DATA WAREHOUSING AND DATA MINING
PART – A

UNIT – 1 **6 Hours**

Data Warehousing:

Introduction, Operational Data Stores (ODS), Extraction Transformation Loading (ETL), Data Warehouses. Design Issues, Guidelines for Data Warehouse Implementation, Data Warehouse Metadata.

UNIT – 2 **6 Hours**

Online Analytical Processing (OLAP): Introduction, Characteristics of OLAP systems, Multidimensional view and Data cube, Data Cube Implementations, Data Cube operations, Implementation of OLAP and overview on OLAP Softwares.

UNIT – 3 **6 Hours**

Data Mining: Introduction, Challenges, Data Mining Tasks, Types of Data,Data Preprocessing, Measures of Similarity and Dissimilarity, Data Mining Applications

UNIT – 4 **8 Hours**

Association Analysis: Basic Concepts and Algorithms: Frequent Itemset Generation, Rule Generation, Compact Representation of Frequent Itemsets, Alternative methods for generating Frequent Itemsets, FP Growth Algorithm,Evaluation of Association Patterns

PART - B

UNIT – 5 **6 Hours**

Classification -1 : Basics, General approach to solve classification problem, Decision Trees, Rule Based Classifiers, Nearest Neighbor Classifiers.

UNIT – 6 **6 Hours**

Classification - 2 : Bayesian Classifiers, Estimating Predictive accuracy of classification methods, Improving accuracy of clarification methods, Evaluation criteria for classification methods, Multiclass Problem.

UNIT – 7 **8 Hours**

Clustering Techniques: Overview, Features of cluster analysis, Types of Data and Computing Distance, Types of Cluster Analysis Methods, Partitional Methods, Hierarchical Methods, Density Based Methods, Quality and Validity of Cluster Analysis.

UNIT – 8 **6 Hours**

Web Mining: Introduction, Web content mining, Text Mining, Unstructured Text, Text clustering, Mining Spatial and Temporal Databases.

Text Books:

1. Pang-Ning Tan, Michael Steinbach, Vipin Kumar: Introduction to Data Mining, Pearson Education, 2005.

2. G. K. Gupta: Introduction to Data Mining with Case Studies, 3rdEdition, PHI, New Delhi, 2009.

Reference Books:

1. Arun K Pujari: Data Mining Techniques, 2nd Edition, UniversitiesPress, 2009.

2. Jiawei Han and Micheline Kamber: Data Mining - Concepts and Techniques, 2nd Edition, Morgan Kaufmann Publisher, 2006.

3. Alex Berson and Stephen J. Smith: Data Warehousing,

TABLE OF CONTENTS

UNIT 1

DATA WAREHOUSING

INTRODUCTION

Major enterprises have many computers that run a variety of enterprise applications. For an enterprise with branches in many locations, the branches may have their own systems. For example, in a university with only one campus, the library may run its own catalog and borrowing database system while the student administration may have own systems running on another machine. There might be a separate finance system, a property and facilities management system and another for human resources management. A large company might have the following system.

- Human Resources
- Financials
- Billing
- Sales leads
- Web sales
- Customer support

Such systems are called online transaction processing (OLTP) systems. The OLTP systems are mostly relational database systems designed for transaction processing. The performance of OLTP systems is usually very important since such systems are used to support the users(i.e. staff) that provide service to the customers. The systems

therefore must be able to deal with insert and update operations as well as answering simple queries quickly.

OPERATIONAL DATA STORES

An ODS has been defined by Inmon and Imhoff (1996) as a *subject-oriented, integrated, volatile, current valued data store, containing only corporate detailed data.* A data warehouse is a reporting database that contains relatively recent as well as historical data and may also contain aggregate data.

The ODS is *subject-oriented*. That is, it is organized around the major data subjects of an enterprise. In a university, the subjects might be students, lecturers and courses while in company the subjects might be customers, salespersons and products.

The ODS is *integrated*. That is, it is a collection of subject-oriented data from a variety of systems to provide an enterprise-wide view of the data.

The ODS is *current valued.* That is, an ODS is up-to-date and reflects the current status of the information. An ODS does not include historical data. Since the OLTP systems data is changing all the time, data from underlying sources refresh the ODS as regularly and frequently as possible.

The ODS is *volatile*. That is, the data in the ODS changes frequently as new information refreshes the ODS.

The ODS is *detailed.* That is, the ODS is detailed enough to serve the needs of the operational management staff in the enterprise. The granularity of the data in the ODS does not have to be exactly the same as in the source OLTP system.

ODS Design and Implementation

The extraction of information from source databases needs to be efficient and the quality of data needs to be maintained. Since the data is refreshed regularly and frequently, suitable checks are required to ensure quality of data after each refresh. An ODS would of course be required to satisfy normal integrity constraints, for example, existential integrity, referential integrity and appropriate action to deal with nulls. An ODS is a read only database other than regular refreshing by the OLTP systems. Users should not be allowed to update ODS information.

Populating an ODS involves an acquisition process of extracting, transforming and loading data from OLTP source systems. This process is ETL. Completing populating the database, checking for anomalies and testing for performance are necessary before an ODS system can go online.

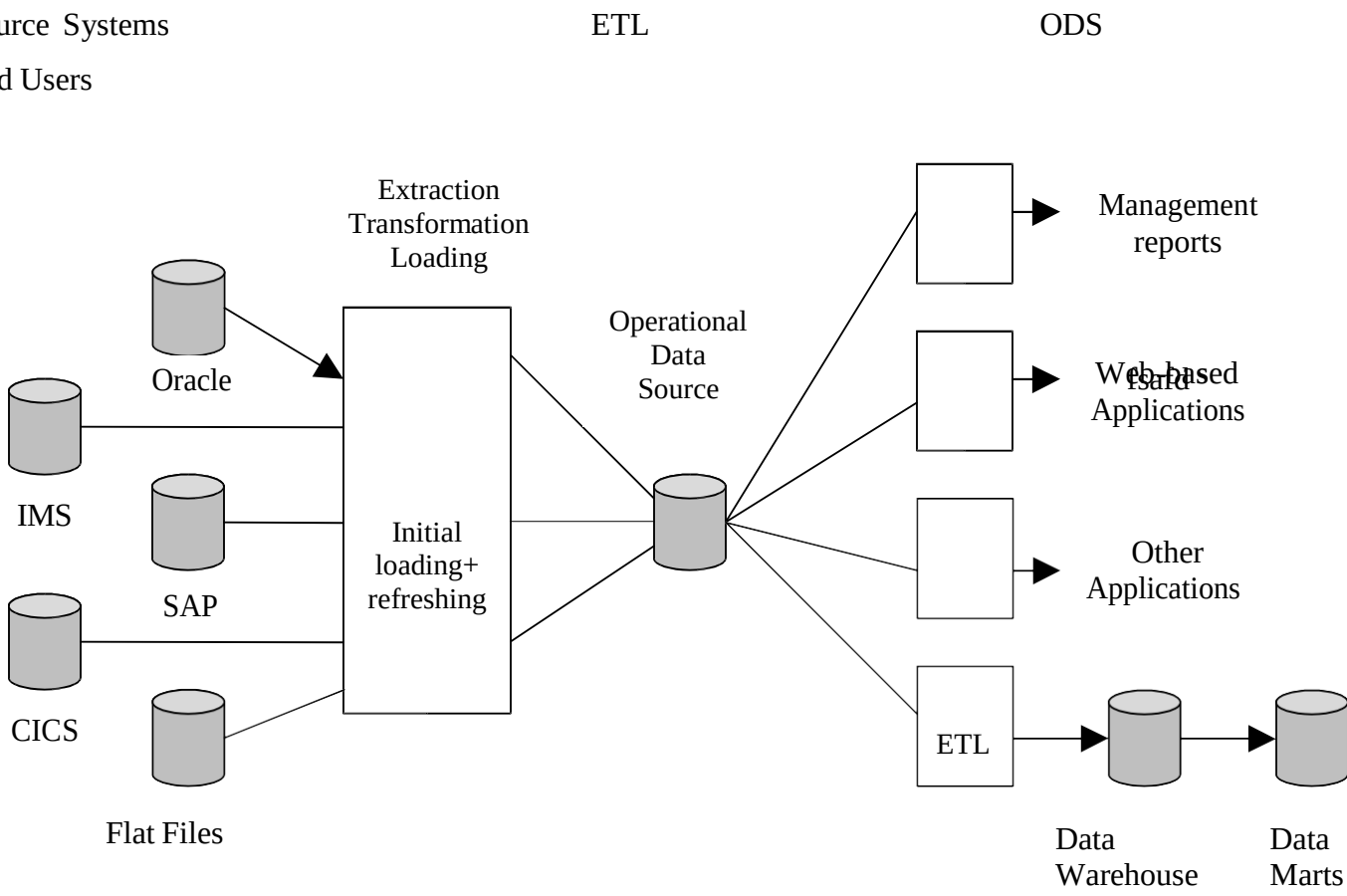


Fig :1.1 A possible Operational Data Store structure

Zero Latency Enterprise (ZLE)

The Gantner Group has used a term **Zero Latency Enterprise (ZLE)** for near real-time integration of operational data so that there is no significant delay in getting information from one part or one system of an enterprise to another system that needs the information.

The heart of a ZLE system is an operational data store.

A ZLE data store is something like an ODS that is integrated and up-to-date. The aim of a ZLE data store is to allow management a single view of enterprise information

by bringing together relevant data in real-time and providing management a “360-degree” view of the customer.

A ZLE usually has the following characteristics. It has a unified view of the enterprise operational data. It has a high level of availability and it involves online refreshing of information. The achieve these, a ZLE requires information that is as current as possible. Since a ZLE needs to support a large number of concurrent users, for example call centre users, a fast turnaround time for transactions and 24/7 availability is required.

ETL

An ODS or a data warehouse is based on a single global schema that integrates and consolidates enterprise information from many sources. Building such a system requires data acquisition from OLTP and legacy systems. The ETL process involves extracting, transforming and loading data from source systems. The process may sound very simple since it only involves reading information from source databases, transforming it to fit the ODS database model and loading it in the ODS.

As different data sources tend to have different conventions for coding information and different standards for the quality of information, building an ODS requires data filtering, data cleaning, and integration.

The following examples show the importance of data cleaning:

- If an enterprise wishes to contact its customers or its suppliers, it is essential that a complete, accurate and up-to-date list of contact addresses, email addresses and telephone numbers be available. Correspondence sent to a wrong address that is then redirected does not create a very good impression about the enterprise.
- If a customer or supplier calls, the staff responding should be quickly ale to find the person in the enterprise database but this requires that the caller’s name or his/her company name is accurately listed in the database.
- If a customer appears in the databases with two or more slightly different names or different account numbers, it becomes difficult to update the customer’s information.

ETL requires skills in management, business analysis and technology and is often a significant component of developing an ODS or a data warehouse. The ETL process tends to be different for every ODS and data warehouse since every system is different. It should not be assumed that an off-the-shelf ETL system can magically solve all ETL problems.

ETL Functions

The ETL process consists of data extraction from source systems, data transformation which includes data cleaning, and loading data in the ODS or the data warehouse.

Transforming data that has been put in a staging area is a rather complex phase of ETL since a variety of transformations may be required. Large amounts of data from different sources are unlikely to match even if belonging to the same person since people using different conventions and different technology and different systems would have created records at different times in a different environment for different purposes. Building an integrated database from a number of such source systems may involve solving some or all of the following problems, some of which may be single-source problems while others may be multiple-source problems:

1. ***Instance identity problem:*** The same customer or client may be represented slightly different in different source systems. For example, my name is represented as Gopal Gupta in some systems and as GK Gupta in others. Given that the name is unusual for data entry staff in Western countries, it is sometimes misspelled as Gopal Gopta or Gopal Gupta or some other way. The name may also be represented as Professor GK Gupta, Dr GK Gupta or Mr GK Gupta. There is thus a possibility of mismatching between the different systems that needs to be identified and corrected.

2. ***Data errors:*** Many different types of data errors other than identity errors are possible. For example:
   - Data may have some missing attribute values.

- Coding of some values in one database may not match with coding in other databases (i.e. different codes with the same meaning or same code for different meanings)
- Meanings of some code values may not be known.
- There may be duplicate records.
- There may be wrong aggregations.
- There may be inconsistent use of nulls, spaces and empty values.
- Some attribute values may be inconsistent (i.e. outside their domain)
- Some data may be wrong because of input errors.
- There may be inappropriate use of address lines.
- There may be non-unique identifiers.

The ETL process needs to ensure that all these types of errors and others are resolved using a sound Technology.

3. ***Record linkage problem:*** Record linkage relates to the problem of linking information from different databases that relate to the same customer or client. The problem can arise if a unique identifier is not available in all databases that are being linked. Perhaps records from a database are being linked to records from a legacy system or to information from a spreadsheet. Record linkage can involve a large number of record comparisons to ensure linkages that have a high level of accuracy.

4. ***Semantic integration problem:*** This deals with the integration of information found in heterogeneous OLTP and legacy sources. Some of the sources may be relational, some may not be. Some may be even in text documents. Some data may be character strings while others may be integers.

5. ***Data integrity problem:*** This deals with issues like referential integrity, null values, domain of values, etc.

Overcoming all these problems is often a very tedious work. Many errors can be difficult to identify. In some cases one may be forced to ask the question how accurate the data ought to be since improving the accuracy is always going to require more and more resources and completely fixing all problems may be unrealistic.

Checking for duplicates is not always easy. The data can be sorted and duplicates removed although for large files this can be expensive. In some cases the duplicate records are not identical. In these cases checks for primary key may be required. If more than one record has the same primary key then it is likely to be because of duplicates.

A sound theoretical background is being developed for data cleaning techniques. It has been suggested that data cleaning should be based on the following five steps:

1. ***Parsing:*** Parsing identifies various components of the source data files and then establishes relationships between those and the fields in the target files. The classical example of parsing is identifying the various components of a person's name and address.
2. ***Correcting:*** Correcting the identified components is usually based on a variety of sophisticated techniques including mathematical algorithms. Correcting may involve use of other related information that may be available in the enterprise.
3. ***Standardizing:*** Business rules of the enterprise may now be used to transform the data to standard form. For example, in some companies there might be rules on how name and address are to be represented.
4. ***Matching:*** Much of the data extracted from a number of source systems is likely to be related. Such data needs to be matched.
5. ***Consolidating:*** All corrected, standardized and matched data can now be consolidated to build a single version of the enterprise data.

Selecting an ETL Tool

Selection of an appropriate ETL Tool is an important decision that has to be made in choosing components of an ODS or data warehousing application. The ETL tool is required to provide coordinated access to multiple data sources so that relevant data may

be extracted from them. An ETL tool would normally include tools for data cleansing, reorganization, transformation, aggregation, calculation and automatic loading of data into the target database.

An ETL tool should provide an easy user interface that allows data cleansing and data transformation rules to be specified using a point-and-click approach. When all mappings and transformations have been specified, the ETL tool should automatically generate the data extract/transformation/load programs, which typically run in batch mode.

DATA WAREHOUSES

Data warehousing is a process for assembling and managing data from various sources for the purpose of gaining a single detailed view of an enterprise. Although there are several definitions of data warehouse, a widely accepted definition by Inmon (1992) is *an integrated subject-oriented and time-variant repository of information in support of management's decision making process.* The definition of an ODS to except that an ODS is a current-valued data store while a data warehouse is a time-variant repository of data.

The benefits of implementing a data warehouse are as follows:

- To provide a single version of truth about enterprise information. This may appear rather obvious but it is not uncommon in an enterprise for two database systems to have two different versions of the truth. In many years of working in universities, I have rarely found a university in which everyone agrees with financial figures of income and expenditure at each reporting time during the year.
- To speed up ad hoc reports and queries that involve aggregations across many attributes (that is, may GROUP BY's) which are resource intensive. The managers require trends, sums and aggregations that allow, for example, comparing this year's performance to last year's or preparation of forecasts for next year.
- To provide a system in which managers who do not have a strong technical background are able to run complex queries. If the managers are able to access the

information they require, it is likely to reduce the bureaucracy around the managers.

- To provide a database that stores relatively clean data. By using a good ETL process, the data warehouse should have data of high quality. When errors are discovered it may be desirable to correct them directly in the data warehouse and then propagate the corrections to the OLTP systems.
- To provide a database that stores historical data that may have been deleted from the OLTP systems. To improve response time, historical data is usually not retained in OLTP systems other than that which is required to respond to customer queries. The data warehouse can then store the data that is purged from the OLTP systems.

A useful way of showing the relationship between OLTP systems, a data warehouse and an ODS is given in Figure 7.2. The data warehouse is more like long term memory of an enterprise. The objectives in building the two systems, ODS and data warehouse, are somewhat conflicting and therefore the two databases are likely to have different schemes.

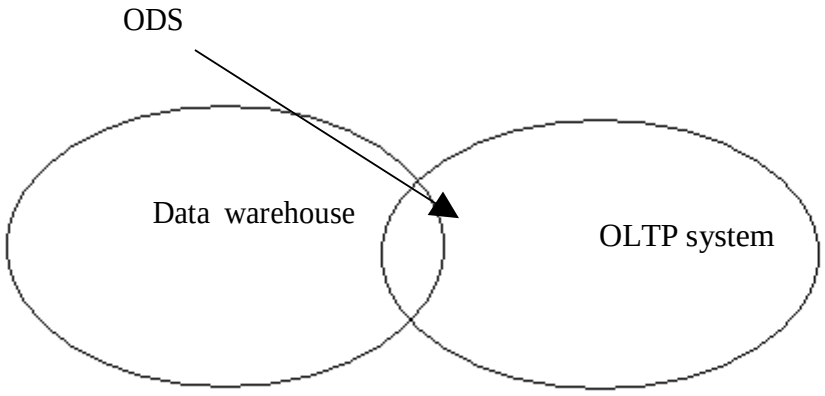


Figure 7.2 Relationship between OLTP, ODS and DW systems.

In building and ODS, data warehousing is a process of integrating enterprise-wide data, originating from a variety of sources, into a single repository. As shown in Figure 7.3, the data warehouse may be a central enterprise-wide data warehouse for use by all the decision makers in the enterprise or it may consist of a number of smaller data warehouse (often called data marts or local data warehouses)

A data mart stores information for a limited number of subject areas. For example, a company might have a data mart about marketing that supports marketing and sales. The data mart approach is attractive since beginning with a single data mart is relatively inexpensive and easier to implement.

A data mart may be used as a proof of data warehouse concept. Data marts can also create difficulties by setting up “silos of information” although one may build dependent data marts, which are populated form the central data warehouse.

Data marts are often the common approach for building a data warehouse since the cost curve for data marts tends to be more linear. A centralized data warehouse project can be very resource intensive and requires significant investment at the beginning although overall costs over a number of years for a centralized data warehouse and for decentralized data marts are likely to be similar.

A centralized warehouse can provide better quality data and minimize data inconsistencies since the data quality is controlled centrally. The tools and procedures for putting data in the warehouse can then be better controlled. Controlling data quality with a decentralized approach is obviously more difficult. As with any centralized function, though, the units or branches of an enterprise may feel no ownership of the centralized warehouse may in some cases not fully cooperate with the administration of the warehouse. Also, maintaining a centralized warehouse would require considerable coordination among the various units if the enterprise is large and this coordination may incur significant costs for the enterprise.

As an example of a data warehouse application we consider the telecommunications industry which in most countries has become very competitive during the last few years. If a company is able to identify a market trend before its competitors do, then that can lead to a competitive advantage. What is therefore needed is to analyse customer needs and behaviour in an attempt to better understand what the

customers want and need. Such understanding might make it easier for a company to

identify, develop, and deliver some relevant new products or new pricing schemes to

retain and attract customers. It can also help in improving profitability since it can help the company understand what type of customers are the most profitable.

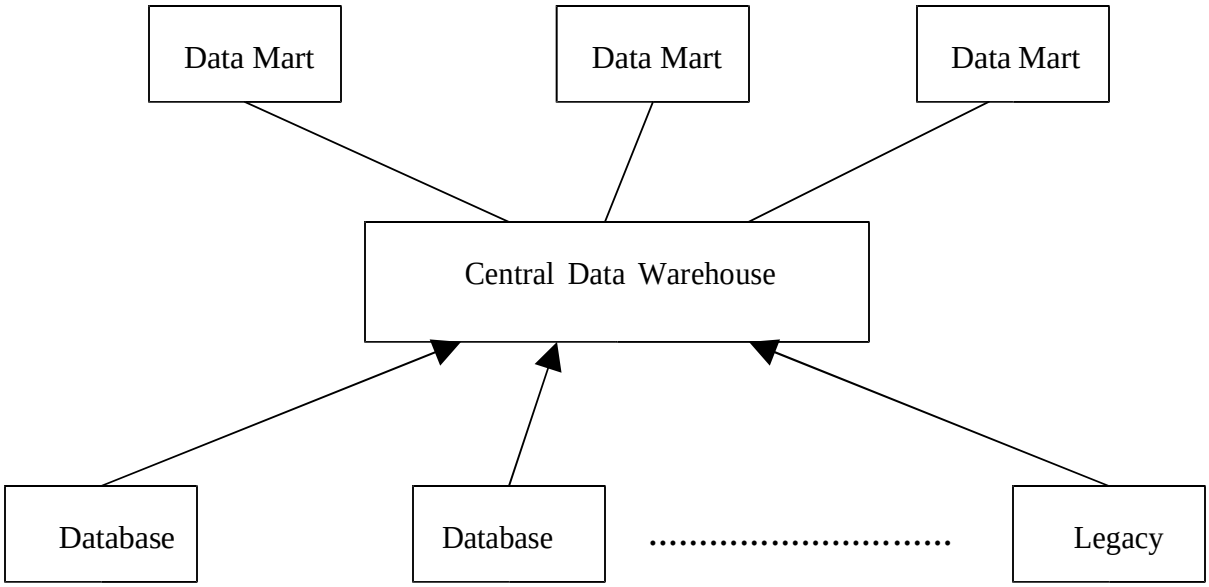


Figure 7.3 Simple structure of a data warehouse system.

ODS and DW Architecture

A typical ODS structure was shown in Figure 7.1. It involved extracting information from source systems by using ETL processes and then storing the information in the CICS ,Flat Files, Oracle

The ODS could then be used for producing a variety of reports for management.

ODS.

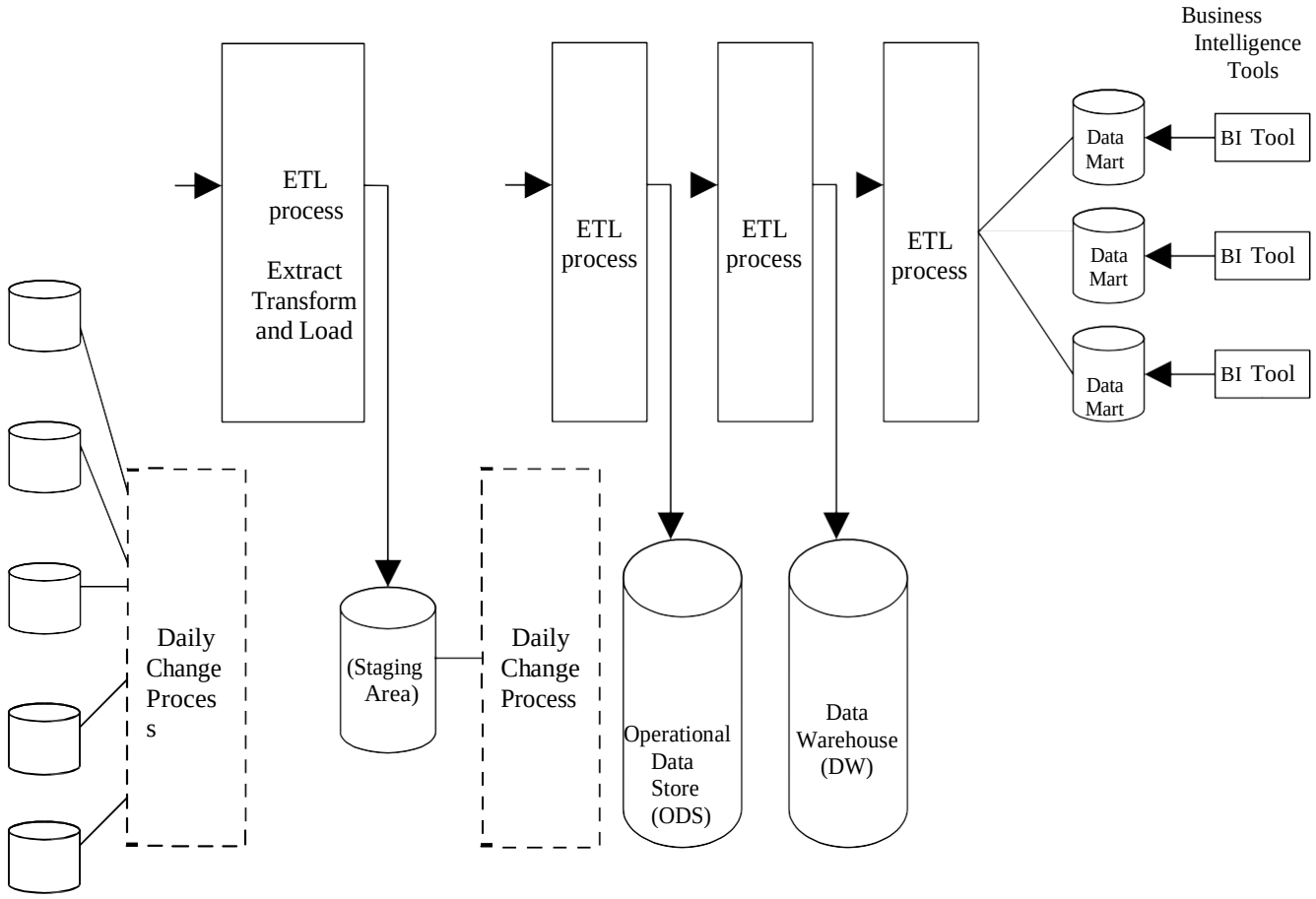


Figure 7.4 Another structure for ODS and DW

The architecture of a system that includes an ODS and a data warehouse shown in Figure 7.4 is more complex. It involves extracting information from source systems by using an ETL process and then storing the information in a staging database. The daily changes also come to the staging area. Another ETL process is used to transform information from the staging area to populate the ODS. The ODS is then used for supplying information via another ETL process to the area warehouse which in turn feeds a number of data marts that generate the reports required by management. It should be noted that not all ETL processes in this architecture involve data cleaning, some may only involve data extraction and transformation to suit the target systems.

DATA WAREHOUSE DESIGN

There are a number of ways of conceptualizing a data warehouse. One approach is to view it as a three-level structure. The lowest level consists of the OLTP and legacy systems, the middle level consists of the global or central data warehouse while the top level consists of local data warehouses or data marts. Another approach is possible if the enterprise has an ODS. The three levels then might consist of OLTP and legacy systems at the bottom, the ODS in the middle and the data warehouse at the top.

Whatever the architecture, a data warehouse needs to have a data model that can form the basis for implementing it. To develop a data model we view a data warehouse as a multidimensional structure consisting of dimensions, since that is an intuitive model that matches the types of OLAP queries posed by management. A dimension is an ordinate within a multidimensional structure consisting of a list of ordered values (sometimes called members) just like the x-axis and y-axis values on a two-dimensional graph.

