensional tree structure (based on R-trees, described in Section 23.3.5.3), and guided to appropriate leaf nodes based on nearness to representative points in the internal nodes of the tree. Nearby points are thus clustered together in leaf nodes, and summarized if there are more points than fit in memory. Some post processing after insertion of all points gives the desired overall clustering. See the bibliographical notes for references to the Birch algorithm, and other techniques for clustering, including algorithms for hierarchical clustering.
An interesting application of clustering is to predict what new movies (or books, or music) a person is likely to be interested in, on the basis of:

1. The person's past preferences in movies
2. Other people with similar past preferences
3. The preferences of such people for new movies

One approach to this problem is as follows. To find people with similar past preferences we create clusters of people based on their preferences for movies. The accuracy of clustering can be improved by previously clustering movies by their similarity, so even if people have not seen the same movies, if they have seen similar movies they would be clustered together. We can repeat the clustering, alternately clustering people, then movies, then people, and so on till we reach an equilibrium. Given a new user, we find a cluster of users most similar to that user, on the basis of the user's preferences for movies already seen. We then predict movies in movie clusters that are popular with that user's cluster as likely to be interesting to the new user. In fact, this problem is an instance of *collaborative filtering*, where users collaborate in the task of filtering information to find information of interest.