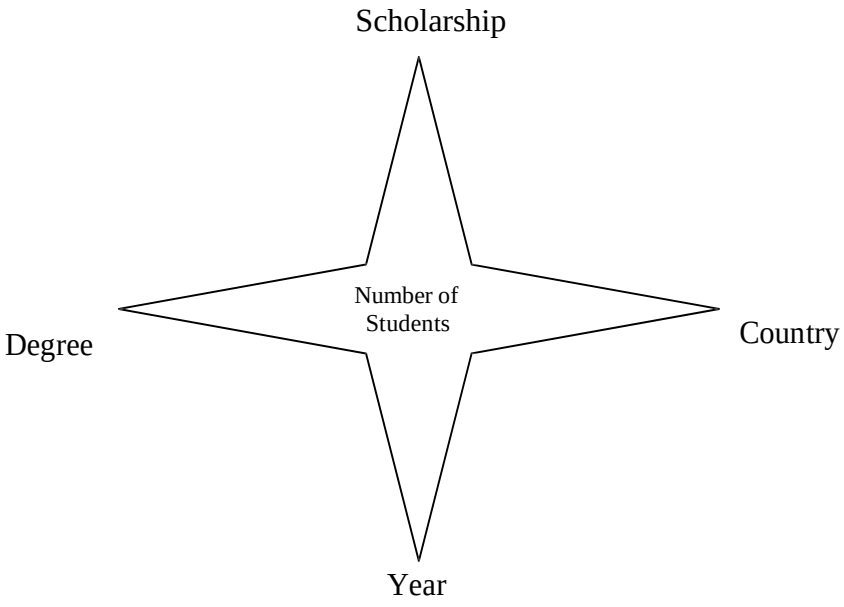


Figure 7.5 A simple example of a star schema.

A data warehouse model often consists of a central fact table and a set of surrounding dimension tables on which the facts depend. Such a model is called a star schema because of the shape of the model representation. A simple example of such a schema is shown in Figure 7.5 for a university where we assume that the number of students is given by the four dimensions – degree, year, country and scholarship. These four dimensions were chosen because we are interested in finding out how many students come to each degree program, each year, from each country under each scholarship scheme.

A characteristic of a star schema is that all the dimensions directly link to the fact table. The fact table may look like table 7.1 and the dimension tables may look Tables 7.2 to 7.5.

Table 7.1 An example of the fact table

Year	*Degree name*	*Country name*	*Scholarship name*	*Number*
200301	BSc	Australia	Govt	35
199902	MBBS	Canada	None	50
200002	LLB	USA	ABC	22
199901	BCom	UK	Commonwealth	7
200102	LLB	Australia	Equity	2

The first dimension is the degree dimension. An example of this dimension table is Table 7.2.

Table 7.2 An example of the degree dimension table

Name	*Faculty*	*Scholarship eligibility*	*Number of semesters*
BSc	Science	Yes	6
MBBS	Medicine	No	10
LLB	Law	Yes	8
BCom	Business	No	6
LLB	Arts	No	6

We now present the second dimension, the country dimension. An example of this dimension table is Table 7.3.

Table 7.3 An example of the country dimension table

Name	*Continent*	*Education Level*	*Major religion*
Nepal	Asia	Low	Hinduism
Indonesia	Asia	Low	Islam
Norway	Europe	High	Christianity
Singapore	Asia	High	NULL
Colombia	South America	Low	Christianity

The third dimension is the scholarship dimension. The dimension table is given in Table 7.4.

Table 7.4 An example of the scholarship dimension table

Name	*Amount (%)*	*Scholarship eligibility*	*Number*
Colombo	100	All	6
Equity	100	Low income	10
Asia	50	Top 5%	8
Merit	75	Top 5%	5
Bursary	25	Low income	12

The fourth dimension is the year dimension. The dimension table is given in Table 7.5.

Table 7.5 An example of the year dimension table

Name	*New programs*
2001	Journalism
2002	Multimedia

2003	Biotechnology

We now present further examples of the star schema. Figure 7.7 shows a star schema for a model with four dimensions.

Star schema may be refined into *snowflake schemas* if we wish to provide support for dimension hierarchies by allowing the dimension tables to have subtables to represent the hierarchies. For example, Figure 7.8 shows a simple snowflake schema for a two-dimensional example.

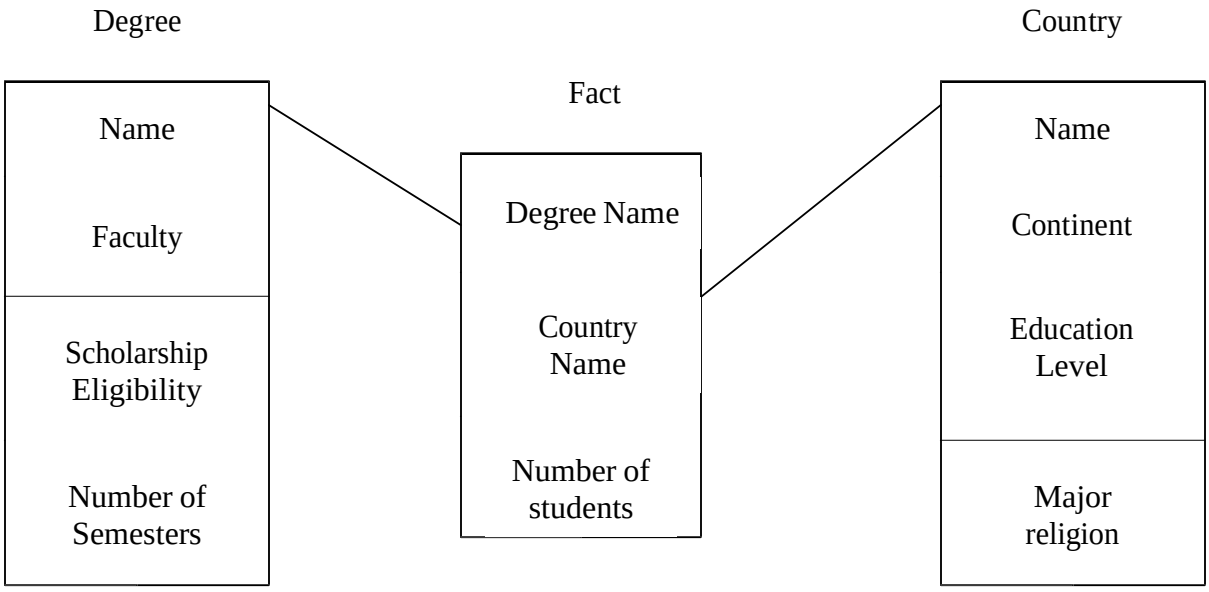


Figure 7.6 Star schema for a two-dimensional example.

The star and snowflake schemas are intuitive, easy to understand, can deal with aggregate data and can be easily extended by adding new attributes or new dimensions. They are the popular modeling techniques for a data warehouse. Entry-relationship modeling is often not discussed in the context of data warehousing although it is quite straightforward to look at the star schema as an ER model. Each dimension may be considered an entity and the fact may be considered either a relationship between the dimension entities or an

entity in which the primary key is the combination of the foreign keys that refer to the dimensions.

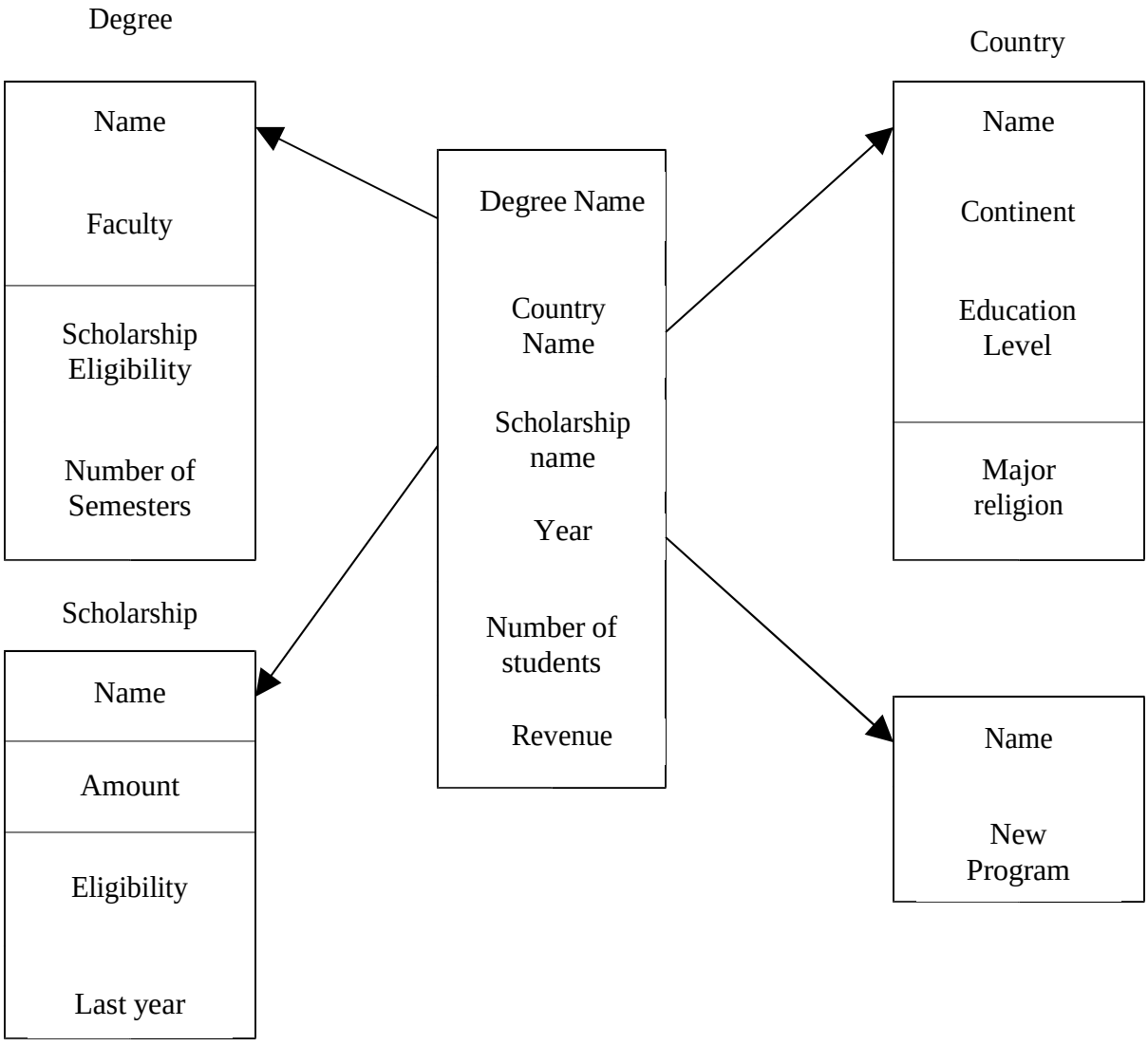


Figure 7.7 Star schema for a four-dimensional example.

The star and snowflake schemas are intuitive, easy to understand, can deal with aggregate data and can be easily extended by adding new attributes or new dimensions. They are the popular modeling techniques for a data warehouse. Entity-relationship modeling is

often not discussed in the context of data warehousing although it is quite straightforward to look at the star schema as an ER model.

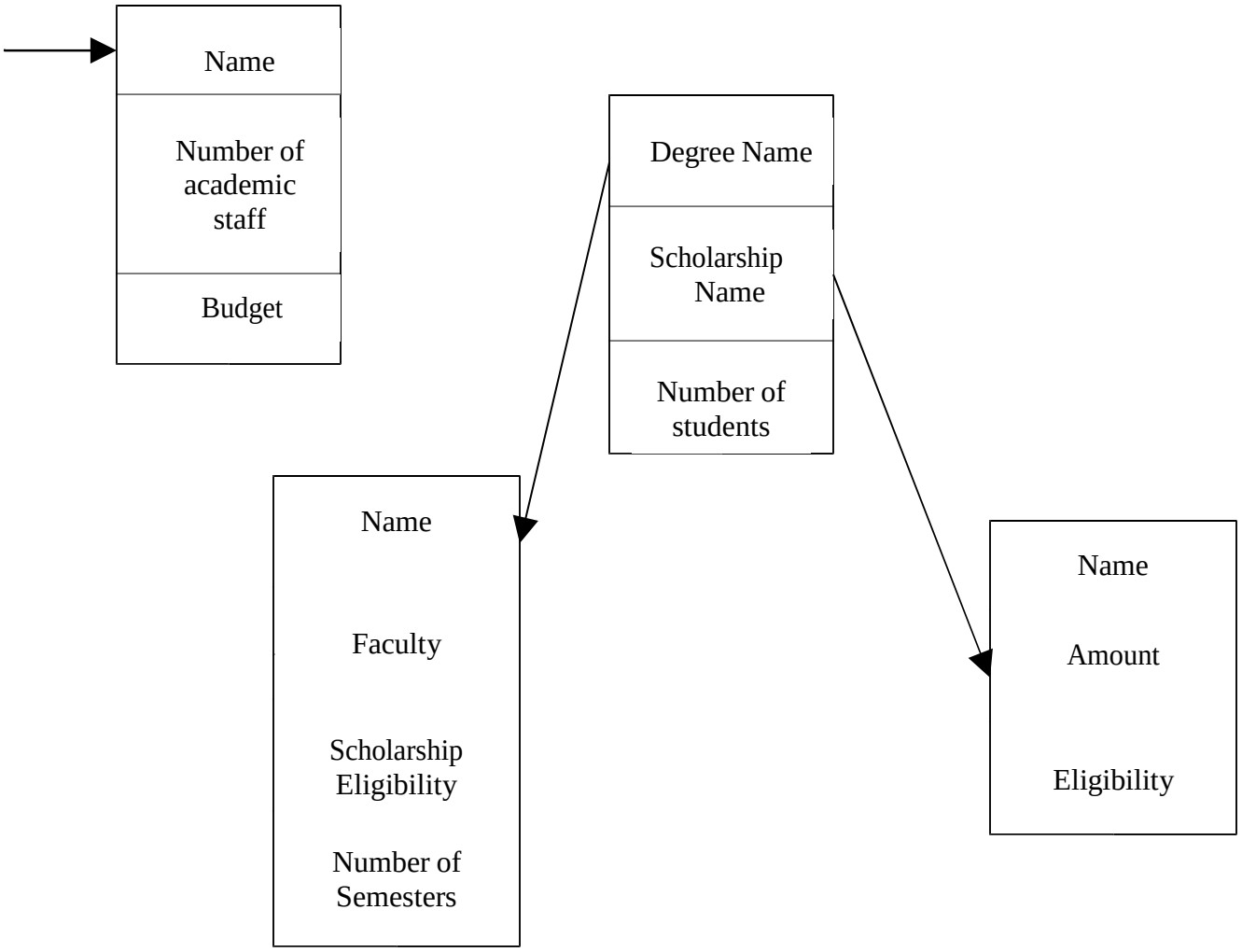


Figure 1.8 An example of a snowflake schema.

The dimensional structure of the star schema is called a multidimensional cube in online analytical processing (OALP). The cubes may be precomputed to provide very quick response to management OLAP queries regardless of the size of the data warehouse.

GUIDELINES FOR DATA WAREHOUSE IMPLEMENTATION

Implementation steps

1. ***Requirements analysis and capacity planning:*** In other projects, the first step in data warehousing involves defining enterprise needs, defining architecture, carrying out capacity planning and selecting the hardware and software tools. This step will involve consulting senior management as well as the various stakeholders.
2. ***Hardware integration:*** Once the hardware and software have been selected, they need to be put together by integrating the servers, the storage devices and the client software tools.
3. ***Modelling:*** Modelling is a major step that involves designing the warehouse schema and views. This may involve using a modelling tool if the data warehouse is complex.
4. ***Physical modelling:*** For the data warehouse to perform efficiently, physical modelling is required. This involves designing the physical data warehouse organization, data placement, data partitioning, deciding on access methods and indexing.
5. ***Sources:*** The data for the data warehouse is likely to come from a number of data sources. This step involves identifying and connecting the sources using gateways, ODBC drives or other wrappers.
6. ***ETL:*** The data from the source systems will need to go through an ETL process. The step of designing and implementing the ETL process may involve identifying a suitable ETL tool vendor and purchasing and implementing the tool. This may include customizing the tool to suit the needs of the enterprise.
7. ***Populate the data warehouse:*** Once the ETL tools have been agreed upon, testing the tools will be required, perhaps using a staging area. Once everything is

working satisfactorily, the ETL tools may be used in populating the warehouse given the schema and view definitions.

8. ***User applications:*** For the data warehouse to be useful there must be end-user applications. This step involves designing and implementing applications required by the end users.
9. ***Roll-out the warehouse and applications:*** Once the data warehouse has been populated and the end-user applications tested, the warehouse system and the applications may be rolled out for the user community to use.

Implementation Guidelines

1. ***Build incrementally:*** Data warehouses must be built incrementally. Generally it is recommended that a data mart may first be built with one particular project in mind and once it is implemented a number of other sections of the enterprise may also wish to implement similar systems. An enterprise data warehouse can then be implemented in an iterative manner allowing all data marts to extract information from the data warehouse. Data warehouse modelling itself is an iterative methodology as users become familiar with the technology and are then able to understand and express their requirements more clearly.
2. ***Need a champion:*** A data warehouse project must have a champion who is willing to carry out considerable research into expected costs and benefits of the project. Data warehousing projects require inputs from many units in am enterprise and therefore need to be driven by someone who is capable of interaction with people in the enterprise and can actively persuade colleagues. Without the cooperation of other units, the data model for the warehouse and the data required to populate the warehouse may be more complicated than they need to be. Studies have shown that having a champion can help adoption and success of data warehousing projects.
3. ***Senior management support:*** A data warehouse project must be fully supported by the senior management. Given the resource intensive nature of such projects and the time they can take to implement, a warehouse project calls for a sustained commitment from senior management. This can sometimes be difficult since it

may be hard to quantify the benefits of data warehouse technology and the managers may consider it a cost without any explicit return on investment. Data warehousing project studies show that top management support is essential for the success of a data warehousing project.

4. ***Ensure quality:*** Only data that has been cleaned and is of a quality that is understood by the organization should be loaded in the data warehouse. The data quality in the source systems is not always high and often little effort is made to improve data quality in the source systems. Improved data quality, when recognized by senior managers and stakeholders, is likely to lead to improved support for a data warehouse project.
5. ***Corporate strategy:*** A data warehouse project must fit with corporate strategy and business objectives. The objectives of the project must be clearly defined before the start of the project. Given the importance of senior management support for a data warehousing project, the fitness of the project with the corporate strategy is essential.
6. ***Business plan:*** The financial costs (hardware, software, peopleware), expected benefits and a project plan (including an ETL plan) for a data warehouse project must be clearly outlined and understood by all stakeholders. Without such understanding, rumours about expenditure and benefits can become the only source of information, undermining the project.
7. ***Training:*** A data warehouse project must not overlook data warehouse training requirements. For a data warehouse project to be successful, the users must be trained to use the warehouse and to understand its capabilities. Training of users and professional development of the project team may also be required since data warehousing is a complex task and the skills of the project team are critical to the success of the project.
8. ***Adaptability:*** The project should build in adaptability so that changes may be made to the data warehouse if and when required. Like any system, a data warehouse will need to change, as needs of an enterprise change. Furthermore, once the data warehouse is operational, new applications using the data

warehouse are almost certain to be proposed. The system should be able to support such new applications.

9. ***Joint management:*** The project must be managed by both IT and business professionals in the enterprise. To ensure good communication with the stakeholders and that the project is focused on assisting the enterprise's business, business professionals must be involved in the project along with technical professionals.

DATA WAREHOUSE METADATA

Given the complexity of information in an ODS and the data warehouse, it is essential that there be a mechanism for users to easily find out what data is there and how it can be used to meet their needs. Providing *metadata* about the ODS or the data warehouse achieves this. Metadata is data about data or documentation about the data that is needed by the users. Another description of metadata is that it is structured data which describes the characteristics of a resource. Metadata is stored in the system itself and can be queried using tools that are available on the system.

Several examples of metadata that should be familiar to the reader:

1. A library catalogue may be considered metadata. The catalogue metadata consists of a number of predefined elements representing specific attributes of a resource, and each element can have one or more values. These elements could be the name of the author, the name of the document, the publisher's name, the publication date and the category to which it belongs. They could even include an abstract of the data.
2. The table of contents and the index in a book may be considered metadata for the book.
3. Suppose we say that a data element about a person is 80. This must then be described by noting that it is the person's weight and the unit is kilograms. Therefore (weight, kilogram) is the metadata about the data 80.
4. Yet another example of metadata is data about the tables and figures in a document like this book. A table (which is data) has a name (e.g. table titles in

this chapter) and there are column names of the table that may be considered metadata. The figures also have titles or names.

There are many metadata standards. For example, the AGLS (Australian Government Locator Service) Metadata standard is a set of 19 descriptive elements which Australian government departments and agencies can use to improve the visibility and accessibility of their services and information over the Internet.

In a database, metadata usually consists of table (relation) lists, primary key names, attributes names, their domains, schemas, record counts and perhaps a list of the most common queries. Additional information may be provided including logical and physical data structures and when and what data was loaded.

In the context of a data warehouse, metadata has been defined as "all of the information in the data warehouse environment that is not the actual data itself".

In the data warehouse, metadata needs to be much more comprehensive. It may be classified into two groups: back room metadata and front room metadata. Much important information is included in the back room metadata that is process related and guides, for example, the ETL processes.

SOFTWARE FOR ODS, ZLE, ETL AND DATA WAREHOUSING

ODS Software

- IQ Solutions: Dynamic ODS from Sybase offloads data from OLTP systems and makes if available on a Sybase IQ platform for queries and analysis.

- ADH Active Data Hub from Glenridge Solutions is a real-time data integration and reporting solution for PeopleSoft, Oracle and SAP databases. ADH includes an ODS, an enterprise data warehouse, a workflow initiator and a meta library.

ZLE Software

HP ZLE framework based on the HP NonStop platform combines application and data integration to create an enterprise-wide solution for real-time information. The ZLE solution is targeted at retail, telecommunications, healthcare, government and finance.

ETL Software

- Aradyme Data Services from Aradyme Corporation provides data migration services for extraction, cleaning, transformation and loading from any source to any destination. Aradyme claims to minimize the risks inherent in many-to-one, many-to-many and similar migration projects.

- DataFlux from a company with the same name (acquired by SAS in 2000) provides solutions that help inspect, correct, integrate, enhance, and control data. Solutions include data

- Dataset V from Intercon Systems Inc is an integrated suite for data cleaning, matching, positive identification, de-duplication and statistical analysis.

- WinPure List Cleaner Pro from WinPure provides a suite consisting of eight modules that clean, correct unwanted punctuation and spelling errors, identify missing data via graphs and a scoring system and removes duplicates from a variety of data sources.

Data Warehousing Software

- mySAP Business Intelligence provides facilities of ETL, data warehouse management and business modelling to help build data warehouse, model information architecture and manage data from multiple sources.

- SQL Server 2005 from Microsoft provides ETL tools as well as tools for building a relational data warehouse and a multidimensional database.

- Sybase IQ is designed for reporting, data warehousing and analytics. It claims to deliver high query performance and storage efficiency for structured and unstructured data. Sybase has partnered with Sun in providing data warehousing solutions.

UNIT 2

ONLINE ANALYTICAL PROCESSING (OLAP)

INTRODUCTION

A *dimension* is an attribute or an ordinate within a multidimensional structure consisting of a list of values (members). For example, the degree, the country, the scholarship and the year were the four dimensions used in the student database. Dimensions are used for selecting and aggregating data at the desired level. A dimension does not include ordering of values, for example there is no ordering associated with values of each of the four dimensions, but a dimension may have one or more hierarchies that show parent /child relationship between the members of a dimension.

For example, the dimension country may have a hierarchy that divides the world into continents and continents into regions followed by regions into countries if such a hierarchy is useful for the applications. Multiple hierarchies may be defined on a dimension. For example, counties may be defined to have a geographical hierarchy and may have another hierarchy defined on the basis of their wealth or per capita income (e.g. high, medium, low).

The non-null values of facts are the numerical values stored in each data cube cell. They are called *measures.* A measure is a non-key attribute in a fact table and the value of the measure is dependent on the values of the dimensions. Each unique combination of

members in a Cartesian product dimensions of the cube identifies precisely one data cell within the cube and that cell stores the values of the measures.

The SQL command GROUP BY is unusual aggregation operator in that a table is divided into sub-tables based on the attribute values in the GROUP BY clause so that each sub-table has the same values for the attribute and then aggregations over each sub-table are carried out. SQL has a variety of aggregation functions including max, min, average, count which are used by employing the GROUP BY facility.

A data cube computes aggregates overall subsets of dimensions specified in the cube. A cube may be found at the union of (large) number of SQL GROUP-BY operations. Generally, all or some of the aggregates are pre-computed to improve query response time. A decision has to be made as to what and how much should be pre-computed since pre-computed queries require storage and time to compute them.

A data cube is often implemented as a database in which there are dimension tables each of which provides details of a dimension. The database may be the enterprise data warehouse.

OLAP

OLAP systems are data warehouse front-end software tools to make aggregate data available efficiently, for advanced analysis, to managers of an enterprise. The analysis often requires resource intensive aggregations processing and therefore it becomes necessary to implement a special database (e.g. data warehouse) to improve OLAP response time. It is essential that an OLAP system provides facilities for a manager to pose ad hoc complex queries to obtain the information that he/she requires.

Another term that is being used increasingly is *business intelligence*. It is used to mean both data warehousing and OLAP. It has been defined as a user-centered process of exploring data, data relationships and trends, thereby helping to improve overall decision making. Normally this involves a process of accessing data (usually stored within the data warehouse) and analyzing it to draw conclusions and derive insights with the

purpose of effecting positive change within an enterprise. Business intelligence is closely related to OLAP.

A data warehouse and OLAP are based on a multidimensional conceptual view of the enterprise data. Any enterprise data is multidimensional consisting of dimensions degree, country, scholarship, and year. Data that is arranged by the dimensions is like a spreadsheet, although a spreadsheet presents only two-dimensional data with each cell containing an aggregation. As an example, table 8.1 shows one such two-dimensional spreadsheet with dimensions Degree and Country, where the measure is the number of students joining a university in a particular year or semester.

Table 8.1 A multidimensional view of data for two dimensions

Degree / Country	c	LLB	MBBS	BCom	BIT	ALL
Australia	5	20	15	50	11	101
India	10	0	15	25	17	67
Malaysia	5	1	10	12	23	51
Singapore	2	2	10	10	31	55
Sweden	5	0	5	25	7	42
UK	5	15	20	20	13	73
USA	0	2	20	15	19	56
ALL	32	40	95	157	121	445

Table 8.1 be the information for the year 2001. Similar spreadsheet views would be available for other years. Three-dimensional data can also be organized in a spreadsheet using a number of sheets or by using a number of two-dimensional tables in the same sheet.

Although it is useful to think of OLAP systems as a generalization of spreadsheets, spreadsheets are not really suitable for OLAP in spite of the nice user-friendly interface that they provide. Spreadsheets tie data storage too tightly to the

presentation. It is therefore difficult to obtain other desirable views of the information. Furthermore it is not possible to query spreadsheets. Also, spreadsheets become unwieldy when more than three dimensions are to be represented. It is difficult to imagine a spreadsheet with millions of rows or with thousands of formulas. Even with small spreadsheets, formulas often have errors. An error-free spreadsheet with thousands of formulas would therefore be very difficult to build. Data cubes essentially generalize spreadsheets to any number of dimensions.

OLAP is the dynamic enterprise analysis required to create, manipulate, animate and synthesize information from exegetical, contemplative and formulaic data analysis models.

Essentially what this definition means is that the information is manipulated from the point if view of a manager (exegetical), from the point of view of someone who has thought about it(contemplative) and according to some formula(formulaic).

Another definition of OLAP, which is software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that, has been transformed from raw data to reflect that real dimensional of the enterprise as understood by the user.

An even simpler definition is that OLAP is a fast analysis of shared multidimensional information for advanced analysis. This definition (sometimes called FASMI) implies that most OLAP queries should be answered within seconds. Furthermore, it is expected that most OLAP queries can be answered without any programming.

In summary, a manager would want even the most complex query to be answered quickly; OLAP is usually a multi-user system that may be run on a separate server using specialized OLAP software. The major OLAP applications are trend analysis over a

number of time periods, slicing, dicing, drill-down and roll-up to look at different levels of detail and pivoting or rotating to obtain a new multidimensional view.

CHARACTERISTICS OF OLAP SYSTEMS

The following are the differences between OLAP and OLTP systems.

1. Users: OLTP systems are designed for office workers while the OLAP systems are designed for decision makers. Therefore while an OLTP system may be accessed by hundreds or even thousands of users in a large enterprise, an OLAP system is likely to be accessed only by a select group of managers and may be used only by dozens of users.

2. Functions: OLTP systems are mission-critical. They support day-to-day operations of an enterprise and are mostly performance and availability driven. These systems carry out simple repetitive operations. OLAP systems are management-critical to support decision of an enterprise support functions using analytical investigations. They are more functionality driven. These are ad hoc and often much more complex operations.

3. Nature: Although SQL queries often return a set of records, OLTP systems are designed to process one record at a time, for example a record related to the customer who might be on the phone or in the store. OLAP systems are not designed to deal with individual customer records. Instead they involve queries that deal with many records at a time and provide summary or aggregate data to a manager. OLAP applications involve data stored in a data warehouse that has been extracted from many tables and perhaps from more than one enterprise database.

4. Design: OLTP database systems are designed to be application-oriented while OLAP systems are designed to be subject-oriented. OLTP systems view the enterprise data as a collection of tables (perhaps based on an entity-relationship model). OLAP systems view enterprise information as multidimensional).

5. Data: OLTP systems normally deal only with the current status of information. For example, information about an employee who left three years ago may not be available

on the Human Resources System. The old information may have been achieved on some type of stable storage media and may not be accessible online. On the other hand, OLAP systems require historical data over several years since trends are often important in decision making.

6. Kind of use: OLTP systems are used for reading and writing operations while OLAP systems normally do not update the data.

The differences between OLTP and OLAP systems are:

Property	OLTP	OLAP
Nature of users	Operations workers	Decision makers
Functions	Mission-critical	Management-critical
Nature of queries	Mostly simple	Mostly complex
Nature of usage	Mostly repetitive	Mostly ad hoc
Nature of design	Application oriented	Subject oriented
Number of users	Thousands	Dozens
Nature of data	Current, detailed, relational	Historical, summarized, multidimensional
Updates	All the time	Usually not allowed

Table 8.1 Comparison of OLTP and OLAP system

FASMI Characteristics

In the FASMI characteristics of OLAP systems, the name derived from the first letters of the characteristics are:

Fast: As noted earlier, most OLAP queries should be answered very quickly, perhaps within seconds. The performance of an OLAP system has to be like that of a search engine. If the response takes more than say 20 seconds, the user is likely to move away to something else assuming there is a problem with the query. Achieving such performance is difficult. The data structures must be efficient. The hardware must be powerful enough for the amount of data and the number of users. Full pre-computation of aggregates helps but is often not practical due to the large number of aggregates. One

approach is to pre-compute the most commonly queried aggregates and compute the remaining on-the-fly.

Analytic: An OLAP system must provide rich analytic functionality and it is expected that most OLAP queries can be answered without any programming. The system should be able to cope with any relevant queries for the application and the user. Often the analysis will be using the vendor's own tools although OLAP software capabilities differ widely between products in the market.

Shared: An OLAP system is shared resource although it is unlikely to be shared by hundreds of users. An OLAP system is likely to be accessed only by a select group of managers and may be used merely by dozens of users. Being a shared system, an OLAP system should be provide adequate security for confidentiality as well as integrity.

Multidimensional: This is the basic requirement. Whatever OLAP software is being used, it must provide a multidimensional conceptual view of the data. It is because of the multidimensional view of data that we often refer to the data as a cube. A dimension often has hierarchies that show parent / child relationships between the members of a dimension. The multidimensional structure should allow such hierarchies.

Information: OLAP systems usually obtain information from a data warehouse. The system should be able to handle a large amount of input data. The capacity of an OLAP system to handle information and its integration with the data warehouse may be critical.

Codd's OLAP Characteristics

Codd et al's 1993 paper listed 12 characteristics (or rules) OLAP systems. Another six in 1995 followed these. Codd restructured the 18 rules into four groups. These rules provide another point of view on what constitutes an OLAP system.

All the 18 rules are available at http://www.olapreport.com/fasmi.htm. Here we discuss 10 characteristics, that are most important.

1. Multidimensional conceptual view: As noted above, this is central characteristic of an OLAP system. By requiring a multidimensional view, it is possible to carry out operations like slice and dice.

2. Accessibility (OLAP as a mediator): The OLAP software should be sitting between data sources (e.g data warehouse) and an OLAP front-end.

3. Batch extraction vs interpretive: An OLAP system should provide multidimensional data staging plus precalculation of aggregates in large multidimensional databases.

4. Multi-user support: Since the OLAP system is shared, the OLAP software should provide many normal database operations including retrieval, update, concurrency control, integrity and security.

5. Storing OLAP results: OLAP results data should be kept separate from source data. Read-write OLAP applications should not be implemented directly on live transaction data if OLAP source systems are supplying information to the OLAP system directly.

6. Extraction of missing values: The OLAP system should distinguish missing values from zero values. A large data cube may have a large number of zeros as well as some missing values. If a distinction is not made between zero values and missing values, the aggregates are likely to be computed incorrectly.

7. Treatment of missing values: An OLAP system should ignore all missing values regardless of their source. Correct aggregate values will be computed once the missing values are ignored.

8. Uniform reporting performance: Increasing the number of dimensions or database size should not significantly degrade the reporting performance of the OLAP system. This is a good objective although it may be difficult to achieve in practice.

9. Generic dimensionality: An OLAP system should treat each dimension as equivalent in both is structure and operational capabilities. Additional operational capabilities may be granted to selected dimensions but such additional functions should be grantable to any dimension.

10. Unlimited dimensions and aggregation levels: An OLAP system should allow unlimited dimensions and aggregation levels. In practice, the number of dimensions is rarely more than 10 and the number of hierarchies rarely more than six.

MOTIVATIONS FOR USING OLAP

1. Understanding and improving sales: For an enterprise that has many products and uses a number of channels for selling the products, OLAP can assist in

finding the most popular products and the most popular channels. In some cases it may be possible to find the most profitable customers. For example, considering the telecommunications industry and only considering one product, communication minutes, there is a large amount of data if a company wanted to analyze the sales of product for every hour of the day (24 hours), differentiate between weekdays and weekends (2 values) and divide regions to which calls are made into 50 regions.

2. Understanding and reducing costs of doing business: Improving sales is one aspect of improving a business, the other aspect is to analyze costs and to control them as much as possible without affecting sales. OLAP can assist in analyzing the costs associated with sales. In some cases, it may also be possible to identify expenditures that produce a high return on investment (ROI). For example, recruiting a top salesperson may involve significant costs, but the revenue generated by the salesperson may justify the investment.

MULTIDIMENSIONAL VIEW AND DATA CUBE

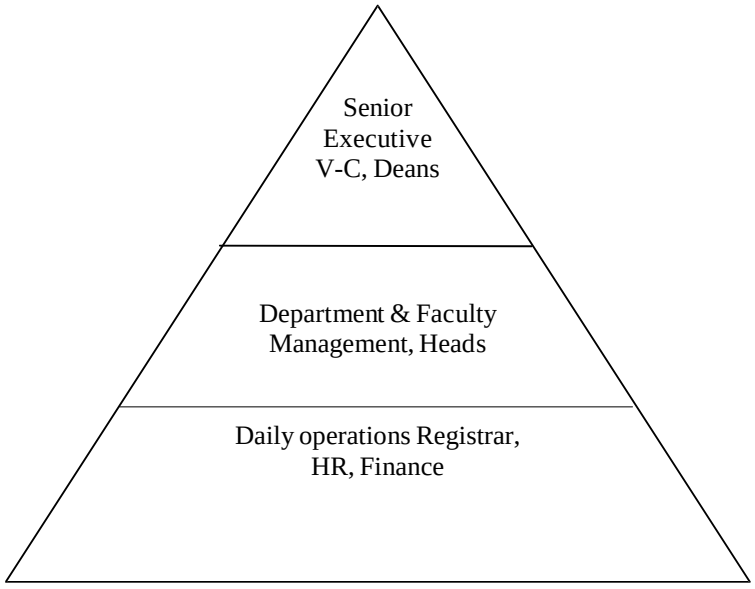


Figure 2.1 A typical University management hierarchy

The multidimensional view of data is in some ways natural view of any enterprise of managers. The triangle diagram in Figure 8.1 shows that as we go higher in the triangle hierarchy the managers need for detailed information declines.

The multidimensional view of data by using an example of a simple OLTP database consists of the three tables. Much of the literature on OLAP uses examples of a shoe store selling different colour shoes of different styles.

It should be noted that the relation enrolment would normally not be required since the degree a student is enrolled in could be included in the relation student but some students are enrolled in double degrees and so the relation between the student and the degree is multifold and hence the need for the relation enrolment.

student(Student_id, Student_name, Country, DOB, Address)

enrolment(Student_id, Degree_id, SSemester)

degree(Degree_id, Degree_name, Degree_length, Fee, Department)

An example of the first relation, i.e. student, is given in **Table 2.2**

Student_id	Student_name	Country	DOB	Address
8656789	Peta Williams	Australia	1/1/1980	Davis Hall
8700020	John Smith	Canada	2/2/1981	9 Davis Hall
8900020	Arun Krishna	USA	3/3/1983	90 Second Hall
8801234	Peter Chew	UK	4/4/1983	88 Long Hall
8654321	Reena Rani	Australia	5/5/1984	88 Long Hall
8712374	Kathy Garcia	Malaysia	6/6/1980	88 Long Hall
8612345	Chris Watanabe	Singapore	7/7/1981	11 Main street
8744223	Lars Anderssen	Sweden	8/8/1982	Null
8977665	Sachin Singh	UAE	9/9/1983	Null
9234567	Rahul Kumar	India	10/10/1984	Null
9176543	Saurav Gupta	UK	11/11/1985	1, Captain Drive

Table 8.3 presents an example of the relation enrolment. In this table, the attribute SSemester in the semester in which the student started the current degree. We code it by using the year followed by 01 for the first semester and 02 for the second. We assume that new students are admitted in each semester. Table 8.4 is an example of the relation degree. In this table, the degree length is given in terms of the number of semester it normally takes to finish it. The fee is assumed to be in thousands of dollars per year.

Table 2.3 The relation enrolment

Student_id	Degree_id	SSemester
8900020	1256	2002-01
8700074	3271	2002-01
8700074	3321	2002-02
8900020	4444	2000-01
8801234	1256	2000-01
8801234	3321	1999-02
8801234	3333	1999-02
8977665	3333	2000-02

Table 2.4 The relation degree

Degree_id	Degree_name	Degree_length	Fee	Department
1256	BIT	6	18	Computer Sci.
2345	BSc	6	20	Computer Sci
4325	BSc	6	20	Chemistry
3271	BSc	6	20	Physics
3321	BCom	6	16	Business
4444	MBBS	12	30	Medicine
3333	LLB	8	22	Law

It is clear that the information given in Tables 8.2, 8.3 and 8.4, although suitable for a student enrolment OLTP system, is not suitable for efficient management decision making. The managers do not need information about the individual students, the degree they are enrolled in, and the semester they joined the university. What the managers need is the trends in student numbers in different degree programs and from different countries.

We first consider only two dimensions. Let us say we are primarily interested in finding out how many students from each country came to do a particular degree. Therefore we may visualize the data as two-dimensional, i.e.,

Country x Degree

A table that summarizes this type of information may be represented by a two-dimensional spreadsheet given in Table 8.5 (the numbers in Table 8.5 do not need relate to the numbers in Table 8.3). We may call that this table gives the number of students admitted (in say, 2000-01) a two-dimensional "cube".

Table 2.5 A two-dimensional table of aggregates for semester 2000-01

Country \ Degree	BSc	LLB	MBBS	BCom	BIT	ALL
Australia	5	20	15	50	11	101
India	10	0	15	25	17	67
Malaysia	5	1	10	12	23	51
Singapore	2	2	10	10	31	55
Sweden	5	0	5	25	7	42
UK	5	15	20	20	13	73
USA	0	2	20	15	19	56
ALL	32	40	95	157	121	445

Using this two-dimensional view we are able to find the number of students joining any degree from any country (only for semester 2000-01). Other queries that we are quickly able to answer are:

- How many students started BIT in 2000-01?
- How many students joined from Singapore in 2000-01?

The data given in Table 8.6 is for a particular semester, 2000-01. A similar table would be available for other semesters. Let us assume that the data for 2001-01 is given in Table 8.7.

Table 2.6 A two-dimensional table of aggregates for semester 2001-01

Country \ Degree	BSc	LLB	MBBS	BCom	BIT	ALL
Australia	7	10	16	53	10	96
India	9	0	17	22	13	61
Malaysia	5	1	19	19	20	64
Singapore	2	2	10	12	23	49
Sweden	8	0	5	16	7	36
UK	4	13	20	26	11	74
USA	4	2	10	10	12	38
ALL	39	28	158	158	96	418

Let us now imagine that Table 8.6 is put on top of Table 8.5. We now have a three-dimensional cube with SSemester as the vertical dimension. We now put on top of these two tables another table that gives the vertical sums, as shown in Table 8.7.

Table 2.7 Two-dimensional table of aggregates for both semesters

Country \ Degree	BSc	LLB	MBBS	BCom	BIT	ALL
Australia	12	30	31	103	21	197
India	19	0	32	47	30	128
Malaysia	10	2	29	31	43	115
Singapore	4	4	20	22	54	104
Sweden	13	0	10	41	14	78
UK	9	28	40	46	24	147
USA	4	4	30	25	31	94
ALL	71	68	192	315	217	863

Tables 8.5, 8.6 and 8.7 together now form a three-dimensional cube. The table 8.7 provides totals for the two semesters and we are able to "drill-down" to find numbers in individual semesters. Note that a cube does not need to have an equal number of members in each dimension. Putting the three tables together gives a cube of 8 x 6 x 3 (= 144) cells including the totals along every dimension.

A cube could be represented by:

Country x Degree x Semester

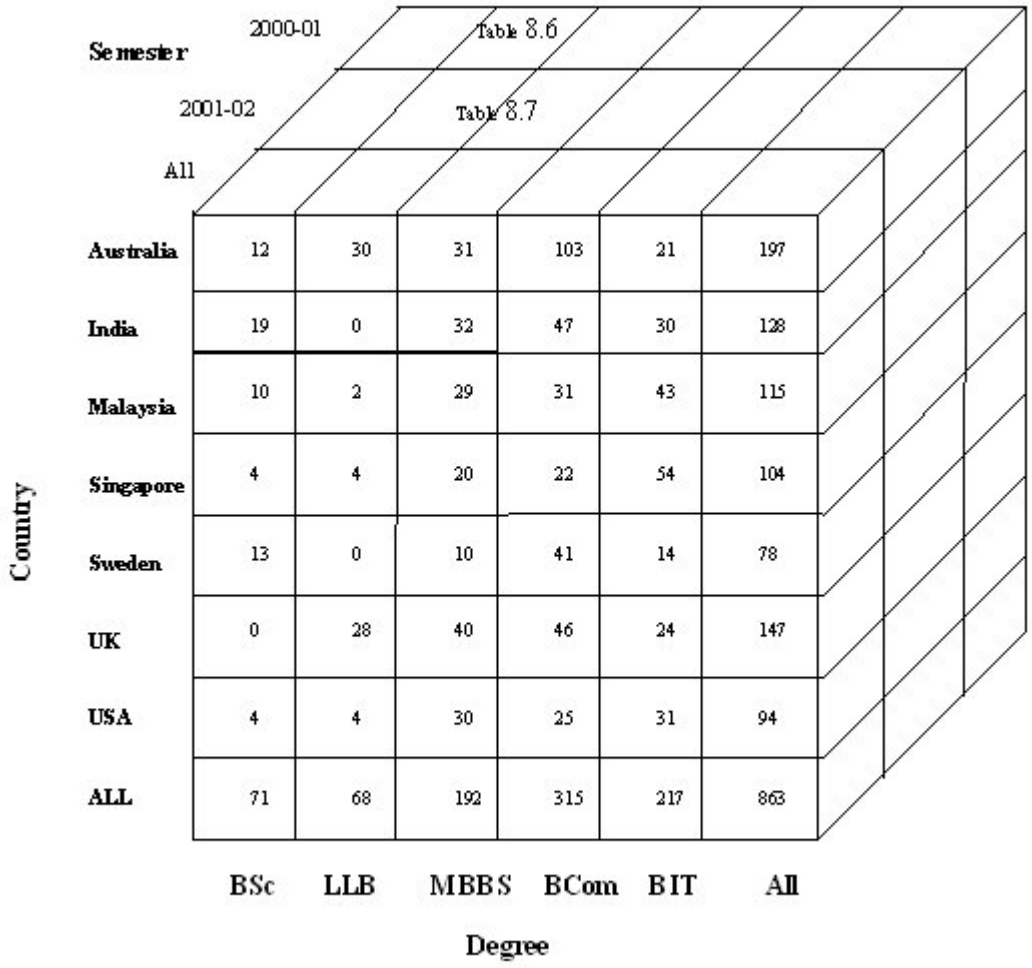


Figure 2.2 The cube formed by Tables 8.6, 8.7 and 8.8

In the three-dimensional cube, the following eight types of aggregations or queries are possible:

1. null (e.g. how many students are there? Only 1 possible query)
2. degrees (e.g. how many students are doing BSc? 5 possible queries if we assume 5 different degrees)

3. semester (e.g. how many students entered in semester 2000-01? 2 possible queries if we only have data about 2 semesters)
4. country (e.g. how many students are from the USA? 7 possible queries if there are 7 countries)
5. degrees, semester (e.g. how many students entered in 2000-01 to enroll in BCom? With 5 degrees and 2 different semesters 10 queries)
6. (ALL, b, c) semester, country (e.g. how many students from the UK entered in 2000-01? 14 queries)
7. (a, b, ALL) degrees, country (e.g. how many students from Singapore are enrolled in BCom? 35 queries)
8. (a, b, c) all (e.g. how many students from Malaysia entered in 2000-01 to enroll in BCom? 70 queries)

DATA CUBE IMPLEMENTATIONS

1. ***Pre-compute and store all:*** This means that millions of aggregates will need to be computed and stored. Although this is the best solution as far as query response time is concerned, the solution is impractical since resources required to compute the aggregates and to store them will be prohibitively large for a large data cube. Indexing large amounts of data is also expensive.

2. ***Pre-compute (and store) none:*** This means that the aggregates are computed on-the-fly using the raw data whenever a query is posed. This approach does not require additional space for storing the cube but the query response time is likely to be very poor for large data cubes.

3. ***Pre-compute and store some:*** This means that we pre-compute and store the most frequently queried aggregates and compute others as the need arises. We may also be able to derive some of the remaining aggregates using the aggregates that have already been computed. It may therefore be worthwhile also to pre-

compute some aggregates that are not most frequently queried but help in deriving many other aggregates. It will of course not be possible to derive all the aggregates from the pre-computed aggregates and it will be necessary to access the database (e.g the data warehouse) to compute the remaining aggregates. The more aggregates we are able to pre-compute the better the query performance.

It can be shown that large numbers of cells do have an "ALL" value and may therefore be derived from other aggregates. Let us reproduce the list of queries we had and define them as (a, b, c) where a stands for a value of the degree dimension, b for country and c for starting semester:

1. (ALL, ALL, ALL) null (e.g. how many students are there? Only 1 query)
2. (a, ALL, ALL) degrees (e.g. how many students are doing BSc? 5 queries)
3. (ALL, ALL, c) semester (e.g. how many students entered in semester 2000-01? 2 queries)
4. (ALL, b, ALL) country (e.g. how many students are from the USA? 7 queries)
5. (a, ALL, c) degrees, semester (e.g. how many students entered in 2000-01 to enroll in BCom? 10 queries)
6. (ALL, b, c) semester, country (e.g. how many students from the UK entered in 2000-01? 14 queries)
7. (a, b, ALL) degrees, country (e.g. how many students from Singapore are enrolled in BCom? 35 queries)
8. (a, b, c) all (e.g. how many students from Malaysia entered in 2000-01 to enroll in BCom? 70 queries)

It is therefore possible to derive the other 74 of the 144 queries from the last 70 queries of type (a, b, c). Of course in a very large data cube, it may not be practical even to pre-compute all the (a, b, c) queries and decision will need to be made which ones should be pre-computed given that storage availability may be limited and it may be required to minimize the average query cost.

In Figure 8.3 we show how the aggregated above are related and how an aggregate at the higher level may be computed from the aggregates below. For example, aggregates (ALL, ALL, c) may be derived from either (a, ALL, c) by summing over all a values from (ALL, b, c) by summing over all b values.

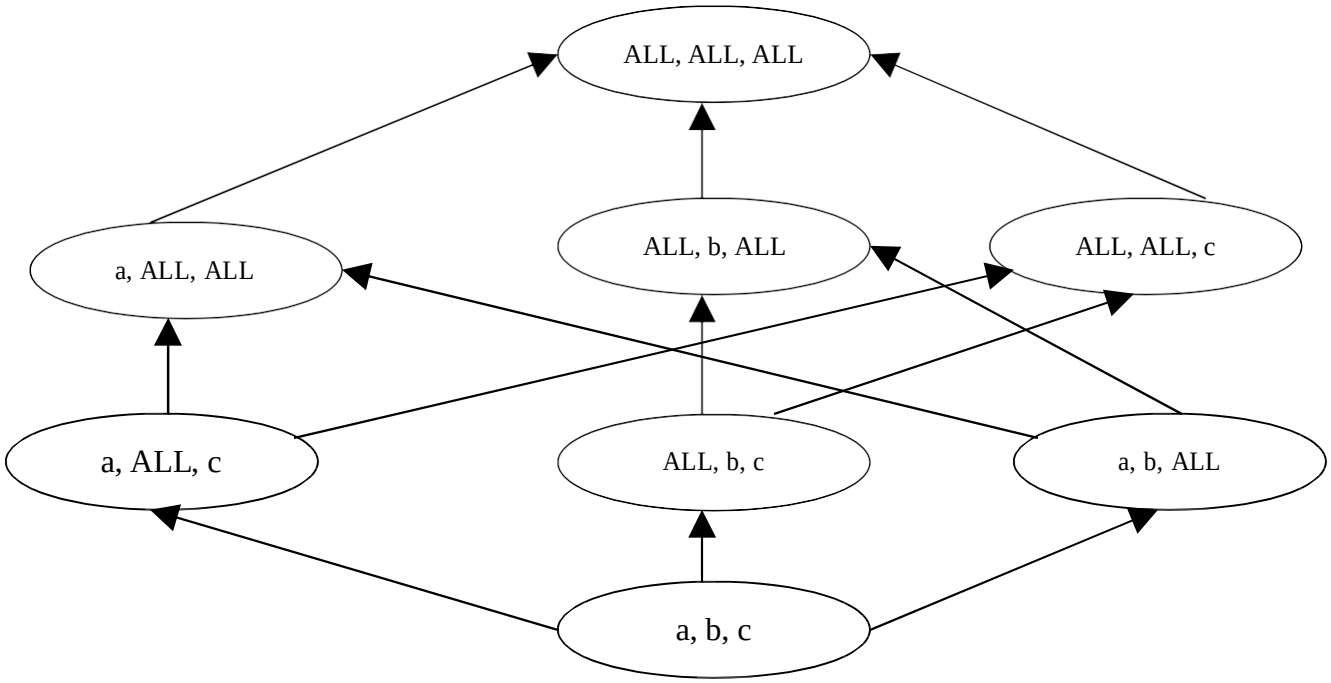


Figure 2.3 Relationships between aggregations of a three-dimensional cube

Another related issue is where the data used by OLAP will reside. We assume that the data is stored in a data warehouse or in one or more data marts.

Data cube products use different techniques for pre-computing aggregates and storing them. They are generally based on one of two implementation models. The first model, supported by vendors of traditional relational model databases, is called the ROLAP model or the Relational OLAP model. The second model is called the MOLAP model for multidimensional OLAP. The MLOAP model provides a direct

multidimensional view of the data whereas the RLOAP model provides a relational view of the multidimensional data in the form of a fact table.

ROLAP

ROLAP uses a relational DBMS to implement an OLAP environment. It may be considered a bottom-up approach which is typically based on using a data warehouse that has been designed using a star schema. The data therefore is likely to be in a denormalized structure. A normalized database avoids redundancy but is usually not appropriate for high performance. The summary data will be held in aggregate tables. The data warehouse provides the multidimensional capabilities by representing data in fact table(s) and dimension tables. The fact table contains one column for each dimension and one column for each measure and every row of the table [rovides one fact. A fact then is represented as (BSc, India, 2001-01) with the last column as 30. An OLAP tool is then provided to manipulate the data in these data warehouse tables. This tool essentially groups the fact table to find aggregates and uses some of the aggregates already computed to find new aggregates.

The advantage of using ROLAP is that it is more easily used with existing relational DBMS and the data can be stored efficiently using tables since no zero facts need to be stored. The disadvantage of the ROLAP model is its poor query performance. Proponents of the MLOAP model have called the ROLAP model SLOWLAP. Some products in this category are Oracle OLAP mode, OLAP Discoverer, MicroStrategy and Microsoft Analysis Services.

MOLAP

MOLAP is based on using a multidimensional DBMS rather than a data warehouse to store and access data. It may be considered as a top-down approach to OLAP. The multidimensional database systems do not have a standard approach to storing and maintaining their data. They often use special-purpose file systems or indexes that store pre-computation of all aggregations in the cube. For example, in ROLAP a cell was

represented as (BSc, India, 2001-01) with a value 30 stored in the last column. In MOLAP, the same information is stored as 30 and the storage location implicitly gives the values of the dimensions. The dimension values do not need to be stored since all the values of the cube could be stored in an array in a predefined way. For example, the cube in Figure 8.2 may be represented as an array like the following:

12	30	31	103	21	197	19	0	32	47	30	128	10	2	29	31	43	…

If the values of the dimensions are known, we can infer the cell location in the array. If the cell location is known, the values of the dimension may be inferred. This is obviously a very compact notation for storing a multidimensional data cube although a couple of problems remain. Firstly the array is likely to be too large to be stored in the main memory. Secondly, this representation does not solve the problem of efficiently representing sparse cubes. To overcome the problem of the array being too large for main memory, the array may be split into pieces called ‘chunks’, each of which is small enough to fit in the main memory. To overcome the problem of sparseness, the “chunks” may be compressed.

MOLAP systems have to deal with sparsity since a very percentage of the cells can be empty in some applications. The MOLAP implementation is usually exceptionally efficient. The disadvantage of using MOLAP is that it is likely to be more expensive than OLAP, the data is not always current, and it may be more difficult to scale a MOLAP system for very large OLAP problems. Some MOLAP products are Hyperion Essbase and Applix iTM1. Oracle and Microsoft are also competing in this segment of the OLAP market.

The differences between ROLAP and MOLAP are summarized in Table 8.8

Table 2.8 Comparison of MOLAP and ROLAP

Property	MOLAP	ROLAP
Data structure	Multidimensional database using sparse arrays	Relational tables (each cell is a row)
Disk space	Separate database for data cube; large for large data cubes	May not require any space other than that available in the data warehouse
Retrieval	Fast(pre-computed)	Slow(computes on-the-fly)
Scalability	Limited (cubes can be very large)	Excellent
Best suited for	Inexperienced users, limited set of queries	Experienced users, queries change frequently
DBMS facilities	Usually weak	Usually very strong

DATA CUBE OPERATIONS

A number of operations may be applied to data cubes. The common ones are:

- Roll-p
- Drill-down
- Slice and dice
- Pivot

Roll-up

Roll-up is like zooming out on the data cube. It is required when the user needs further abstraction or less detail. This operation performs further aggregations on the data, for example, from single degree programs to all programs offered by a School or department, from single countries to a collection of countries, and from individual semesters to

academic years. Often a hierarchy defined on a dimension is useful in the roll-up operation as suggested by the example of countries and regions.

We provide an example of roll-up based on Table =s 8.6, 8.7 and 8.8. We first define hierarchies on two dimensions. Amongst countries, let us define:

1. Asia (India, Malaysia, Singapore)
2. Europe (Sweden, UK)
3. Rest (Australia, USA)

Another hierarchy is defined on the dimension degree:

1. Science (BSc, BIT)
2. Medicine (MBBS)
3. Business and Law (BCom, LLB)

The result of a roll-up for both semesters together from Table 8.8 then is given in Table 8.9.

Table 2.9 Result of a roll-up operation using Table 8.7

Country \ Degree	Science	Medicine	Business and Law
Asia	160	81	106
Europe	60	50	115
Rest	68	61	162

Drill-down

Drill-down is like zooming in on the data and is therefore the reverse of roll-up. It is an appropriate operation when the user needs further details or when the user wants to partition more finely or wants to focus on some particular values of certain dimensions. Drill-down adds more details to the data. Hierarchy defined on a dimension may be involved in drill-down. For example, a higher level views of student data, for example in Table 8.9, gives student numbers for the two semesters for groups of countries and groups of degrees. If one is interested in more detail then it is possible to drill-down to tables 8.6 and 8.7 for student numbers in each of the semesters for each country and for each degree.

Slice and dice

Slice and dice are operations for browsing the data in the cube. The terms refer to the ability to look at information from different viewpoints.

A slice is a subset of the cube corresponding to a single value for one or more members of the dimensions. For example, a slice operation is performed when the user wants a selection on one dimension of a three-dimensional cube resulting in a two-dimensional site. Let the degree dimension be fixed as degree = BIT. The slice will not include any information about other degrees. The information retrieved therefore is more like a two-dimensional cube for degree = BIT as shown in Table 8.10.

Table 2.10 Result of a slice when degree value is 'BIT'

Country \ Semester	2000-01	2000-02	2001-01	2001-02
Australia	11	5	10	2
India	17	0	13	5
Malaysia	23	2	20	1
Singapore	31	4	23	2
Sweden	7	0	7	4
UK	13	8	11	6
USA	19	4	12	5

It should be noted that Table 8.7 also is a slice (with SSemester = '2000-01') from the cube built by piling several tables like Tables 8.7 and 8.8 about different semesters on top of each other. It is shown in Figure 8.4.

The dice operation is similar to slice but dicing does not involve reducing the number of dimensions. A dice is obtained by performing a selection on two or more dimensions. For example, one may only be interested in degrees BIT and BCom and countries Australia, India, and Malaysia for semesters 2000-01 and 2000-01. The result is a three-dimensional cube and we show it by Table 8.11 and 8.12 placed on top of each other. For example one may only be interested in the degrees BIT and BCom and the countries, Australia, India

and Malaysia for semesters 2000-01 and 2001-01. The result is a three-dimensional cube as shown in Figure 8.4 and we show it by Tables 8.11 and 8.12 for the two semesters placed on top of each other. We have left out the table that shows the totals from these tables.

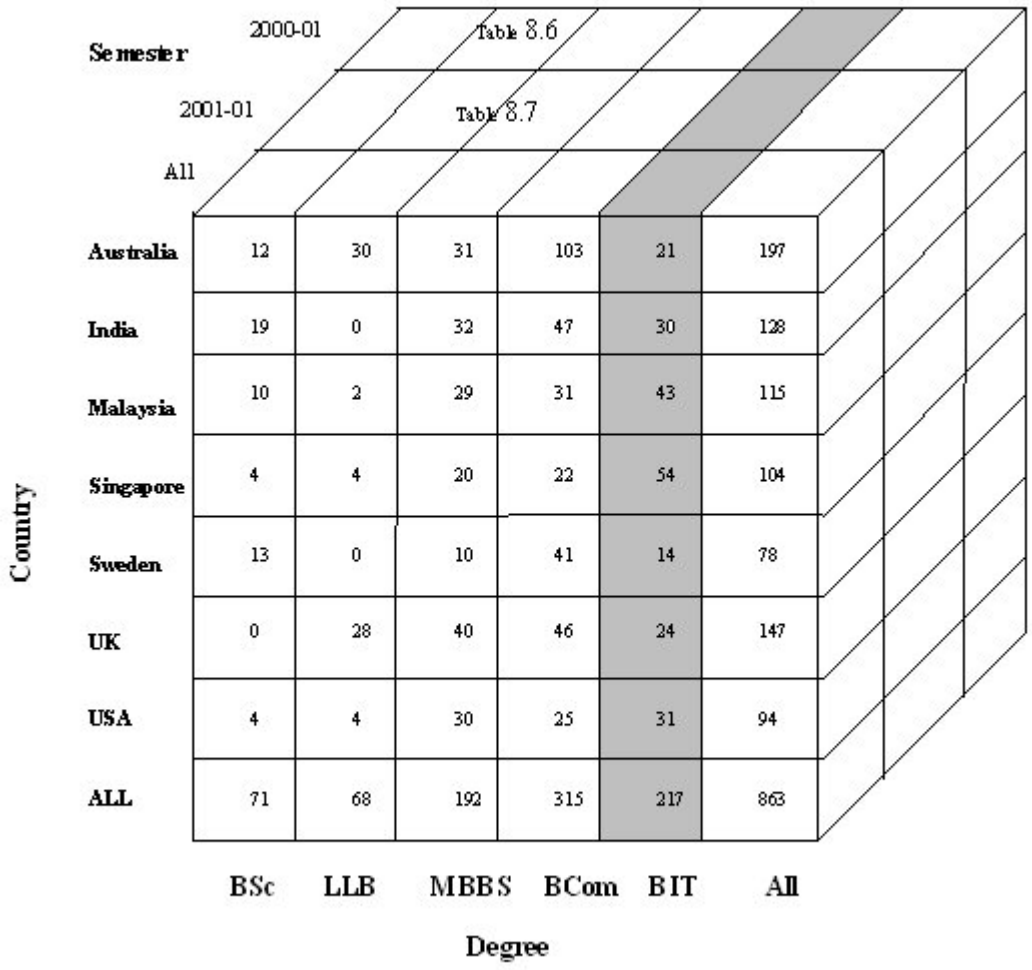


Figure 2.4 A slice from the cube in Figure 8.2

Table 2.11 A three-dimensional dice from a three-dimensional cube (SSemester 2000-01)

Country \ Degree	BCom	BIT

Australia	50	11
India	25	17
Malaysia	12	23

Table 2.12 A three-dimensional dice from a three-dimensional cube (SSemester 2001-01)

Country \ Degree	BCom	BIT
Australia	53	10
India	22	13
Malaysia	19	20

A dice may be shown as in Figure 8.5

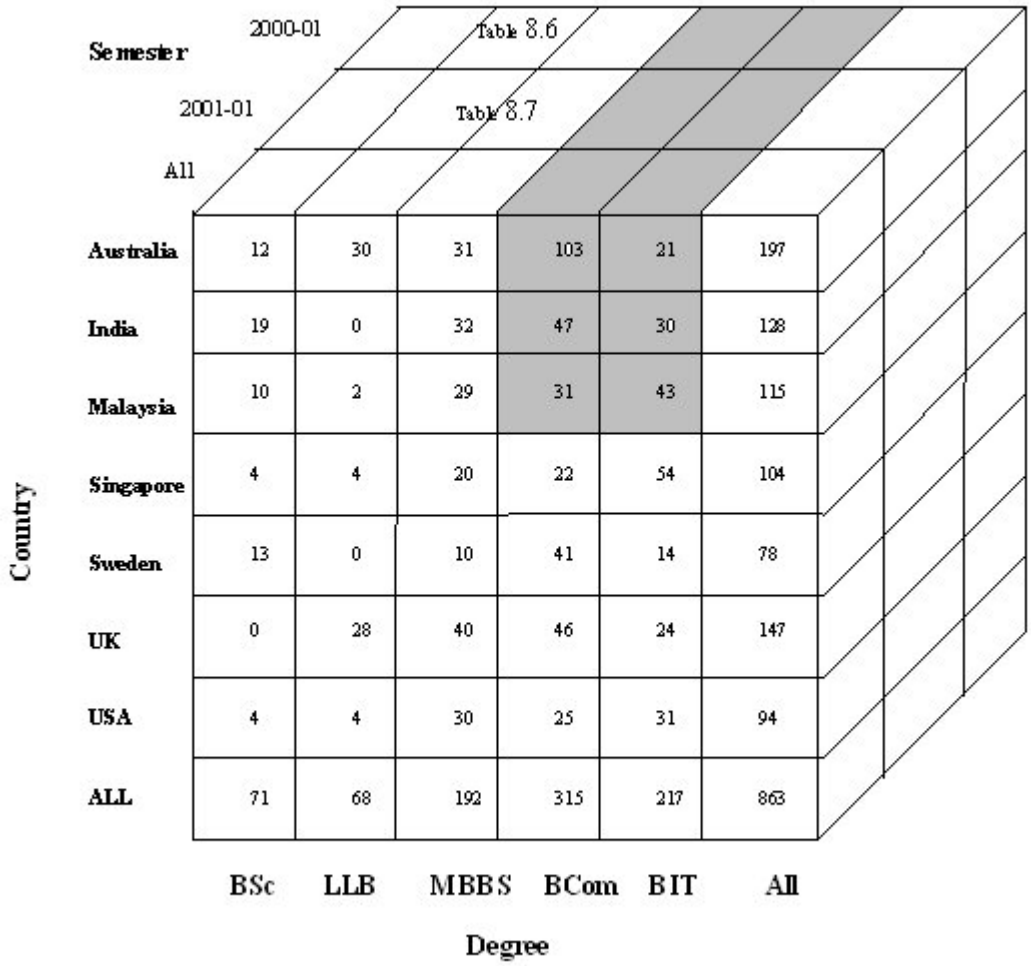


Figure 2.5 A dice from the cube in Figure 8.2

Slice and dice from often figure in interactive use of an OLAP system in which the user can navigate through the cube by specifying either one or two dimensions or values of all three dimensions that are of interest.

Pivot or Rotate

The pivot operation is used when the user wishes to re-orient the view of the data cube. It may involve swapping the rows and columns, or moving one of the row dimensions into the column dimension. For example, the cube consisting of Tables 8.6, 8.7 and 8.8 gives the dimension degree along the x-axis, country along the y-axis and starting semester along the z-axis (or the vertical axis). One may want to swap the dimensions country and

starting semester. The cube will then consist of several tables like those given in Tables 8.13 and 8.14 on top of each other.

Table 2.13 One table on which other similar tables are piled up in a rotated cube (Country dimension value = Australia)

Semester \ Degree	BSc	LLB	MBBS	BCom	BIT	ALL
2000-01	5	20	15	50	11	101
2001-01	7	10	16	53	10	96
ALL	12	30	31	103	21	197

Table 2.14 Another table that is part of a rotated cube (Country dimension value = India)

Semester \ Degree	BSc	LLB	MBBS	BCom	BIT	ALL
2000-01	10	0	15	25	17	67
2001-01	9	0	17	22	13	61
ALL	19	0	32	47	30	128

Clearly this rotated cube gives a different view of the same data. Such views can be particularly useful if the number of dimensions is greater than three.

GUIDELINES FOR OLAP IMPLEMENTATION

Following are a number of guidelines for successful implementation of OLAP. The guidelines are, somewhat similar to those presented for data warehouse implementation.

1. *Vision:* The OLAP team must, in consultation with the users, develop a clear vision for the OLAP system. This vision including the business objectives should be clearly defined, understood, and shared by the stakeholders.

2. *Senior management support:* The OLAP project should be fully supported by the senior managers. Since a data warehouse may have been developed already, this should not be difficult.

3. *Selecting an OLAP tool:* The OLAP team should familiarize themselves with the ROLAP and MOLAP tools available in the market. Since tools are quite different, careful planning may be required in selecting a tool that is appropriate for the enterprise. In some situations, a combination of ROLAP and MOLAP may be most effective.

4. *Corporate strategy:* The OLAP strategy should fit in with the enterprise strategy and business objectives. A good fit will result in the OLAP tools being used more widely.

5. *Focus on the users:* The OLAP project should be focused on the users. Users should, in consultation with the technical professional, decide what tasks will be done first and what will be done later. Attempts should be made to provide each user with a tool suitable for that person's skill level and information needs. A good GUI user interface should be provided to non-technical users. The project can only be successful with the full support of the users.

6. *Joint management:* The OLAP project must be managed by both the IT and business professionals. Many other people should be involved in supplying ideas. An appropriate committee structure may be necessary to channel these ideas.

7. *Review and adapt:* As noted in last chapter, organizations evolve and so must the OLAP systems. Regular reviews of the project may be required to ensure that the project is meeting the current needs of the enterprise.

OLAP SOFTWARE

There is much OLAP software available in the market.

A list is available at http://www.kdnuggets.com/software/dbolap.html.

Another is available at http://www.olapreport.com/market.htm.

The list below provides some major OLAP software.

- BI2M (Business Intelligence to Marketing and Management) from B&M Services has three modules one of which is for OLAP. The OLAP module allows database

exploring including slice and dice, roll-up, drill-down and displays results as 2D charts, 3D charts and tables.

- Business Objects OLAP Intelligence from BusinessObjects allows access to OLAP servers from Microsoft, Hyperion, IBM and SAP. Usual operations like slice and dice, and drill directly on multidimensional sources are possible. BusineeOjects also has widely used Crystal Analysis and Reports.

- ContourCube from Contour Components is an OLAP product that enables users to slice and dice, roll-up, drill-down and pivot efficiently.

- DB2 Cube Views from IBM includes features and functions for managing and deploying multidimensional data. It is claimed that OLAP solutions can be deployed quickly.

- Essbase Integration Services from Hyperion Solutions is a widely used suite of tools. The company's Web sites make it difficult to understand what the software does. It 2005 market ranking was 2.
- Everest from OutlookSoft is a Microsoft-based single application and database that provides operational reporting and analysis including OLAP and multidimensional slice and dice and other analysis operations.

- Executive Suite from CIP-Global is an integrated corporate planning, forecasting, consolidation and reporting solution based on Microsoft's SQL server 2000 and analysis Services Platform.

- Executive Viewer from Temtec provides users real-time Web access to OLAP databases such as Microsoft Analysis Services and Hyperion Essbase for advanced and ad hoc analysis as well as reporting.

- Express and the Oracle OLAP Option – Express is a multidimensional database and application development environment for building OLAP applications. It is MOLAP. OLAP Analytic workspaces is a porting of the Oracle Express analytic engine to the Oracle RDBMS kernel which now runs as an OLAP virtual machine.

- MicroStrategy 8 from MicroStrategy provides facilities for query, reporting and advanced analytical needs. It 2005 market ranking was 5.

- NovaView from Panorama extends the Microsoft platform that integrates analysis, reporting and performance measurement information into a single solution.

- PowerPlay from Cognos is widely used OLAP software that allows users to analyze large volumes of data with fast response times. Its 2005 market ranking was 3.
- SQL Server 2000 Analysis Services from Microsoft. SQL Server 2000 Analysis Services is the OLAP Services component in SQL Server 7.0

Unit 3

DATA MINING

INTRODUCTION

The complexity of modern society coupled with growing competition due to trade globalization has fuelled the demand for data mining. Most enterprises have collected information over at least the last 30 years and they are keen to discover business intelligence that might be buried in it. Business intelligence may be in the form of customer profiles which may result in better targeted marketing and other business actions.

During the 1970s and the 1980s, most Western societies developed a similar set of privacy principles and most of them enacted legislation to ensure that governments and the private sector were following good privacy principles. Data mining is a relatively new technology and privacy principles developed some 20-30 years ago are not particularly effective in dealing with privacy concerns that are being raised about data mining. These concerns have been heightened by the dramatically increased use of data mining by governments as a result of the 9/11 terrorist attacks. A number of groups all over the world are trying to wrestle with the issues raised by widespread use of data mining techniques.

Challenges

The daily use of the word privacy about information sharing and analysis is often vague and may be misleading. We will therefore provide a definition (or two). Discussions about the concept of information privacy started in the 1960s when a number of researchers recognized the dangers of privacy violations by large collections of personal information in computer systems. Over the years a number of definitions of information privacy have emerged. One of them defines information privacy as the individual's ability to control the circulation of information relating to him/her. Another widely used definition is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others.

Sometimes privacy is confused with confidentially and at other times with security. Privacy does involve confidentiality and security but it involves more than the two.

BASIC PRINCIPLES TO PROTECT INFORMATION PRIVACY

During the 1970s and 1980s many countries and organizations (e.g. OECD, 1980) developed similar basic information privacy principles which were then enshrined in legislation by many nations. These principles are interrelated and party overlapping and should therefore be treated together. The OECD principles are:

1. ***Collection limitation:*** There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject.
2. ***Data quality:*** Personal data should be relevant to the purposes for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-data.
3. ***Purpose specification:*** The purpose for which personal data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfilment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of purpose.

4. ***Use limitation:*** Personal data should not be disclosed, made available or otherwise used for purposes other than those specified in accordance with Principle 3 except with the consent of the data subject or by the authority of law.
5. ***Security safeguards:*** Personal data should be protected by reasonable security safeguards against such risks as loss of unauthorized access, destruction, use, modification or disclosure of data.
6. ***Openness:*** There should be general policy of openness about developments, practices and policies with respect to personal data. Means should be readily available for establishing the existence and nature of personal data, and the main purposes of their use, as well as the identity and usual residence of the data controller.
7. ***Individual participation:*** An individual should have the right:
   - **(a)** to obtain from a data controller, or otherwise, confirmation of whether or not the data controller has data relating to him;
   - **(b)** to have communicated to him, data relating to him
     within a reasonable time;
     at a charge, if any, that is not excessive;
     in a reasonable manner; and
     in a form that is readily intelligible to him;
   - **(c)** to be given reasons if a request made under subparagraphs (a) and (b) is denied, and to be able to challenge such denial; and
   - **(d)** to challenge data related to him and, if the challenge is successful, to have the data erased, rectified, completed or amended.
8. ***Accountability:*** A data controller should be accountable for complying with measures which give effect to the principles stated above.

These privacy protection principles were developed for online transaction processing (OLTP) systems before technologies like data mining became available. In OLTP systems, the purpose of the system is quite clearly defined since the system is used for a particular operational purpose of an enterprise (e.g. student enrolment). Given a clear purpose of the system, it is then possible to adhere to the above principles.

USES AND MISUSES OF DATA MINING

Data mining involves the extraction of implicit, previously unknown and potentially useful knowledge from large databases. Data mining is a very challenging task since it involves building and using software that will manage, explore, summarize, model, analyse and interpret large datasets in order to identify patterns and abnormalities.

Data mining techniques are being used increasingly in a wide variety of applications. The applications include fraud prevention, detecting tax avoidance, catching drug smugglers, reducing customer churn and learning more about customers' behaviour. There are also some (mis)uses of data mining that have little to do with any of these applications. For example, a number of newspapers in early 2005 have reported results of analyzing associations between the political party that a person votes for and the car the person drives. A number of car models have been listed in the USA for each of the two major political parties.

In the wake of the 9/11 terrorism attacks, considerable use of personal information, provided by individuals for other purposes as well as information collected by governments including intercepted emails and telephone conversations, is being made in the belief that such information processing (including data mining) can assist in identifying persons who are likely to be involved in terrorist networks or individuals who might be in contact with such persons or other individuals involved in illegal activities

drug smuggling). Under legislation enacted since 9/11, many governments are able to demand access to most private sector data. This data can include records on travel, shopping, utilities, credit, telecommunications and so on. Such data can them be mined in the belief that patterns can be found that will help in identifying terrorists or drug smugglers.

Consider a very simple artificial example of data in Table 9.1 being analysed using a data mining technique like the decision tree:

Table 3.1 A simple data mining example

Birth Country	Age	Religion	Visited X	Studied in West	Risk Class
A	<30	P	Yes	Yes	B
B	>60	Q	Yes	Yes	A
A	<30	R	Yes	No	C
X	30-45	R	No	No	B
Y	46-60	S	Yes	No	C
X	>60	P	Yes	Yes	A
Z	<25	P	No	Yes	B
A	<25	Q	Yes	No	A
B	<25	Q	Yes	No	C
B	30-45	S	Yes	No	C

Using the decision tree to analyse this data may result in rules like the following:

If Age = 30-45 and Birth Country = A and Visited X = Yes and Studied in West = Yes and

Religion = R then Risk Class = A.

User profiles are built based on relevant user characteristics. The number of characteristics may be large and may include all kinds of information including telephone zones phoned, travelling on the same flight as a person on a watch list and much more. User profiling is used in a variety of other areas, for example authorship analysis or plagiarism detection.

Once a user profile is formed, the basic action of the detection system is to compare incoming personal data to the profile and make a decision as to whether the data

fit any of the profiles. The comparison can in fact be quite complex because not all of the large numbers of characteristics in the profile are likely to match but a majority might.

Such profile matching can lead to faulty inferences. As an example, it was reported that a person was wrongly arrested just because the person had an Arab name and obtained a driver license at the same motor vehicle office soon after one of the 9/11 hijackers did. Although this incident was not a result of data mining, it does show that an innocent person can be mistaken for a terrorist or a drug smuggler as a result of some matching characteristics.

PRIMARY AIMS OF DATA MINING

Essentially most data mining techniques that we are concerned about are designed to discover and match profiles. The aims of the majority of such data mining activities are laudable but the techniques are not always perfect. What happens if a person matches the profile but does not belong to the category?

Perhaps it is not a matter of great concern if a telecommunications company labels a person as one that is likely to switch and then decides to target that person with a special campaign designed to encourage the person to stay. On the other hand, if the Customs department identifies a person as fitting the profile of a drug smuggler then that person is likely to undergo a special search whenever he/she returns home from overseas and perhaps at other airports if the customs department of one country shares information with other countries. This would be a matter of much more concern to governments.

Knowledge about the classification or profile of an individual who has been so classified or profiled may lead to disclosure of personal information with some given probability. The characteristics that someone may be able to deduce about a person with some possibility may include sensitive information, for example, race, religion, travel history, and level of credit card expenditure.

Data mining is used for many purposes that are beneficial to society, as the list of some of the common aims of data mining below shows.

- The primary aim of many data mining applications is to understand the customer better and improve customer services.

- Some applications aim to discover anomalous patterns in order to help identify, for example, fraud, abuse, waste, terrorist suspects, or drug smugglers.
- In many applications in private enterprises, the primary aim is to improve the profitability of an enterprise
- The primary purpose of data mining is to improve judgement, for example, in making diagnoses, in resolving crime, in sorting out manufacturing problems, in predicting share prices or currency movements or commodity prices.
- In some government applications, one of the aims of data mining is to identify criminal and fraud activities.
- In some situations, data mining is used to find patterns that are simply not possible without the help of data mining, given the huge amount of data that must be processed.

Data Mining Tasks

Data mining functionalities are used to specify the kind of patterns to be found in data mining tasks. In general, data mining tasks can be classified into two categories:

- Descriptive
- predictive

- Predictive tasks. The objective of these tasks is to predict the value of a particular attribute based on the values of other attribute.

  – Use some variables (independent/explanatory variable) to predict unknown or future values of other variables (dependent/target variable).

- Description Methods: Here the objective is to derive patterns that summarize the underlying relationships in data.

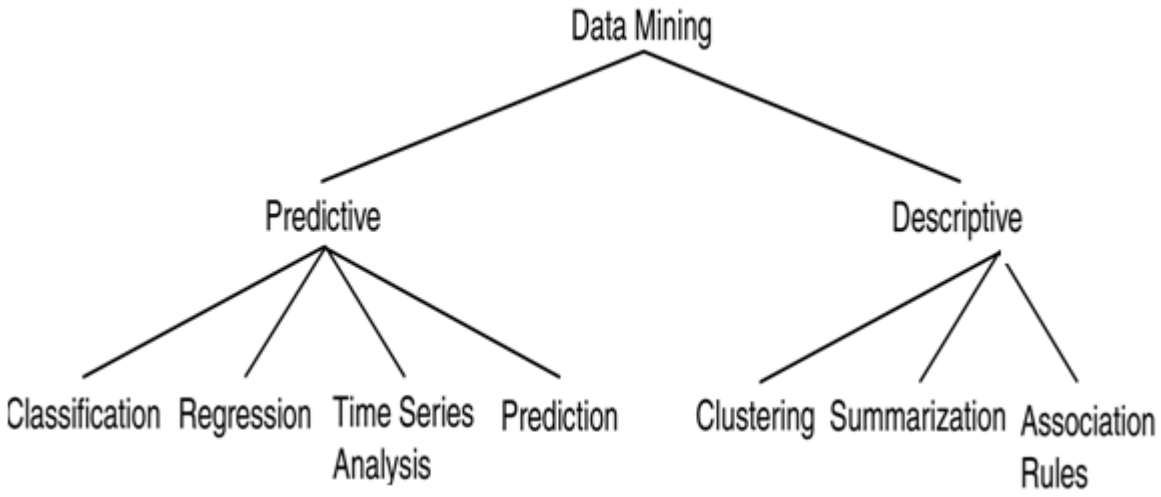
  – Find human-interpretable patterns that describe the data.

There are four core tasks in Data Mining:

i. Predictive modeling ii. Association analysis

iii. Clustering analysis, iv. Anomaly detection

Descriptive mining tasks characterize the general properties of the data in the database. Predictive mining tasks perform inference on the current data in order to make predictions.



Describe data mining functionalities, and the kinds of patterns they can discover (or) define each of the following data mining functionalities: characterization, discrimination, association and correlation analysis, classification, prediction, clustering, and evolution analysis. Give examples of each data mining functionality, using a real-life database that you are familiar with.

Figure 1.3 illustrates four of the core data mining tasks that are described in the remainder of this book.

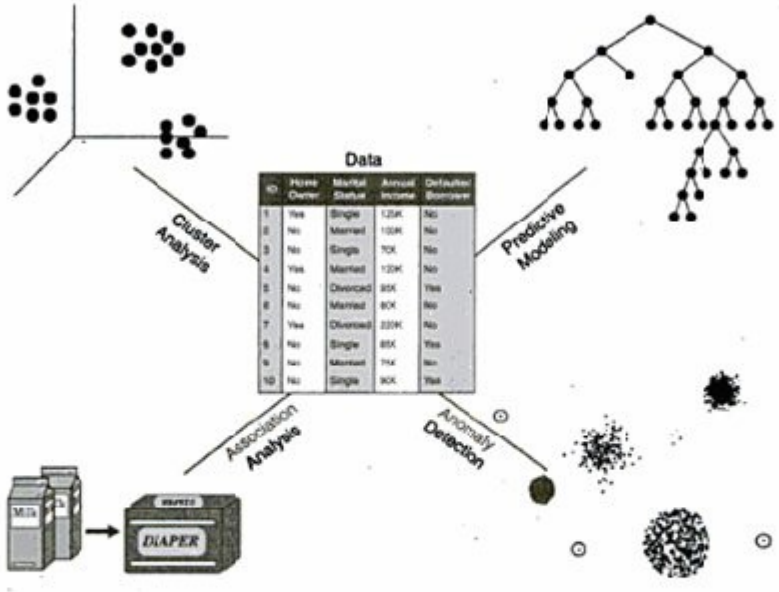


Figure 1.3. Four of the core data mining tasks.

1). predictive method

Find some missing or unavailable data values rather than class labels referred to as prediction. Although prediction may refer to both data value prediction and class label prediction, it is usually confined to data value prediction and thus is distinct from classification. Prediction also encompasses the identification of distribution trends based on the available data.

Example:

Predicting flooding is difficult problem. One approach is uses monitors placed at various points in the river. These monitors collect data relevant to flood prediction: water level, rain amount, time, humidity etc. These water levels at a potential flooding point in the river can be predicted based on the data collected by the sensors upriver from this point. The prediction must be made with respect to the time the data were collected

Classification:

- It predicts categorical class labels
- It classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

- Typical Applications
    - credit approval
    - target marketing
    - medical diagnosis
    - treatment effectiveness analysis

Classification can be defined as the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known).

Example:

An airport security screening station is used to deter mine if passengers are potential terrorist or criminals. To do this, the face of each passenger is scanned and its basic pattern(distance between eyes, size, and shape of mouth, head etc) is identified. This pattern is compared to entries in a database to see if it matches any patterns that are associated with known offenders

A classification model can be represented in various forms, such as

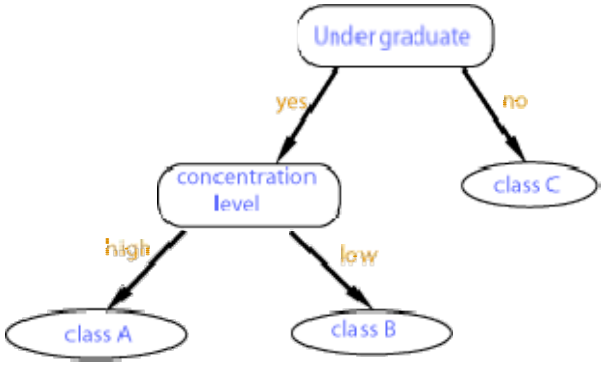
1) IF-THEN rules,

student (class , "undergraduate") AND concentration (level, "high") ==> class A

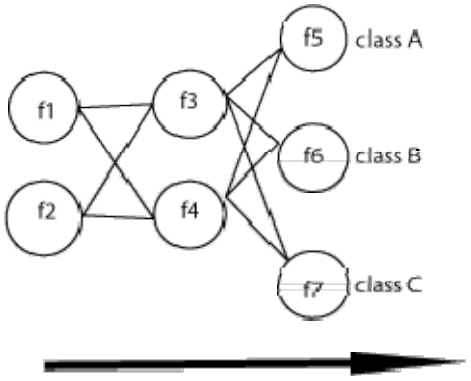
student (class ,"undergraduate") AND concentrtion (level,"low") ==> class B

student (class , "post graduate") ==> class C

2) Decision tree



3) Neural network.



.

Classification vs. Prediction

Classification differs from prediction in that the former is to construct a set of models (or functions) that describe and distinguish data class or concepts, whereas the latter is to predict some missing or unavailable, and often numerical, data values. Their similarity is that they are both tools for prediction: Classification is used for predicting the class label of data objects and prediction is typically used for predicting missing numerical data values.

2). Association Analysis

It is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data. For example, a data mining system may find association rules like

major(X, “computing science””) I owns(X, “personal computer”)

[support = 12%, confidence = 98%]

where X is a variable representing a student. The rule indicates that of the students under study, 12% (support) major in computing science and own a personal computer. There is a 98% probability (confidence, or certainty) that a student in this group owns a personal computer.

Example:

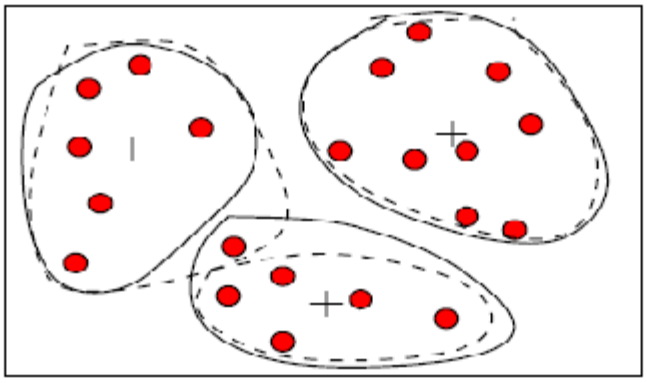
A grocery store retailer to decide whether to but bread on sale. To help determine the impact of this decision, the retailer generates association rules that show what other products are frequently purchased with bread. He finds 60% of the times that bread is sold so are pretzels and that 70% of the time jelly is also sold. Based on these facts, he tries to capitalize on the association between bread, pretzels, and jelly by placing some pretzels and jelly at the end of the aisle where the bread is placed. In addition, he decides not to place either of these items on sale at the same time.

3). Clustering analysis

Clustering analyzes data objects without consulting a known class label. The objects are clustered or grouped based on the principle of maximizing the intra-class similarity and minimizing the interclass similarity. Each cluster that is formed can be viewed as a class of objects.

Example:A certain national department store chain creates special catalogs targeted to various demographic groups based on attributes such as income, location and physical characteristics of potential customers (age, height, weight, etc). To determine the target mailings of the various catalogs and to assist in the creation of new, more specific catalogs, the company performs a clustering of potential customers based on the determined attribute values. The results of the clustering exercise are the used by management to create special catalogs and distribute them to the correct target population based on the cluster for that catalog.

Clustering can also facilitate taxonomy formation, that is, the organization of observations into a hierarchy of classes that group similar events together as shown below:



customer data with respect to customer locations in a city, showing three data clusters. Each cluster 'center' is marked with a '+'.

Classification vs. Clustering

- In general, in classification you have a set of predefined classes and want to know which class a new object belongs to.
- Clustering tries to group a set of objects and find whether there is *some* relationship between the objects.
- In the context of machine learning, classification is *supervised learning* and clustering is *unsupervised learning*.

4). Anomaly Detection

It is the task of identifying observations whose characteristics are significantly different from the rest of the data. Such observations are called anomalies or outliers. This is useful in fraud detection and network intrusions.

Types of Data

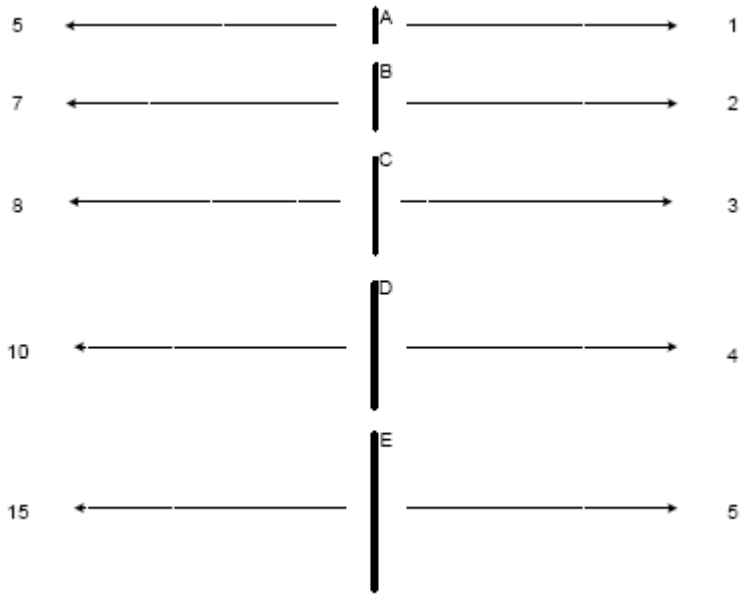
A Data set is a Collection of data objects and their attributes. An data object is also known as record, point, case, sample, entity, or instance. An attribute is a property or characteristic of an object. Attribute is also known as variable, field, characteristic, or feature.

Attributes and Measurements

An attribute is a property or characteristic of an object. Attribute is also known as variable, field, characteristic, or feature. Examples: eye color of a person, temperature, etc. A collection of attributes describe an object.

Attribute Values: Attribute values are numbers or symbols assigned to an attribute. Distinction between attributes and attribute values– Same attribute can be mapped to different attribute values. Example: height can be measured in feet or meters.

The way you measure an attribute is somewhat may not match the attributes properties.



– Different attributes can be mapped to the same set of values. Example: Attribute values for ID and age are integers. But properties of attribute values can be different, ID has no limit but age has a maximum and minimum value.

Attributes

Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

The types of an attribute

A simple way to specify the type of an attribute is to identify the properties of numbers that correspond to underlying properties of the attribute.

- Properties of Attribute Values
  The type of an attribute depends on which of the following properties it possesses:
  – Distinctness: = ≠
  – Order: < >
  – Addition: + -
  – Multiplication: * /

There are different types of attributes
– Nominal
Examples: ID numbers, eye color, zip codes
– Ordinal
Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
– Interval
Examples: calendar dates, temperatures in Celsius or Fahrenheit.
– Ratio
Examples: temperature in Kelvin, length, time, counts

Attribute Type	Desfription	Examples	Opera tions
Nolllillal	The values of a non1il1oll attribute are justollffr"1:-:T11Hi-tlllr:s.lr: .110111i11;sl i1UJ ilJul s inovi11t": ouly ""teh infurn1ation to di tinguish one object fr01n another. (-.¢)	zjp ..:odes. e1nployce 't'<. {tnulf.!..;'P..11y1/ }	mode. eiu.ropy . con1111 11y co11c-la1lc n1, x..'. Ir.st
Ordinal	The values of :t..'1 ordinal anribute pro•.:ide enough Ulfom1a1ion to order obie:ts. ("".,>)	hardnes; of minerals, *{good, i>orror, bast}*, i1ades, t:-eet numbers	m ian, percentile;, rank correlarion, mu te!;;ts, sign tests
T11lr"t Vi'll	Fn1 1111c-1val aU1ll11t1.t"s, tht' differences bet\veei.1*«i2lue* are 1nea.uingful, i.e.• a unit of 1nea$uren:ent exist!;. (+, -)	ca1r:ncLo cla1r'S.. remperature in Cel lus or F:tluenheit	11li-111 .""->ttt(hll il deviation, Pearson's ccrrebtion, *t*a11d *F* te$'i-$
R atio	Fo1 1atin va1ialil s,l1nlh di(fr-tr:111t:!-. and :atio< a:e meaningf ul ('',*f*)	rt"'n1p1.i1httt":i11Kl\>in, 1uonetary qu<:nritie-i, counts, age-,1uass, length, electrical curreut	2,1"01111":1.l H-ut"it11, hanuoni:!.Ue-an, percenr variat:on

Attribute Level	Transformation	Comments
Nominal	Any permutation of values	If all employee ID numbers were reassigned, would it make any difference?
Ordinal	An order preserving change of values, i.e., *new_value = f(old_value)* where *f* is a monotonic function.	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Interval	*new_value =a * old_value + b* where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
Ratio	*new_value = a * old_value*	Length can be measured in meters or feet.

Describing attributes by the number of values

D **Discrete Attribute**– Has only a finite or countably infinite set of values, examples: zip codes, counts, or the set of words in a collection of documents, often represented as integer variables. Binary attributes are a special case of discrete attributes

D **Continuous Attribute**– Has real numbers as attribute values, examples: temperature, height, or weight. Practically, real values can only be measured and represented using a finite number of digits. Continuous attributes are typically represented as floating-point variables.

D **Asymmetric Attribute**-only a non-zero attributes value which is different from other values.

Preliminary investigation of the data to better understand its specific characteristics, it can help to answer some of the data mining questions

– To help in selecting pre-processing tools

– To help in selecting appropriate data mining algorithms

I Things to look at: Class balance, Dispersion of data attribute values, Skewness, outliers, missing values, attributes that vary together, Visualization tools are important, Histograms, box plots, scatter plots Many datasets have a discrete (binary) attribute class

I Data mining algorithms may give poor results due to class imbalance problem, Identify the problem in an initial phase.

General characteristics of data sets:

- Dimensionality: of a data set is the number of attributes that the objects in the data set possess. Curse of dimensionality refers to analyzing high dimensional data.
- Sparsity: data sets with asymmetric features like most attributes of an object with value 0; in some cases it may be with value non-zero.
- Resolution: it is possible to obtain different levels of resolution of the data.

Now there are varieties of data sets are there, let us discuss some of the following.

1. **Record**
   - **Data Matrix**
   - **Document Data**
   - **Transaction Data**

2. **Graph**
   - **World Wide Web**
   - **Molecular Structures**

3. **Ordered**
   - **Spatial Data**
   - **Temporal Data**
   - **Sequential Data**
   - **Genetic Sequence Data**

Record Data

Data that consists of a collection of records, each of which consists of a fixed set of attributes

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Transaction or market basket Data

A special type of record data, where each transaction (record) involves a set of items. For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Transaction data is a collection of sets of items, but it can be viewed as a set of records whose fields are asymmetric attributes.

Transaction data can be represented as sparse data matrix: **market basket representation**

– Each record (line) represents a transaction

– Attributes are binary and asymmetric

Tid	*Bread*	*Coke*	*Milk*	*Beer*	*Diaper*
1	1	1	1	0	0
2	1	0	0	1	0
3	0	1	1	1	1
4	1	0	1	1	1
5	0	1	1	0	1

Data Matrix

An M*N matrix, where there are M rows, one for each object, and N columns, one for each attribute. This matrix is called a data matrix, which holds only numeric values to its cells.

D If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute

D Such data set can be represented by an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

The Sparse Data Matrix

It is a special case of a data matrix in which the attributes are of the same type and are asymmetric; i.e. , only non-zero values are important.

Document Data

Each document becomes a `term' vector, each term is a component (attribute) of the vector, and the value of each component is the number of times the corresponding term occurs in the document.

Graph-based data

In general, the data can take many forms from a single, time-varying real number to a complex interconnection of entities and relationships. While graphs can represent this entire spectrum of data, they are typically used when relationships are crucial to the domain. Graph-based data mining is the extraction of novel and useful knowledge from a graph representation of data. Graph mining uses the natural structure of the application domain and mines directly over that structure. The most natural form of knowledge that can be extracted from graphs is also a graph. Therefore, the knowledge, sometimes referred to as patterns, mined from the data are typically expressed as graphs, which may be sub-graphs of the graphical data, or more abstract expressions of the trends reflected in the data. The need of mining structural data to uncover objects or concepts that relates objects (i.e., sub-graphs that represent associations of features) has increased in the past ten years, involves the automatic extraction of novel and useful knowledge from a graph representation of data. a graph-based knowledge discovery system that finds structural, relational patterns in data representing entities and relationships. This algorithm was the first proposal in the topic and has been largely extended through the years. It is able to develop graph shrinking as well as frequent substructure extraction and hierarchical conceptual clustering.

A graph is a pair G = (V, E) where V is a set of vertices and E is a set of edges. Edges connect one vertices to another and can be represented as a pair of vertices. Typically each edge in a graph is given a label. Edges can also be associated with a weight.

We denote the vertex set of a graph g by V (g) and the edge set by E(g). A label function, L, maps a vertex or an edge to a label. A graph g is a sub-graph of another graph g' if there exists a sub-graph isomorphism from g to g'. (Frequent Graph) Given a labeled graph dataset, D = {G1, G2, . . . , Gn}, support (g) [or frequency(g)] is the percentage (or number) of graphs in D where g is a sub-graph. A frequent (sub) graph is a graph whose support is no less than a minimum support threshold, min support.

Spatial data

Also known as *geospatial data* or *geographic information* it is the data or information that identifies the geographic location of features and boundaries on Earth, such as natural or constructed features, oceans, and more. Spatial data is usually stored as coordinates and topology, and is data that can be mapped. Spatial data is often accessed, manipulated or analyzed through Geographic Information Systems (GIS).

Measurements in spatial data types: In the planar, or flat-earth, system, measurements of distances and areas are given in the same unit of measurement as coordinates. Using the geometry data type, the distance between (2, 2) and (5, 6) is 5 units, regardless of the units used.

In the ellipsoidal or round-earth system, coordinates are given in degrees of latitude and longitude. However, lengths and areas are usually measured in meters and square meters, though the measurement may depend on the spatial reference identifier (SRID) of the geography instance. The most common unit of measurement for the geography data type is meters.

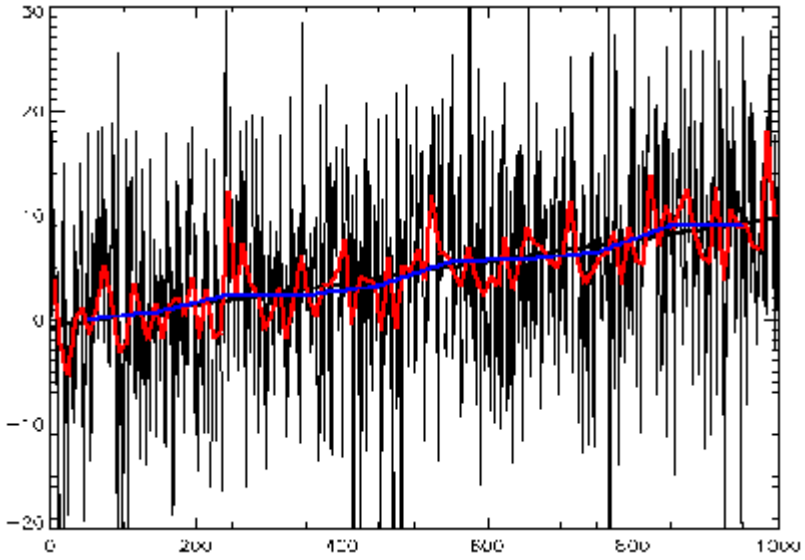
Orientation of spatial data: In the planar system, the ring orientation of a polygon is not an important factor. For example, a polygon described by ((0, 0), (10, 0), (0, 20), (0, 0)) is the same as a polygon described by ((0, 0), (0, 20), (10, 0), (0, 0)). The OGC Simple Features for SQL Specification does not dictate a ring ordering, and SQL Server does not enforce ring ordering.

Time Series Data

A time series is a sequence of observations which are ordered in time (or space). If observations are made on some phenomenon throughout time, it is most sensible to display the data in the order in which they arose, particularly since successive observations will probably be dependent. Time series are best displayed in a scatter plot. The series value X is plotted on the vertical axis and time t on the horizontal axis. Time is called the independent variable (in this case however, something over which you have little control). There are two kinds of time series data:

1. Continuous, where we have an observation at every instant of time, e.g. lie detectors, electrocardiograms. We denote this using observation X at time t, X(t).

2. Discrete, where we have an observation at (usually regularly) spaced intervals. We denote this as Xt.



Examples

Economics - weekly share prices, monthly profits

Meteorology - daily rainfall, wind speed, temperature

Sociology - crime figures (number of arrests, etc), employment figures

Sequence Data

Sequences are fundamental to modeling the three primary medium of human communication: speech, handwriting and language. They are the primary data types in several sensor and monitoring applications. Mining models for network intrusion detection view data as sequences of TCP/IP packets. Text information extraction systems model the input text as a sequence of words and delimiters. Customer data mining applications profile buying habits of customers as a sequence of items purchased. In computational biology, DNA, RNA and protein data are all best modeled as sequences.

A sequence is an ordered set of pairs (t1 x1) . . . (tn xn) where ti denotes an ordered attribute like time (ti−1 _ ti) and xi is an element value. The length n of sequences in a database is typically variable. Often the first attribute is not explicitly specified and the order of the elements is implicit in the position of the element. Thus, a sequence x can be written as x1 . . . xn. The elements of a sequence are allowed to be of many different types. When xi is a real number, we get a time series. Examples of such sequences abound — stock prices along time, temperature

measurements obtained from a monitoring instrument in a plant or day to day carbon monoxide levels in the atmosphere. When si is of discrete or symbolic type we have a categorical sequence.

Measures of Similarity and Dissimilarity, Data Mining Applications

Data mining focuses on (1) the detection and correction of data quality problems (2) the use of algorithms that can tolerate poor data quality. Data are of high quality "if they are fit for their intended uses in operations, decision making and planning" (J. M. Juran). Alternatively, the data are deemed of high quality if they correctly represent the real-world construct to which they refer. Furthermore, apart from these definitions, as data volume increases, the question of *internal consistency* within data becomes paramount, regardless of fitness for use for any external purpose, e.g. a person's age and birth date may conflict within different parts of a database. The first views can often be in disagreement, even about the same set of data used for the same purpose.

Definitions are:

- Data quality: The processes and technologies involved in ensuring the conformance of data values to business requirements and acceptance criteria.
- Data exhibited by the data in relation to the portrayal of the actual scenario.
- The state of completeness, validity, consistency, timeliness and accuracy that makes data appropriate for a specific use.

Data quality aspects: Data size, complexity, sources, types and formats Data processing issues, techniques and measures *We are drowning in data, but starving of knowledge* (Jiawei Han).

Dirty data

What does *dirty data* mean?

Incomplete data(missing attributes, missing attribute values, only aggregated data, etc.) Inconsistent data (different coding schemes and formats, impossible values or out-of-range values), Noisy data (containing errors and typographical variations, outliers, not accurate values)

Data quality is a perception or an assessment of data's fitness to serve its purpose in a given context.

Aspects of data quality include:

- Accuracy
- Completeness
- Update status
- Relevance
- Consistency across data sources
- Reliability
- Appropriate presentation
- Accessibility

Measurement and data collection issues

Just think about the statement below" a person has a height of 2 meters, but weighs only 2kg`s ". This data is inconsistence. So it is unrealistic to expect that data will be perfect.

Measurement error refers to any problem resulting from the measurement process. The numerical difference between measured value to the actual value is called as an **error**. Both of these errors can be random or systematic.

Noise and artifacts

Noise is the random component of a measurement error. It may involve the distortion of a value or the addition of spurious objects. Data Mining uses some robust algorithms to produce acceptable results even when noise is present.

Data errors may be the result of a more deterministic phenomenon called as artifacts.

Precision, Bias, and Accuracy

The quality of measurement process and the resulting data are measured by *Precision* and *Bias.*

Accuracy refers to the degree of measurement error in data.

Outliers

Missing Values

It is not unusual for an object to be missed its attributes. In some cases information is not collected properly. Example application forms , web page forms.

Strategies for dealing with missing data are as follows:

- Eliminate data objects or attributes with missing values.
- Estimate missing values

- Ignore the missing values during analysis

Inconsistent values

Suppose consider a city like kengeri which is having zipcode 560060, if the user will give some other value for this locality then we can say that inconsistent value is present.

Duplicate data

Sometimes Data set contain same object more than once then it is called duplicate data. To detect and eliminate such a duplicate data two main issues are addressed here; first, if there are two objects that actually represent a single object, second the values of corresponding attributes may differ.

Issues related to applications are timelines of the data, knowledge about the data and relevance of the data.

UNIT IV

ASSOCIATION ANALYSIS

This chapter presents a methodology known as association analysis, which is useful for discovering interesting relationships hidden in large data sets. The uncovered relationships can be represented in the form of association rules or sets of frequent items. For example, the following rule can be extracted from the data set shown in Table 4.1:

$$\{\text{Diapers}\} \rightarrow \{\text{Beer}\}.$$

Table 4.1. An example of market basket transactions.

TID	**ITEMS**
1	{Bread, Milk}
2	{ Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

The rule suggests that a strong relationship exists between the sale of diapers and beer because many customers who buy diapers also buy beer. Retailers can use this type of rules to help them identify new opportunities for cross- selling their products to the customers.

.

Basic Concepts and Algorithms

This section reviews the basic terminology used in association analysis and presents a formal description of the task.

Binary Representation Market basket data can be represented in a binary format as shown in Table 4.2, where each row corresponds to a transaction and each column corresponds to an item.

An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise. Because the presence of an item in a transaction is often considered more important than its absence, an item is an asymmetric binary variable.

Table 4.2 A binary 0/1 representation of market basket data.

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

This representation is perhaps a very simplistic view of real market basket data because it ignores certain important aspects of the data such as the quantity of items sold or the price paid to purchase them. Itemset and Support Count Let I = {i1,i2,. . .,id} be the set of all items in a market basket data and T = {t1, t2, . . . , tN } be the set of all transactions. Each transaction ti contains a subset of items chosen from I. In association analysis, a collection of zero or more items is termed an itemset. If an itemset contains k items, it is called a k-itemset. For instance, {Beer, Diapers, Milk} is an example of a 3-itemset. The null (or empty) set is an itemset that does not contain any items.

The transaction width is defined as the number of items present in a transaction. A transaction tj is said to contain an itemset X if X is a subset of tj. For example, the second transaction shown in Table 6.2 contains the item-set {Bread, Diapers} but not {Bread, Milk}. An important property of an itemset is its support count, which refers to the number of transactions that contain a particular itemset. Mathematically, the support count, σ(X), for an itemset X can be stated as follows:

$$\sigma(X) = \left|\{t_i | X \subseteq t_i,\ t_i \in T\}\right|,$$

Where the symbol | · | denote the number of elements in a set. In the data set shown in Table 4.2, the support count for {Beer, Diapers, Milk} is equal to two because there are only two transactions that contain all three items.

Association Rule An association rule is an implication expression of the form X → Y , where X and Y are disjoint itemsets, i.e., X ∩ Y = 0. The strength of an association rule can be measured in terms of its support and confidence. Support determines how often a rule is applicable to a given data set, while confidence determines how frequently items in *Y* appear in transactions that contain *X*. The formal definitions of these metrics are

Support s(X------->Y) = $\frac{\partial(XUY)}{N}$ 4.1

Confidence C(X------>Y) = $\partial(XUY)/\partial(X)$ 4.2

Formulation of Association Rule Mining Problem The association rule mining problem can be formally stated as follows:

Definition 4.1 (Association Rule Discovery). Given a set of transactions T , find all the rules having support ≥ minsup and confidence ≥ minconf, where minsup and minconf are the corresponding support and confidence thresholds.

From Equation 4.2, notice that the support of a rule X -→ Y depends only on the support of its corresponding itemset, X ∩ Y . For example, the following rules have identical support because they involve items from the same itemset,

{Beer, Diapers, Milk}:

{Beer, Diapers} -→{Milk}, {Beer, Milk} -→{Diapers},

{Diapers, Milk} -→{Beer}, {Beer} -→{Diapers, Milk},

{Milk} -→{Beer,Diapers}, {Diapers} -→{Beer,Milk}.

If the itemset is infrequent, then all six candidate rules can be pruned immediately without

our having to compute their confidence values. Therefore, a common strategy adopted by many association rule mining algorithms is to decompose the problem into two major subtasks:

1. Frequent Itemset Generation, whose objective is to find all the item-sets that satisfy the minsup threshold. These itemsets are called frequent itemsets.

2. Rule Generation, whose objective is to extract all the high-confidence rules from the frequent itemsets found in the previous step. These rules are called strong rules.

The computational requirements for frequent itemset generation are generally more expensive than those of rule generation.

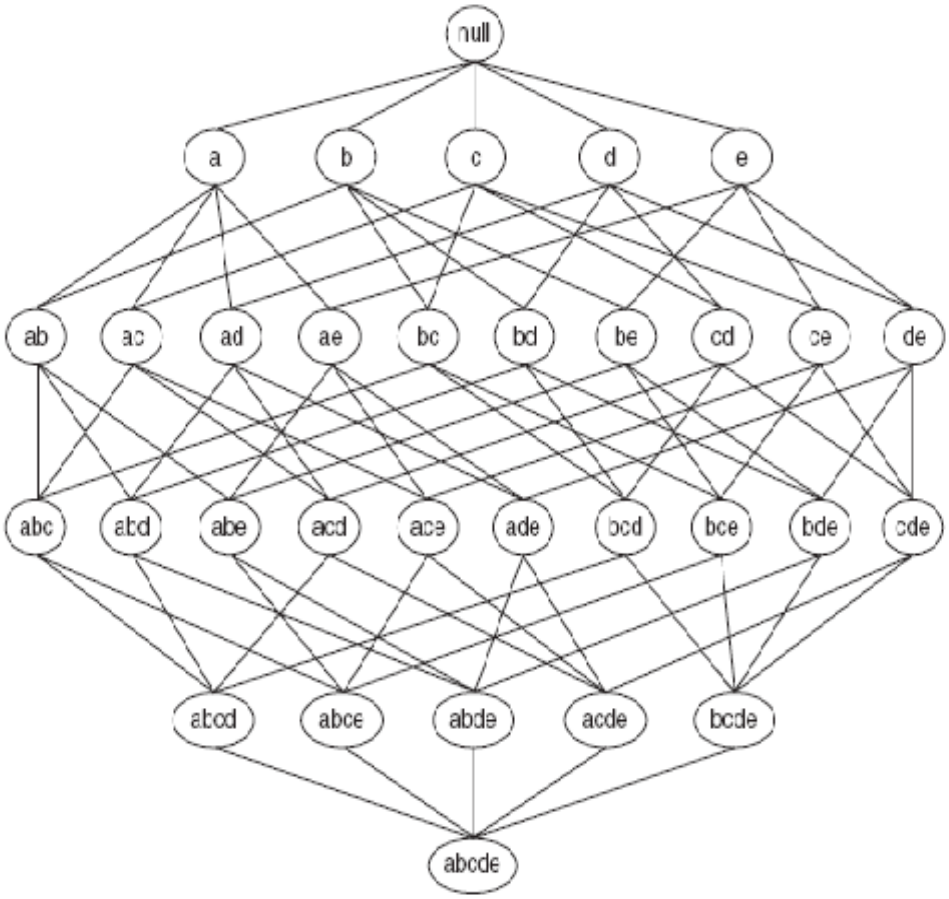


Figure 4.1. An itemset lattice.

Frequent Itemset Generation

A lattice structure can be used to enumerate the list of all possible itemsets. Figure 4.1 shows an itemset lattice for I = {a, b, c, d, e}. In general, a data set that contains k items can potentially generate up to 2k - 1 frequent itemsets, excluding the null set. Because k can be very large in many practical applications, the search space of itemsets that need to be explored is exponentially large.

A brute-force approach for finding frequent itemsets is to determine the support count for every candidate itemset in the lattice structure. To do this, we need to compare each candidate against every transaction, an operation that is shown in Figure 4.2. If the candidate is contained in a transaction, its support count will be incremented. For example, the support for {Bread,Milk} is incremented three times because the itemset is contained in transactions 1, 4, and 5. Such an approach can be very expensive because it requires O(NMw) comparisons, where N is the number of transactions, M =2k - 1 is the number of candidate itemsets, and w is the maximum transaction width.

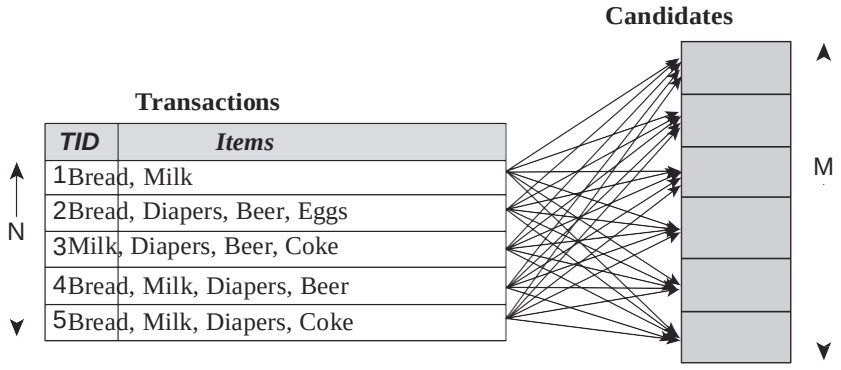


Figure 6.2. Counting the support of candidate itemsets.

There are several ways to reduce the computational complexity of frequent itemset generation.

1. Reduce the number of candidate itemsets (M). The Apriori principle, described in the next section, is an effective way to eliminate some of the candidate itemsets without counting their support values.

2. Reduce the number of comparisons. Instead of matching each candidate itemset against

every transaction, we can reduce the number of comparisons by using more advanced data structures, either to store the candidate itemsets or to compress the data set.

The Apriori Principle

This section describes how the support measure helps to reduce the number of candidate itemsets explored during frequent itemset generation. The use of support for pruning candidate itemsets is guided by the following principle.

Theorem 4.1 (Apriori Principle). If an itemset is frequent, then all of its subsets must also be frequent. To illustrate the idea behind the Apriori principle, consider the itemset lattice shown in Figure 4.3. Suppose {c, d, e} is a frequent itemset. Clearly, any transaction that contains {c, d, e} must also contain its subsets, {c, d},{c, e}, {d, e}, {c}, {d}, and {e}. As a result, if {c, d, e} is frequent, then all subsets of {c, d, e} (i.e., the shaded itemsets in this figure) must also be frequent.

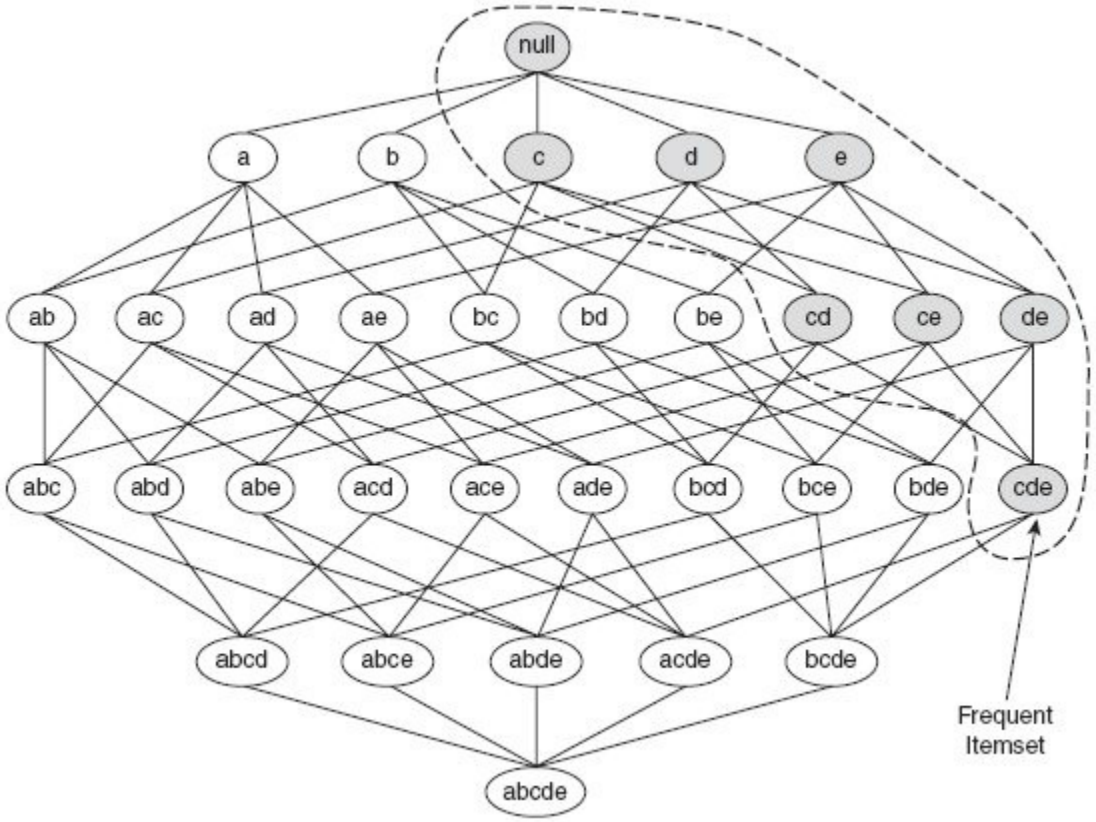


Figure 4.3. An illustration of the Apriori principle.

If {c, d, e} is frequent, then all subsets of this itemset are frequent.

Conversely, if an itemset such as {a, b} is infrequent, then all of its supersets must be infrequent too. As illustrated in Figure 6.4, the entire subgraph containing the supersets of {a, b} can be pruned immediately once {a, b} is found to be infrequent. This strategy of trimming the exponential search space based on the support measure is known as support-based pruning. Such a pruning strategy is made possible by a key property of the support measure, namely, that the support for an itemset never exceeds the support for its subsets. This property is also known as the anti-monotone property of the support measure.

Definition 4.2 (Monotonicity Property). Let I be a set of items, and J =2I be the power set of I. A measure f is monotone (or upward closed) if

$$\forall X, Y \in J : \ (X \subseteq Y) \longrightarrow f(X) \leq f(Y),$$

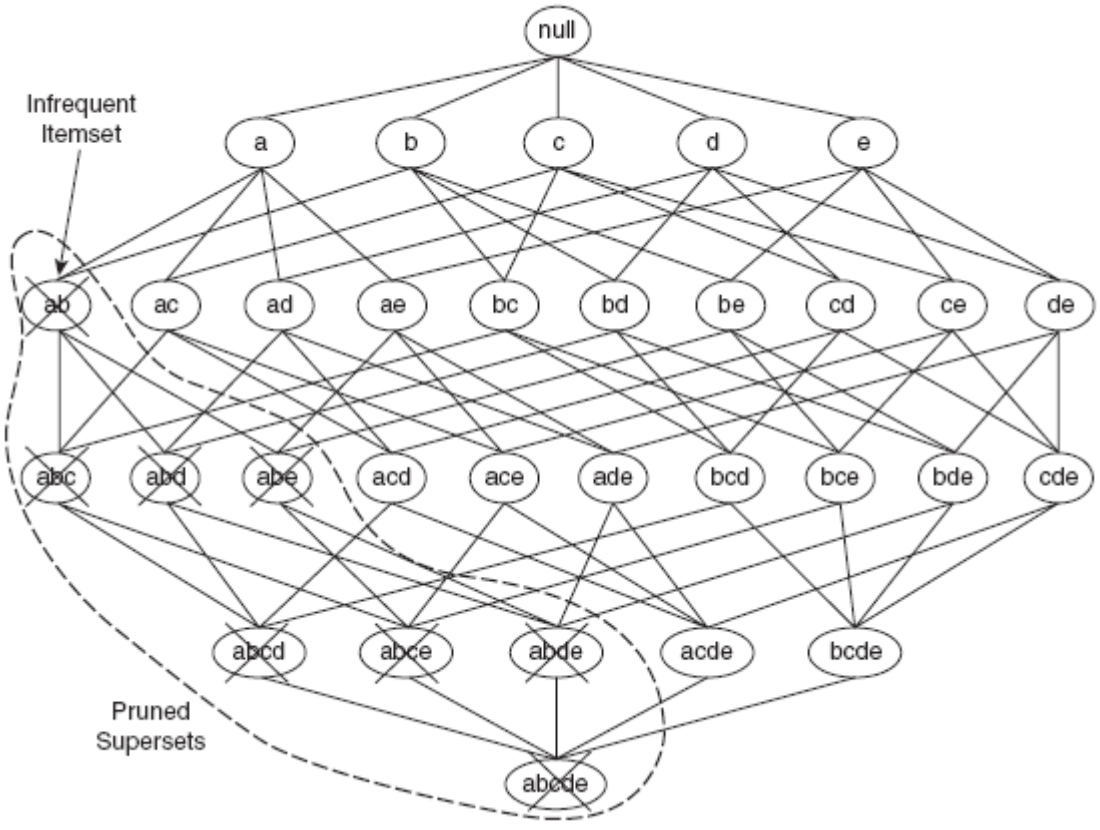


Figure 4.4. An illustration of support-based pruning.

If {a, b} is infrequent, then all supersets of {a, b} are infrequent, which means that if X is a subset of Y , then f(X) must not exceed f(Y). On the other hand, f is anti-monotone (or downward closed) if which means that if X is a subset of Y , then f(Y) must not exceed f(X).

$$\forall X, Y \in J : \ (X \subseteq Y) \longrightarrow f(Y) \leq f(X),$$

Any measure that possesses an anti-monotone property can be incorporated directly into the mining algorithm to effectively prune the exponential search space of candidate itemsets, as will be shown in the next section.

Frequent Itemset Generation in the Apriori Algorithm

Apriori is the first association rule mining algorithm that pioneered the use of support-based pruning to systematically control the exponential growth of candidate itemsets. Figure 4.5 provides a high-level illustration of the frequent itemset generation part of the Apriori algorithm for the transactions shown in

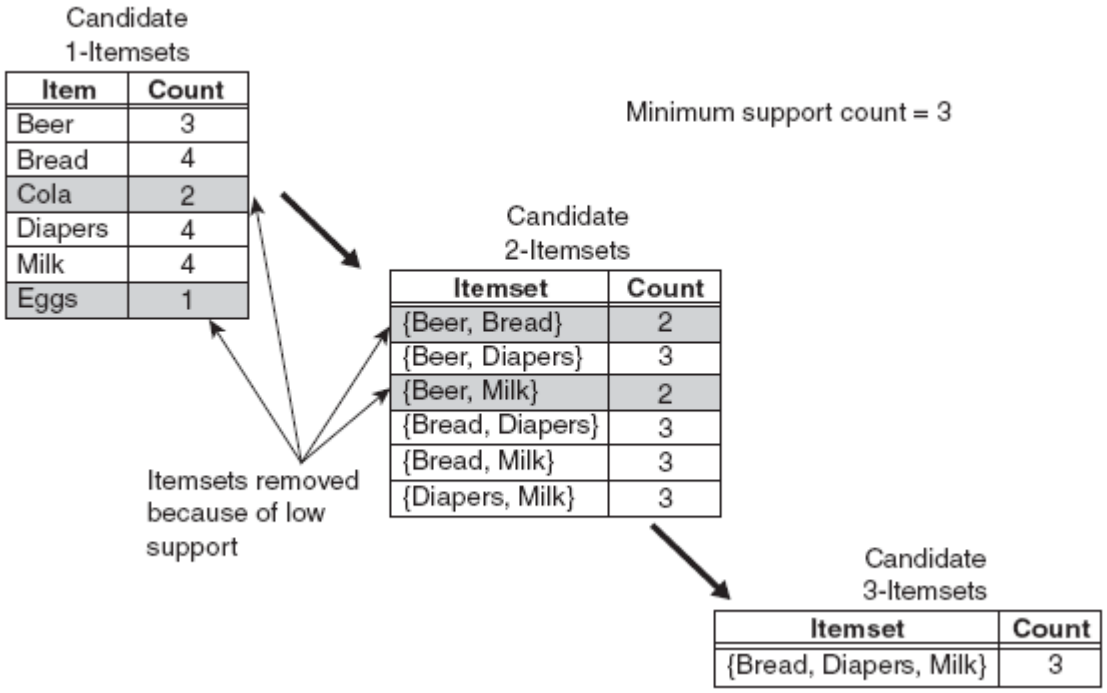


Figure 4.5. Illustration of frequent itemset generation using the Apriori algorithm.

Table 4.1. We assume that the support threshold is 60%, which is equivalent to a minimum support count equal to 3.

Apriori principle ensures that all supersets of the infrequent 1-itemsets must be infrequent. Because there are only four frequent 1-itemsets, the number of candidate 2-itemsets generated by the algorithm is = 6. Two of these six candidates, {Beer, Bread} and {Beer, Milk}, are subsequently found to be infrequent after computing their support values. The remaining four candidates are frequent, and thus will be used to generate candidate 3-itemsets. Without support-based pruning, there are = 20 candidate 3-itemsets that can be formed using the six items given in this example. With the Apriori principle, we only need to keep candidate 3-itemsets whose subsets are frequent. The only candidate that has this property is

{Bread, Diapers,Milk}.

The effectiveness of the Apriori pruning strategy can be shown by counting the number of candidate itemsets generated. A brute-force strategy of enumerating all itemsets (up to size 3) as candidates will produce

$$\binom{6}{1} + \binom{6}{2} + \binom{6}{3} = 6 + 15 + 20 = 41$$

candidates. With the Apriori principle, this number decreases to

$$\binom{6}{1} + \binom{4}{2} + 1 = 6 + 6 + 1 = 13$$

candidates, which represents a 68% reduction in the number of candidate itemsets even in this simple example.

The pseudocode for the frequent itemset generation part of the Apriori algorithm is shown in Algorithm 4.1. Let Ck denote the set of candidate k-itemsets and Fk denote the set of frequent k-itemsets:

• The algorithm initially makes a single pass over the data set to determine the support of each item. Upon completion of this step, the set of all frequent 1-itemsets, F1, will be known (steps 1 and 2).

• Next, the algorithm will iteratively generate new candidate k-itemsets using the frequent (k - 1)-itemsets found in the previous iteration (step 5). Candidate generation is implemented using

a function called apriori-gen, which is described in Section 4.2.3.

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

1: $k = 1$.
2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Find all frequent 1-itemsets}
3: **repeat**
4: $k = k + 1$.
5: C_k = apriori-gen(F_{k-1}). {Generate candidate itemsets}
6: **for** each transaction $t \in T$ **do**
7: C_t = subset(C_k, t). {Identify all candidates that belong to t}
8: **for** each candidate itemset $c \in C_t$ **do**
9: $\sigma(c) = \sigma(c) + 1$. {Increment support count}
10: **end for**
11: **end for**
12: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$. {Extract the frequent k-itemsets}
13: **until** $F_k = \emptyset$
14: Result = $\bigcup F_k$.

• To count the support of the candidates, the algorithm needs to make an additional pass over the data set (steps 6–10). The subset function is used to determine all the candidate itemsets in Ck that are contained in each transaction t.

• After counting their supports, the algorithm eliminates all candidate itemsets whose support counts are less than minsup (step 12).

• The algorithm terminates when there are no new frequent itemsets generated.

Rule Generation

This section describes how to extract association rules efficiently from a given frequent itemset. Each frequent k-itemset, Y , can produce up to 2k-2 association rules, ignoring rules that have empty antecedents or consequents(0→Yor Y → 0). An association rule can be extracted by partitioning the itemset Y into two non-empty subsets, X and Y -X, such that X → Y - X satisfies the confidence threshold. Note that all such rules must have already met the support threshold because they are generated from a frequent itemset.

Example 4 .2. Let X = {1, 2, 3} be a frequent itemset. There are six candidate association rules that can be generated from X: {1, 2} →{3}, {1, 3} →{2}, {2, 3}→{1}, {1}→{2, 3}, {2}→{1, 3}, and {3}→{1, 2}. As each of their support is identical to the support for X, the rules must satisfy the support threshold.

Computing the confidence of an association rule does not require additional scans of the transaction data set. Consider the rule {1, 2} →{3}, which is generated from the frequent itemset X = {1, 2, 3}. The confidence for this rule is σ({1, 2, 3})/σ({1, 2}). Because {1, 2, 3} is frequent, the anti-monotone property of support ensures that {1, 2} must be frequent, too. Since the support counts for both itemsets were already found during frequent itemset generation, there is no need to read the entire data set again.

Confidence-Based Pruning

Unlike the support measure, confidence does not have any monotone property. For example, the confidence for $X \longrightarrow Y$ can be larger, smaller, or equal to the confidence for another rule $\tilde{X} \longrightarrow \tilde{Y}$, where $\tilde{X} \subseteq X$ and $\tilde{Y} \subseteq Y$ (see Exercise 3 on page 405). Nevertheless, if we compare rules generated from the same frequent itemset Y, the following theorem holds for the confidence measure.

Theorem .2. *If a rule $X \longrightarrow Y - X$ does not satisfy the confidence threshold, then any rule $X' \longrightarrow Y - X'$, where X' is a subset of X, must not satisfy the confidence threshold as well.*

To prove this theorem, consider the following two rules: $X' \longrightarrow Y - X'$ and $X \longrightarrow Y - X$, where $X' \subset X$. The confidence of the rules are $\sigma(Y)/\sigma(X')$ and $\sigma(Y)/\sigma(X)$, respectively. Since X' is a subset of X, $\sigma(X') \geq \sigma(X)$. Therefore, the former rule cannot have a higher confidence than the latter rule.

Rule Generation in *Apriori* Algorithm

The Apriori algorithm uses a level-wise approach for generating association rules, where each level corresponds to the number of items that belong to the rule consequent. Initially, all the high-confidence rules that have only one item in the rule consequent are extracted. These rules are then used to generate new candidate rules. For example, if {acd}→{b} and {abd}→{c} are high-confidence rules, then the candidate rule {ad} →{bc} is generated by merging the consequents of both rules. Figure 4.15 shows a lattice structure for the association rules generated from the frequent itemset {a, b, c, d}.

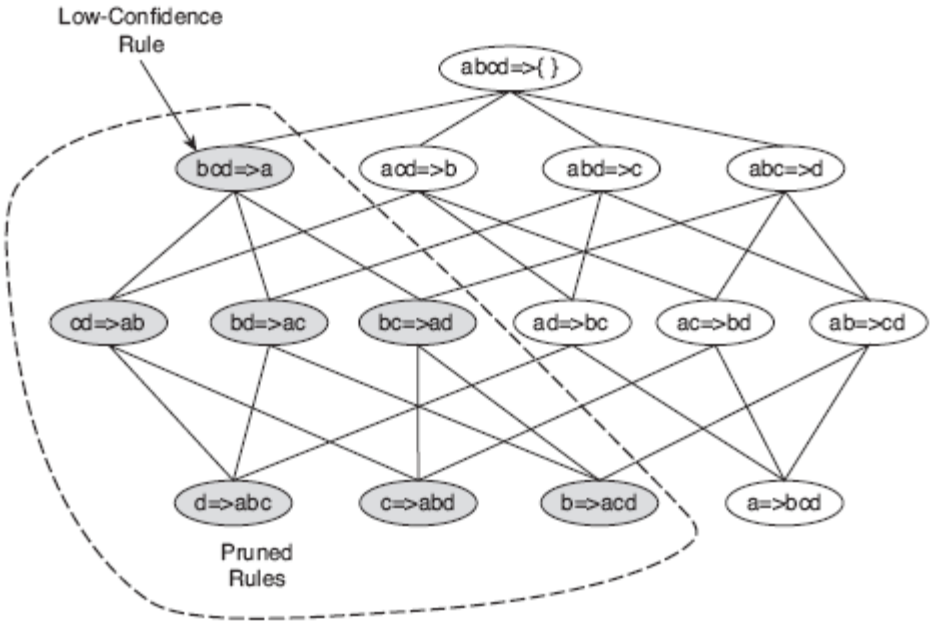


Figure 4.15. Pruning of association rules using the confidence measure.

Suppose the confidence for {bcd} →{a} is low. All the rules containing item a in its consequent, including {cd} →{ab}, {bd}→{ac}, {bc} →{ad}, and {d} →{abc} can be discarded.

The only difference is that, in rule generation, we do not have to make additional passes over the data set to compute the confidence of the candidate rules. Instead, we determine the confidence of each rule by using the support counts computed during frequent itemset generation.

Algorithm 4.2 Rule generation of the *Apriori* algorithm.

```
1: for each frequent k-itemset f_k, k ≥ 2 do
2:   H_1 = {i | i ∈ f_k}        {1-item consequents of the rule.}
3:   call ap-genrules(f_k, H_1.)
4: end for
```

Algorithm 4.3 Procedure ap-genrules(f_k, H_m).

```
1: k = |f_k|   {size of frequent itemset.}
2: m = |H_m|   {size of rule consequent.}
3: if k > m + 1 then
4:   H_{m+1} = apriori-gen(H_m).
5:   for each h_{m+1} ∈ H_{m+1} do
6:     conf = σ(f_k)/σ(f_k - h_{m+1}).
7:     if conf ≥ minconf then
8:       output the rule (f_k - h_{m+1}) ⟶ h_{m+1}.
9:     else
10:      delete h_{m+1} from H_{m+1}.
11:    end if
12:  end for
13:  call ap-genrules(f_k, H_{m+1}.)
14: end if
```

Compact Representation of frequent Itemsets

In practice, the number of frequent itemsets produced from a transaction data set can be very large. It is useful to identify a small representative set of itemsets from which all other frequent itemsets can be derived. Two such representations are presented in this section in the form of maximal and closed frequent itemsets.

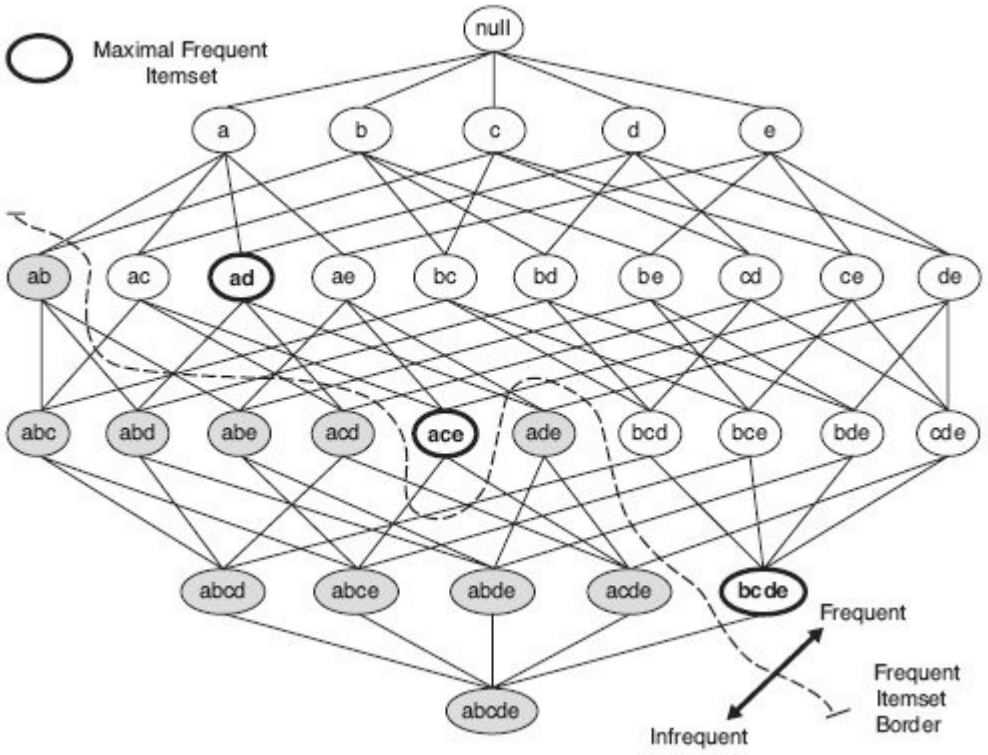


Figure 4.16. Maximal frequent itemset.

Definition 4.3 (Maximal Frequent Itemset). A maximal frequent itemset is defined as a frequent itemset for which none of its immediate supersets are frequent.

To illustrate this concept, consider the itemset lattice shown in Figure 4.16. The itemsets in the lattice are divided into two groups: those that are frequent and those that are infrequent. A frequent itemset border, which is represented by a dashed line, is also illustrated in the diagram. Every itemset located above the border is frequent, while those located below the border (the shaded nodes) are infrequent. Among the itemsets residing near the border, {a, d}, {a, c, e}, and {b, c, d, e} are considered to be maximal frequent itemsets because their immediate supersets are infrequent. An itemset such as {a, d} is maximal frequent because all of its immediate supersets, {a, b, d}, {a, c, d}, and {a, d, e}, are infrequent. In contrast, {a, c} is non-maximal because one of its immediate supersets, {a, c, e}, is frequent.

For example, the frequent itemsets shown in Figure 4.16 can be divided into two groups:

• Frequent itemsets that begin with item a and that may contain items c, d, or e. This group includes itemsets such as {a}, {a, c}, {a, d}, {a, e}, and {a, c, e}.

• Frequent itemsets that begin with items b, c, d, or e. This group includes itemsets such as {b}, {b, c}, {c, d},{b, c, d, e}, etc.

4.4.2 Closed Frequent Itemsets

Closed itemsets provide a minimal representation of itemsets without losing their support information. A formal definition of a closed itemset is presented below.

Definition 4.4 (Closed Itemset). An itemset X is closed if none of its immediate supersets has exactly the same support count as X. Put another way, X is not closed if at least one of its immediate supersets has the same support count as X. Examples of closed itemsets are shown in

Figure 4.17 illustrate the support count of each itemset, we have associated each node (itemset) in the lattice with a list of its corresponding transaction IDs.

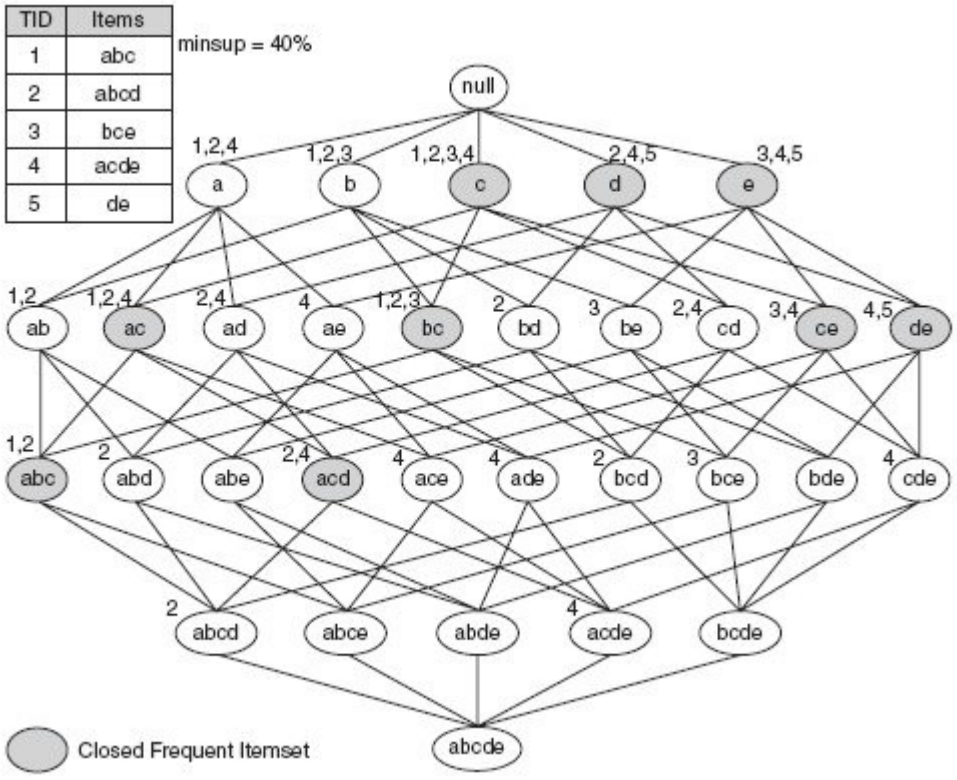


Figure 4.17. An example of the closed frequent itemsets

Definition 4.5 (Closed Frequent Itemset). An itemset is a closed frequent itemset if it is closed

and its support is greater than or equal to minsup. Algorithms are available to explicitly extract closed frequent itemsets from a given data set. Interested readers may refer to the bibliographic notes at the end of this chapter for further discussions of these algorithms. We can use the closed frequent itemsets to determine the support counts for the non-closed Representation of Frequent Itemsets

Algorithm 4.4 Support counting using closed frequent itemsets.

1: Let C denote the set of closed frequent itemsets
2: Let $k_{\max}$ denote the maximum size of closed frequent itemsets
3: $F_{k_{\max}} = \{f | f \in C, |f| = k_{\max}\}$ {Find all frequent itemsets of size $k_{\max}$.}
4: **for** $k = k_{\max} - 1$ downto 1 **do**
5: $F_k = \{f | f \in F_{k+1}, |f| = k\}$ {Find all frequent itemsets of size k.}
6: **for** each $f \in F_k$ **do**
7: **if** $f \in C$ **then**
8: $f.support = \max\{f'.support | f' \in F_{k+1}, f \subset f'\}$
9: **end if**
10: **end for**
11: **end for**

The algorithm proceeds in a specific-to-general fashion, i.e., from the largest to the smallest frequent itemsets. This is because, in order to find the support for a non-closed frequent itemset, the support for all of its supersets must be known.

TID	a_1	a_2	a_3	a_4	a_5	b_1	b_2	b_3	b_4	b_5	c_1	c_2	c_3	c_4	c_5
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
5	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
6	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

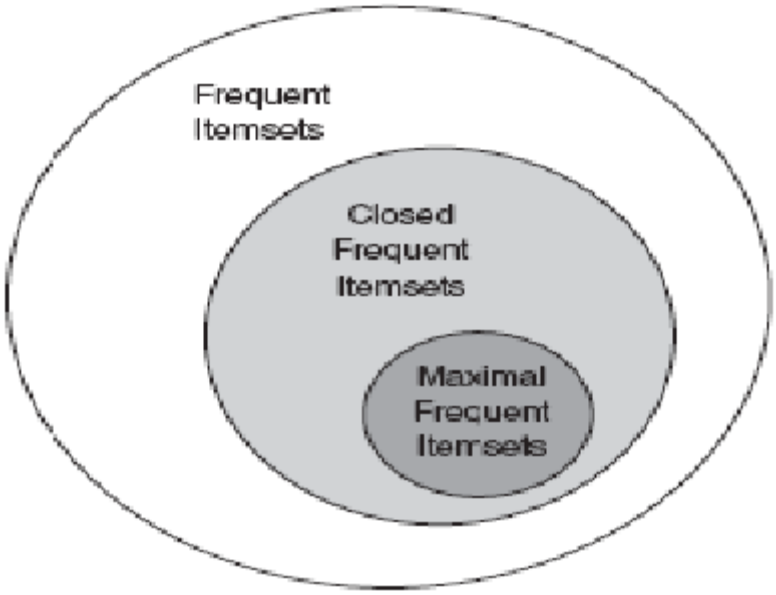


Figure 4.17 relationship among frequent, maximum frequent, and closed frequent itemset.

Closed frequent itemsets are useful for removing some of the redundant association rules. An association rule $X \rightarrow Y$ is redundant if there exists another rule $X \rightarrow Y$, where X is a subset of X and Y is a subset of Y , such that the support and confidence for both rules are identical. In the example shown in Figure 4.17, {b} is not a closed frequent itemset while {b, c} is closed.

The association rule {b} →{d, e} is therefore redundant because it has the same support and confidence as {b, c} →{d, e}. Such redundant rules are not generated if closed frequent itemsets are used for rule generation.

Alternative Methods for Generating Frequent Itemsets

Apriori is one of the earliest algorithms to have successfully addressed the combinatorial explosion of frequent itemset generation. It achieves this by applying the Apriori principle to prune the exponential search space. Despite its significant performance improvement, the algorithm still incurs considerable I/O overhead since it requires making several passes over the transaction data set.

• General-to-Specific versus Specific-to-General: The Apriori algorithm uses a general-to-specific search strategy, where pairs of frequent (k-1)-itemsets are merged to obtain candidate k-itemsets. This general-to-specific search strategy is effective, provided the maximum length of a frequent itemset is not too long. The configuration of frequent item-sets that works best with this strategy is shown in Figure 4.19(a), where the darker nodes represent infrequent itemsets.

Alternatively, a specific-to-general search strategy looks for more specific frequent itemsets first, before finding the more general frequent itemsets. This strategy is use-ful to discover maximal frequent itemsets in dense transactions, where the frequent itemset border is located near the bottom of the lattice, as shown in Figure 4.19(b). The Apriori principle can be applied to prune all subsets of maximal frequent itemsets. Specifically, if a candidate k-itemset is maximal frequent, we do not have to examine any of its subsets of size k - 1. However, if the candidate k-itemset is infrequent, we need to check all of its k - 1 subsets in the next iteration. Another approach is to combine both general-to-specific and specific-to-general search strategies. This bidirectional approach requires more space to

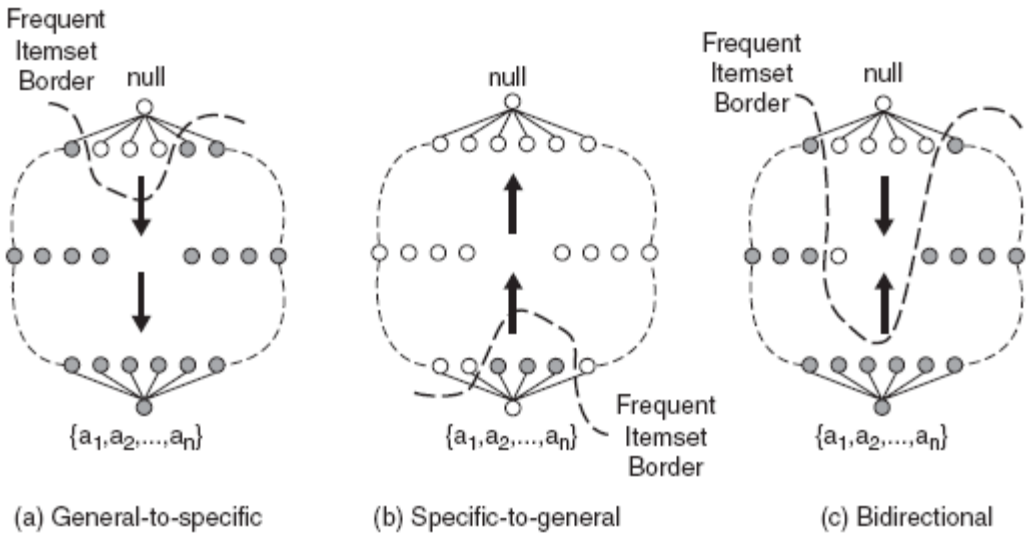


Figure 6.19. General-to-specific, specific-to-general, and bidirectional search.

• **Equivalence Classes:** Another way to envision the traversal is to first partition the lattice into disjoint groups of nodes (or equivalence classes). A frequent itemset generation algorithm searches for frequent itemsets within a particular equivalence class first before moving to another equivalence class. As an example, the level-wise strategy used in the Apriori algorithm can be considered to be partitioning the lattice on the basis of itemset sizes;

• **Breadth-First versus Depth-First:** The Apriori algorithm traverses the lattice in a breadth-first manner, as shown in Figure 6.21(a). It first discovers all the frequent 1-itemsets, followed by the frequent 2-itemsets, and so on, until no new frequent itemsets are generated.

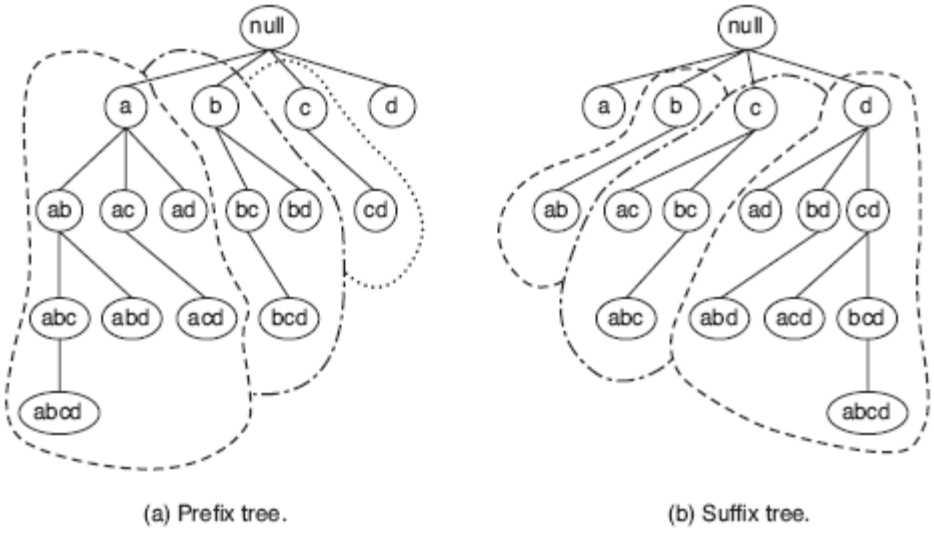


Figure 6.20. Equivalence classes based on prefix and suffix labels of item sets

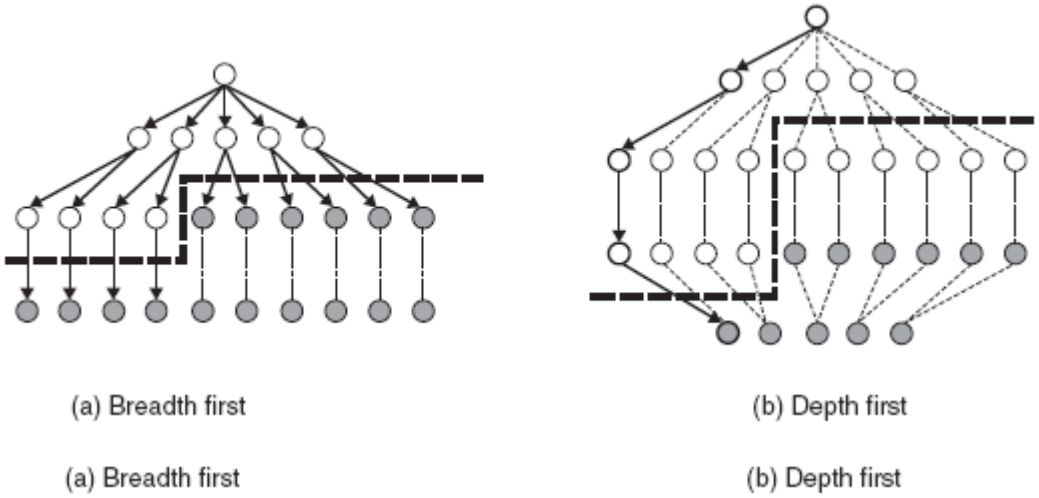


Figure 6.21. Breadth first and depth first traversal

Representation of Transaction Data Set There are many ways to represent a transaction data set. The choice of representation can affect the I/O costs incurred when computing the support of candidate itemsets. Figure 6.23 shows two different ways of representing market basket transactions. The representation on the left is called a **horizontal** data layout, which is adopted by many association rule mining algorithms, including Apriori. Another possibility is to store the list of transaction identifiers (TID-list) associated with each item. Such a representation is known as the

vertical data layout. The support for each candidate itemset is obtained by intersecting the TID-lists of its subset items. The length of the TID-lists shrinks as we progress to larger sized itemsets.

Horizontal Data Layout

TID	Items
1	a,b,e
2	b,c,d
3	c,e
4	a,c,d
5	a,b,c,d
6	a,e
7	a,b
8	a,b,c
9	a,c,d
10	b

Vertical Data Layout

a	b	c	d	e	
1	1	2	2	1	
4	2	3	4	3	
5	5	4	5	6	
6	7	8	9		
7	8	9			
8	10				
9					

Figure 6.23. Horizontal and vertical data format.

However, one problem with this approach is that the initial set of TID-lists may be too large to fit into main memory, thus requiring more sophisticated techniques to compress the TID-lists. We describe another effective approach to represent the data in the next section.

UNIT V & VI

CLASSIFICATION

5.1 Basics

The input data for a classification task is a collection of records. Each record, also known as an instance or example, is characterized by a tuple (x, y), where x is the attribute set and y is a special attribute, designated as the class label. sample data set used for classifying vertebrates into one of the following categories: mammal, bird, fish, reptile, or amphibian. The attribute set includes properties of a vertebrate such as its body temperature, skin cover, method of reproduction ability to fly, and ability to live in water. the attribute set can also contain continuous features. The class label, on the other hand, must be a discrete attribute. This is a key characteristic that distinguishes classification from regression, a predictive modelling task in which y is a continuous attribute. .

Definition 3.1 (Classification). Classification is the task of learning a target function f that maps each attribute set x to one of the predefined class labels y. The target function is also known informally as a classification model.A classification model is useful for the following purposes.

Descriptive Modeling

A classification model can serve as an explanatory tool to distinguish between objects of different classes. For example, it would be useful for both biologists and others to have a descriptive model.

Predictive Modeling

A classification model can also be used to predict the class label of unknown records. As a classification model can be treated as a black box that automatically assigns a class label when presented with the attribute set of an unknown record. Suppose we are given the following characteristics of a creature known as a gila monster: Classification techniques are most suited for predicting or describing data sets with binary or nominal categories. They are less effective for ordinal categories (e.g., to classify a person as a member of high-, medium-, or low-income group) because they do not consider the implicit order among the categories. Other forms of relationships, such as the subclass–super class relationships among categories

5.2 General Approach to Solving a Classification Problem

A classification technique (or classifier) is a systematic approach to building classification models from an input data set. Examples include decision tree classifiers, rule-based classifiers, neural networks, support vector machines and na¨ıve Bayes classifiers. Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data. The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before. Therefore, a key objective of the learning algorithm is to build models with good generalization capability; i.e., models that accurately predict the class labels of previously unknown records.

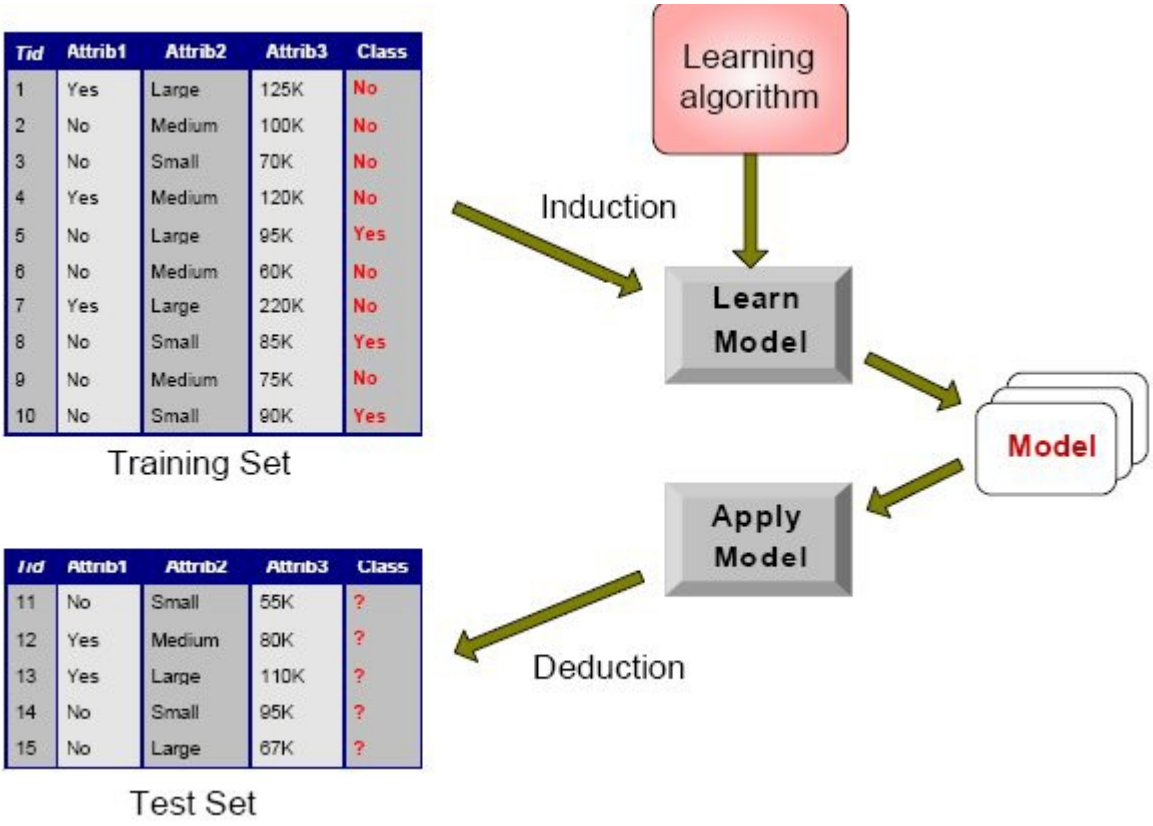


Figure 3.3. General approach for building a classification model.

Figure 3.3 shows a general approach for solving classification problems. First, a training set consisting of records whose class labels are known must be provided. The training set is used to build a classification model, which is subsequently applied to the test set, which consists of records with unknown class labels.

		Predicted Class	
		$Class = 1$	$Class = 0$
Actual Class	$Class = 1$	f_{11}	f_{10}
	$Class = 0$	f_{01}	f_{00}

Table 3.2. Confusion matrix for a 2-class problem

Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model. These counts are tabulated in a table known as a confusion matrix. Table 4.2 depicts the confusion matrix for a binary classification problem. Each entry fij in this table denotes the number of records from class i predicted to be of class j. For instance, $f_{01}$is the number of records from class 0 incorrectly predicted as class 1. Based on the entries in the confusion matrix, the total number of correct predictions made by the model is (f11 + f00) and the total number of incorrect predictions is (f_{10}+ f_{01}). Although a confusion matrix provides the information needed to determine how well a classification model performs, summarizing this information with a single number would make it more convenient to compare

the performance of different models. This can be done using a performance metric such as accuracy, which is defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}. \tag{3.1}$$

Equivalently, the performance of a model can be expressed in terms of its error rate, which is given by the following equation:

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}.$$

Most classification algorithms seek models that attain the highest accuracy, or equivalently, the lowest error rate when applied to the test set.

5.3 Decision Tree Induction

This section introduces a decision tree classifier, which is a simple yet widely used classification technique.

5.3.1 How a Decision Tree Works

To illustrate how classification with a decision tree works, consider a simpler version of the vertebrate classification problem described in the previous section. Instead of classifying the vertebrates into five distinct groups of species, we assign them to two categories: mammals and non-mammals. Suppose a new species is discovered by scientists. How can we tell whether it is a mammal or a non-mammal? One approach is to pose a series of questions about the characteristics of the species. The first question we may ask is whether the species is cold- or warm-blooded. If it is cold-blooded, then it is definitely not a mammal. Otherwise, it is either a bird or a mammal. In the latter case, we need to ask a follow-up question: Do the females of the species give birth to their young? Those that do give birth are definitely mammals, while those that do not are likely to be non-mammals (with the exception of egg-laying mammals such as the platypus and spiny anteater) The previous example illustrates how we can solve a classification problem by asking a series of carefully crafted questions about the attributes of the test record. Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class label of the record. The series of questions and their possible answers can be organized in the form of a decision tree, which is a hierarchical structure consisting of nodes and directed edges. Figure 4.4 shows the decision tree for the mammal classification problem. The tree has three types of nodes:

- **A root node** that has no incoming edges and zero or more outgoing edges.
- **Internal nodes,** each of which has exactly one incoming edge and two or more outgoing edges.
- **Leaf or terminal** nodes, each of which has exactly one incoming edge and no outgoing edges.

In a decision tree, each leaf node is assigned a class label. The non terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate

records that have different characteristics. For example, the root node shown in Figure 4.4 uses

the attribute Body.

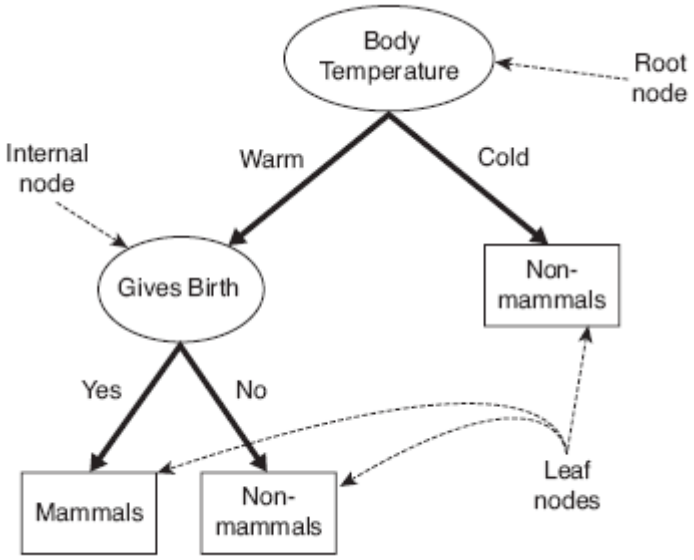


Figure 5.4. A decision tree for the mammal classification problem.

Temperature to separate warm-blooded from cold-blooded vertebrates. Since all cold-blooded vertebrates are non-mammals, a leaf node labeled Non-mammals is created as the right child of the root node. If the vertebrate is warm-blooded, a subsequent attribute, Gives Birth, is used to distinguish mammals from other warm-blooded creatures, which are mostly birds. Classifying a test record is straightforward once a decision tree has been constructed. Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test.

This will lead us either to another internal node, for which a new test condition is applied, or to a leaf node. The class label associated with the leaf node is then assigned to the record. As an illustration, Figure 4.5 traces the path in the decision tree that is used to predict the class label of a flamingo. The path terminates at a leaf node labeled Non-mammals.

How to Build a Decision Tree

There are exponentially many decision trees that can be constructed from a given set of attributes. While some of the trees are more accurate than others, finding the optimal tree is computationally infeasible because of the exponential size of the search space. Nevertheless,

efficient algorithms have been developed to induce a reasonably accurate, albeit suboptimal,

decision tree in a reasonable amount of time.

These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data. One such algorithm is Hunt's algorithm, which is the basis of many existing decision tree induction algorithms, including ID3, C4.5, and CART.

This section presents a high-level discussion of Hunt's algorithm and illustrates some of its design issues.

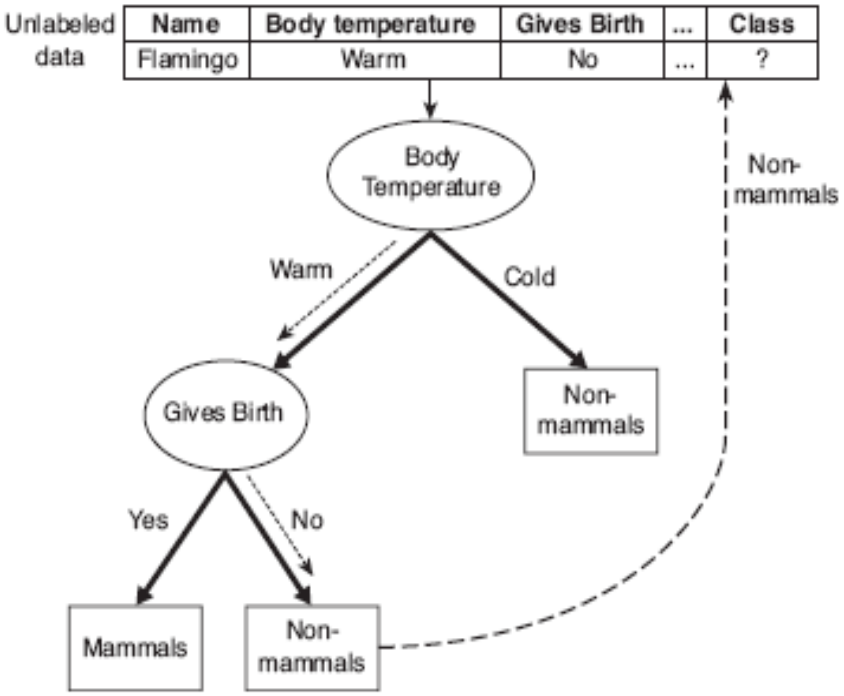


Figure 5.5. Classifying an unlabeled vertebrate.
The dashed lines represent the outcomes of applying various attribute test conditions on the unlabeled vertebrate. The vertebrate is eventually assigned to the Non-mammal class.

Hunt's Algorithm

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets. Let D_t be the set of training records that are associated with node t and y = {$_y$1, y2, . . . , yc} be the class labels. The following is a recursive

definition of Hunt's algorithm.

Step 1: If all the records in Data belong to the same class yt, then t is a leaf node labeled as y_t.

Step 2: If Data contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in Dt are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 3.6. Training set for predicting borrowers who will default on loan payments.

To illustrate how the algorithm works, consider the problem of predicting whether a loan applicant will repay her loan obligations or become delinquent, subsequently defaulting on her loan. A training set for this problem can be constructed by examining the records of previous borrowers. In the example shown in Figure 4.6, each record contains the personal information of a borrower along with a class label indicating whether the borrower has defaulted on loan payments.

The initial tree for the classification problem contains a single node with class label Defaulted = No (see Figure 3.7(a)), which means that most of the borrowers successfully repaid their loans. The tree, however, needs to be redefined since the root node contains records from both classes. The records are

Subsequently divided into smaller subsets based on the outcomes of the Home Owner test condition, as shown in Figure 3.7(b). The justification for choosing this attribute test condition will be discussed later. For now, we will assume that this is the best criterion for splitting the data at this point. Hunt's algorithm is then applied recursively to each child of the root node. From the training set given in Figure 3.6, notice that all borrowers who are home owners successfully repaid their loans. The left child of the root is therefore a leaf node labelled Defaulted = No (see Figure 3.7(b)). For the right child, we need to continue applying the recursive step of Hunt's algorithm until all the

records belong to the same class. The trees resulting from each recursive step are shown in Figures 3.7(c) and (d).

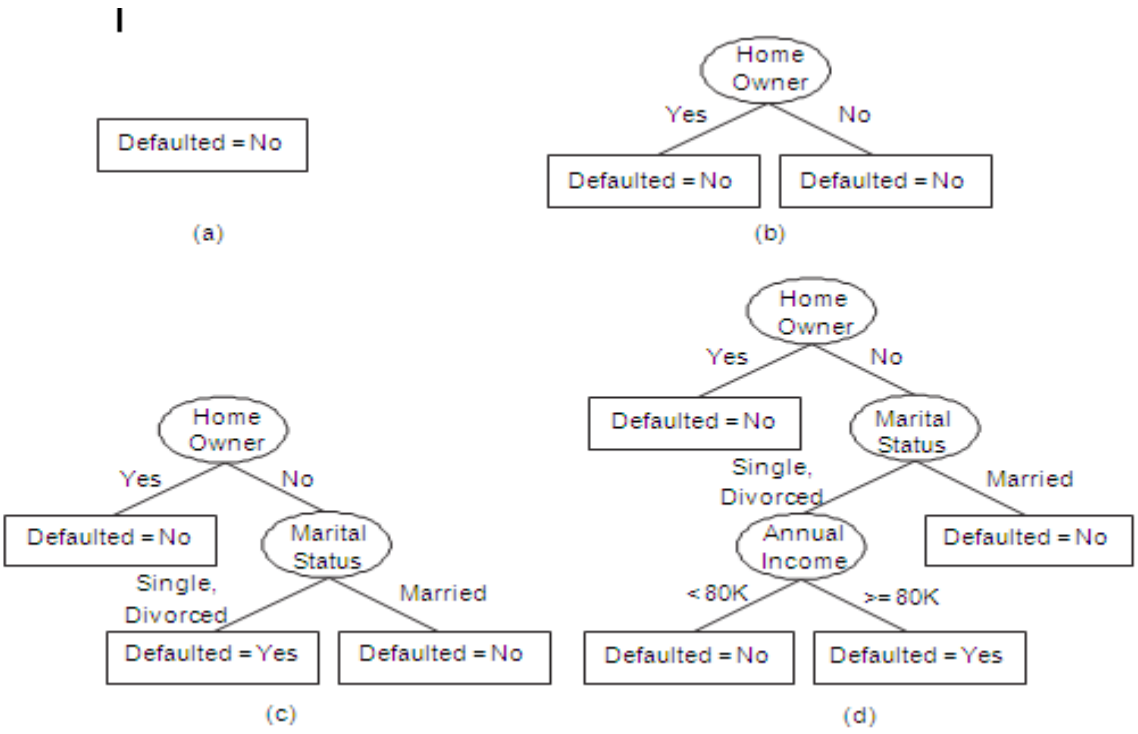


Figure 5 **.7** Hunt's algorithm for inducing decision trees.

Hunt's algorithm will work if every combination of attribute values is present in the training data and each combination has a unique class label. These assumptions are too stringent for use in most practical situations. Additional conditions are needed to handle the following cases:

1. It is possible for some of the child nodes created in Step 2 to be empty; i.e., there are no records associated with these nodes. This can happen if none of the training records have the combination of attribute values associated with such nodes. In this case the node is declared a leaf node with the same class label as the majority class of training records associated with its parent node.

2. In Step 2, if all the records associated with D_t have identical attribute values (except for the class label), then it is not possible to split these records any further. In this case, the node is declared a leaf node with the same class label as the majority class of training records associated with this node.

Design Issues of Decision Tree Induction

A learning algorithm for inducing decision trees must address the following two issues.

a) How should the training records be split? Each recursive step of the tree-growing process must select an attribute test condition to divide the records into smaller subsets. To implement this step, the algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.

b) How should the splitting procedure stop? A stopping condition is needed to terminate the tree-growing process. A possible strategy is to continue expanding a node until either all the records belong to the same class or all the records have identical attribute values. Although both conditions are sufficient to stop any decision tree induction algorithm, other criteria can be imposed to allow the tree-growing procedure to terminate earlier.

Methods for Expressing Attribute Test Conditions

Decision tree induction algorithms must provide a method for expressing an attribute test condition and its corresponding outcomes for different attribute types. Binary Attributes The test condition for a binary attribute generates two potential outcomes, as shown in Figure 3.8.

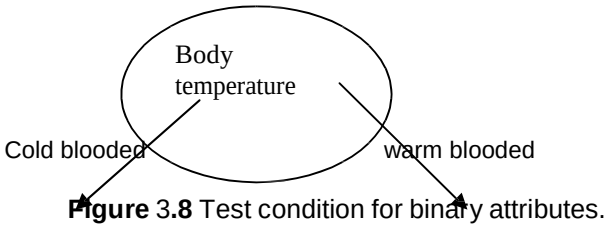


Figure 3.8 Test condition for binary attributes.

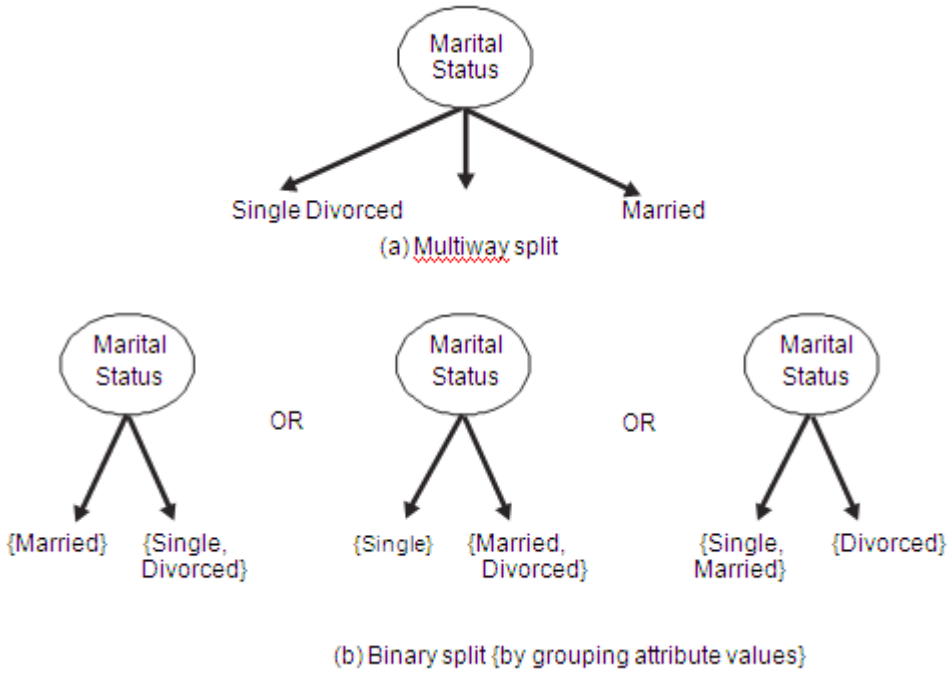


Figure 3.9 Test conditions for nominal attributes.

Nominal Attributes:

Since a nominal attribute can have many values, its test condition can be expressed in two ways, as shown in Figure 3.9. For a multiway split (Figure 3.9(a)), the number of outcomes depends on the number of distinct values for the corresponding attribute. For example, if an attribute such as marital status has three distinct values—single, married, or divorced its test condition will produce a three-way split. On the other hand, some decision tree algorithms, such as CART, produce only binary splits by considering all $2^{k-1} - 1$ ways of creating a binary partition of k attribute values. Figure 3.9(b) illustrates three different ways of grouping the attribute values for marital status into two subsets.

Ordinal Attributes:

Ordinal attributes can also produce binary or multiway splits. Ordinal attribute values can be grouped as long as the grouping does not violate the order property of the attribute values. Figure 3.10 illustrates various ways of splitting training records based on the Shirt Size attribute.

The groupings shown in Figures 3.10(a) and (b) preserve the order among the attribute values, whereas the grouping shown in Figure 3.10(c) violates this property because it combines the attribute values Small and Large into the same partition while Medium and Extra Large are combined into another partition.

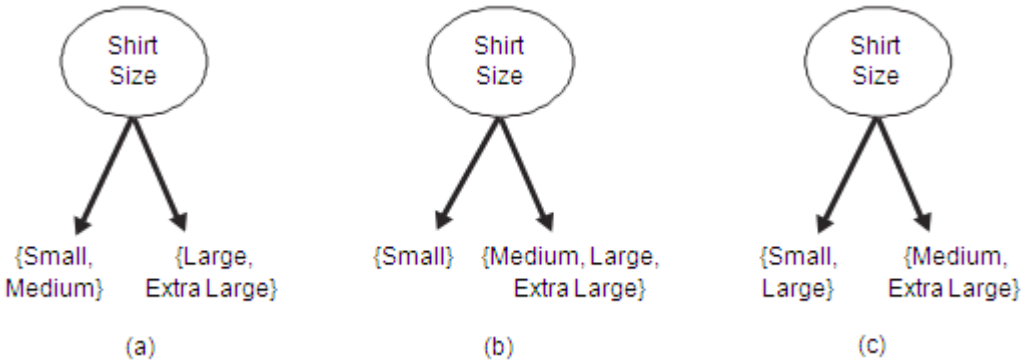


Figure 3.**10** Different ways of grouping ordinal attribute values.

Continuous Attributes:

For continuous attributes, the test condition can be expressed as a comparison test (A < v) or (A ≥ v) with binary outcomes, or a range query with outcomes of the form vi ≤ A < vi+1, for i = 1. . . k.

The difference between these approaches is shown in Figure 3.11. For the binary case, the decision tree algorithm must consider all possible split positions v, and it selects the one that produces the best partition. For the multiway split, the algorithm must consider all possible ranges of continuous values. One approach is to apply the discretization strategies described. After discretization, a new ordinal value will be assigned to ach discretized interval. Adjacent intervals can also be aggregated into wider ranges as long as the order property is preserved.

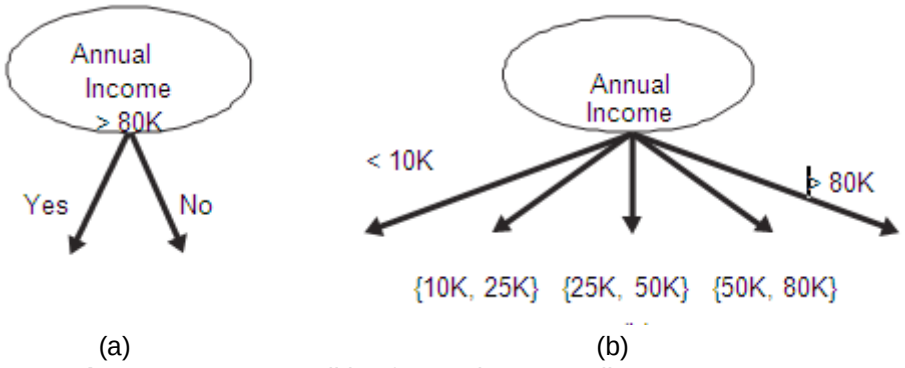


Figure 3.11 Test condition for continuous attributes.

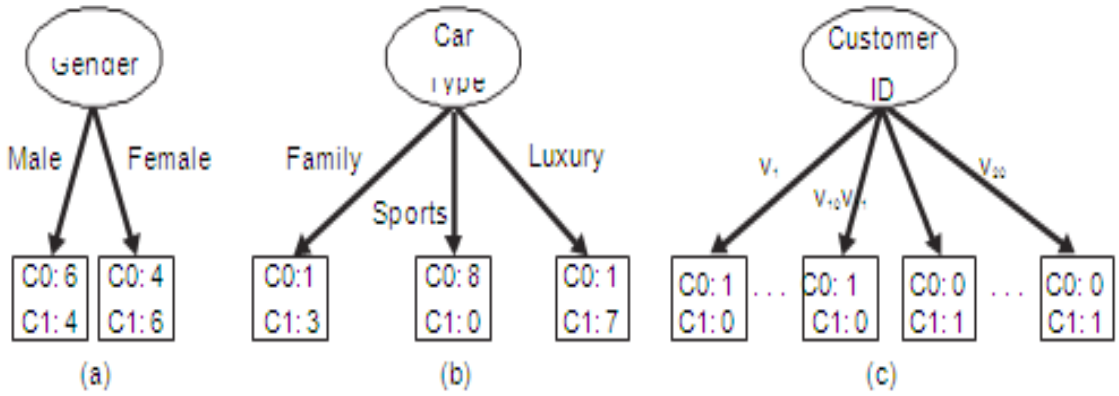


Figure 3.12 Multiway versus binary splits.

5.3.4 Measures for Selecting the Best Split

There are many measures that can be used to determine the best way to split the records. These measures are defined in terms of the class distribution of the records before and after splitting.

Let $p(i|_t)$ denote the fraction of records belonging to class i at a given node t. We sometimes omit the reference to node t and express the fraction as pi. In a two-class problem, the class distribution at any node can be written as (p0, p1), where p1 = 1 − p0. The class distribution before splitting is (0.5, 0.5) because there are an equal number of records from each class. If we split the data using the Gender attribute, then the class distributions of the child nodes are (0.6, 0.4) and (0.4, 0.6), respectively. Although the classes are no longer evenly distributed, the child nodes still contain records from both classes. Splitting on the second attribute, Car Type will result in purer partitions.

The measures developed for selecting the best split are often based on the degree of impurity of the child nodes. The smaller the degree of impurity, the more skewed the class distribution. For example, a node with class distribution (0, 1) has zero impurity, whereas a node with uniform class distribution (0.5, 0.5) has the highest impurity. Examples of impurity measures include

$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t), \quad (3.3)$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2, \quad (3.4)$$

$$\text{Classification error}(t) = 1 - \max [p(i|_t)],$$

(3.5
)

Where c is the number of classes and $0 \log_2 0 = 0$ in entropy calculations.

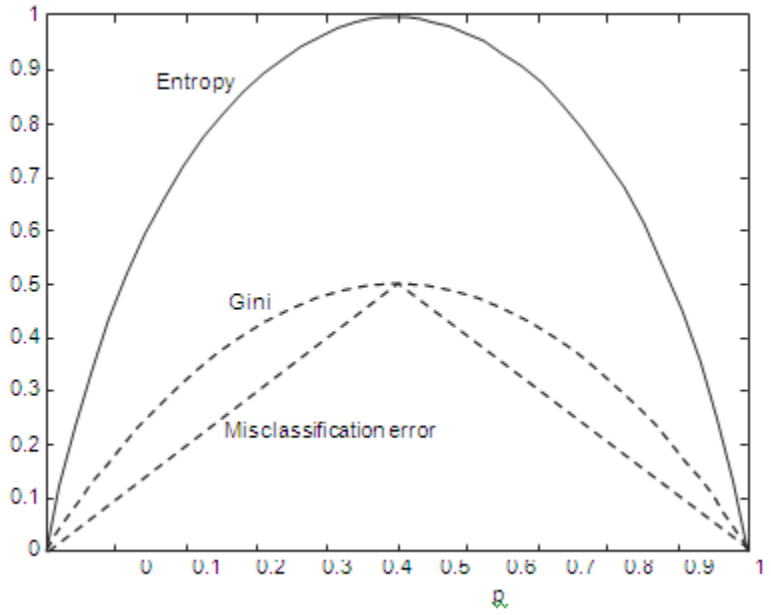


Figure 3.**13** Comparison among the impurity measures for binary classification problems.

Figure 3.13 compares the values of the impurity measures for binary classification problems. p refers to the fraction of records that belong to one of the two classes. Observe that all three measures attain their maximum value when the class distribution is uniform (i.e., when p = 0.5). The minimum values for the measures are attained when all the records belong to the same class (i.e., when p equals 0 or 1). We next provide several examples of computing the different impurity measures.

Node N_1	Count
Class=0	0
Class=1	6

$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$
$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$
$\text{Error} = 1 - \max[0/6, 6/6] = 0$

Node N_2	Count
Class=0	1
Class=1	5

$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$
$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$
$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$

Node N_3	Count
Class=0	3
Class=1	3

$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$
$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$
$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$

Gain Ratio

Impurity measures such as entropy and Gini index tend to favor attributes that have a large

number of distinct values. Comparing the first test condition, Gender, with the second, Car Type, it is easy to see that Car Type seems to provide a better way of splitting the data since it produces purer descendent nodes. However, if we compare both conditions with Customer ID, the latter appears to produce purer partitions.

Yet Customer ID is not a predictive attribute because its value is unique for each record. Even in a less extreme situation, a test condition that results in a

large number of outcomes may not be desirable because the number of record associated with each partition is too small to enable us to make any reliable predictions.

There are two strategies for overcoming this problem. The first strategy is to restrict the test conditions to binary splits only. This strategy is employed by decision tree algorithms such as CART. Another strategy is to modify the splitting criterion to take into account the number of outcomes produced by the attribute test condition. For example, in the C4.5 decision tree algorithm,a splitting criterion known as gain ratio is used to determine the goodness of a split. This criterion is defined as follows:

$$\text{GAIN RATIO} = \frac{\Delta \text{INFO}}{\text{Split info}}$$

Here, Split Info = $-\sum_{i=1}^{k} P(v_i) \log_2 P(v_i)$ and k is the total number of splits. For example, if each attribute value has the same number of records, then $\forall_i$: P (vi) = 1/k and the split information would be equal to $\log_2 k$. This example suggests that if an attribute produces a large number of splits, its split information will also be large, which in turn reduces its gain ratio.

Algorithm for Decision Tree Induction

A skeleton decision tree induction algorithm called Tree Growth is shown in Algorithm 4.1. The input to this algorithm consists of the training records E and the attribute set F. The algorithm works by recursively selecting the best attribute to split the data (Step 7) and expanding the leaf nodes of the tree.

Algorithm 4.1 A skeleton decision tree induction algorithm.

```
TreeGrowth (E, F)
 1: if stopping_cond(E,F) = true then
 2:    leaf = createNode().
 3:    leaf.label = Classify(E).
 4:    return leaf.
 5: else
 6:    root = createNode().
 7:    root.test_cond = find_best_split(E, F).
 8:    let V = {v|v is a possible outcome of root.test_cond }.
 9:    for each v ∈ V do
10:       E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:       child = TreeGrowth(E_v, F).
12:       add child as descendent of root and label the edge (root → child) as v.
13:    end for
14: end if
15: return root.
```

(Steps 11 and 12) until the stopping criterion is met (Step 1). The details of this algorithm are explained below:

1. The createNode() function extends the decision tree by creating a new node. A node in the decision tree has either a test condition, denoted as node. Test cond, or a class label, denoted as node. Label.

2. The find best split () function determines which attribute should be selected as the test condition for splitting the training records. As previously noted, the choice of test condition depends on which impurity measure is used to determine the goodness of a split. Some widely used measures include entropy, the Gini index, and the χ^2 statistic.

3. The Classify() function determines the class label to be assigned to a leaf node. For each leaf node t, let p(i|t) denote the fraction of training records from class i associated with the node t. In most cases, the leaf node is assigned to the class that has the majority number of training records:

$$\text{leaf.label} = \text{argmax } p(i|_t),i$$

where the argmax operator returns the argument i that maximizes the expression $p(i|_t)$. Besides providing the information needed to determine the class label of a leaf node, the fraction p(i|t) can also be used to estimate the probability that a record assigned to the leaf node t belongs to class i.

4. The stopping cond() function is used to terminate the tree-growing process by testing whether all the records have either the same class label or the same attribute values. Another way to terminate the recursive function is to test whether the number of records has fallen below some

Minimum threshold.

After building the decision tree, a tree-pruning step can be performed to reduce the size of the decision tree. Decision trees that are too large are susceptible to a phenomenon known as overfitting. Pruning helps by trimming the branches of the initial tree in a way that improves the generalization capability of the decision tree.

Characteristics of Decision Tree Induction

The following is a summary of the important characteristics of decision tree induction algorithms.

1. Decision tree induction is a nonparametric approach for building classification models. In other words, it does not require any prior assumptions regarding the type of probability distributions satisfied by the class and other attributes.

2. Finding an optimal decision tree is an NP-complete problem. Many decision tree algorithms employ a heuristic-based approach to guide their search in the vast hypothesis space. For example, the algorithm presented in Section 3.3.5 uses a greedy, top-down, recursive partitioning strategy for growing a decision tree.

3. Techniques developed for constructing decision trees are computationally inexpensive, making it possible to quickly construct models even when the training set size is very large. Furthermore, once a decision tree has been built, classifying a test record is extremely fast, with a worst-case complexity of O(w), where w is the maximum depth of the tree.

4. Decision trees, especially smaller-sized trees, are relatively easy to interpret. The accuracies of the trees are also comparable to other classification techniques for many simple data sets.

5. Decision trees provide an expressive representation for learning discrete valued functions. However, they do not generalize well to certain types of Boolean problems. One notable example

is the parity function, whose value is 0 (1) when there is an odd (even) number of Boolean attributes with the value True. Accurate modeling of such a function requires a full decision tree with 2d nodes, where d is the number of Boolean attributes

6. Decision tree algorithms are quite robust to the presence of noise, especially when methods for avoiding overfitting, are employed.

7. The presence of redundant attributes does not adversely affect the accuracy of decision trees. An attribute is redundant if it is strongly correlated with another attribute in the data. One of the two redundant attributes will not be used for splitting once the other attribute has been chosen. However, if the data set contains many irrelevant attributes, i.e., attributes that are not useful for the classification task, then some of the irrelevant attributes may be accidently chosen during the tree-growing process, which results in a decision tree that is larger than necessary.

8. Since most decision tree algorithms employ a top-down, recursive partitioning approach, the number of records becomes smaller as we traverse down the tree. At the leaf nodes, the number of records may be too small to make a statistically significant decision about the class representation of the nodes. This is known as the data fragmentation problem. One possible solution is to disallow further splitting when the number of records falls below a certain threshold.

9. A subtree can be replicated multiple times in a decision tree, This makes the decision tree more complex than necessary and perhaps more difficult to interpret. Such a situation can arise from decision tree implementations that rely on a single attribute test condition at each internal node. Since most of the decision tree algorithms use a divide-and-conquer partitioning strategy, the same test condition can be applied to different parts of the attribute space, thus leading to the subtree replication problem.

10. The test conditions described so far in this chapter involve using only a single attribute at a time. As a consequence, the tree-growing procedure can be viewed as the process of partitioning the attribute space into disjoint regions until each region contains records of the same class. The border between two neighboring regions of different classes is known as a decision boundary. Constructive induction provides another way to partition the data into homogeneous, nonrectangular regions

5.4 Rule-Based Classification

In this section, we look at rule-based classifiers, where the learned model is represented as a set of IF-THEN rules. We first examine how such rules are used for classification. We then study ways in which they can be generated, either froma decision tree or directly from the training data using a *sequential covering algorithm.*

5.4.1 Using IF-THEN Rules for Classification

Rules are a good way of representing information or bits of knowledge. A rule-based classifier uses a set of IF-THEN rules for classification. An IF-THEN rule is an expression of the form

IF *condition* THEN *conclusion.*

An example is rule *R*1,

R1: IF *age = youth* AND *student = yes* THEN *buys computer = yes*.

The "IF"-part (or left-hand side) of a rule is known as the rule antecedent or precondition. The "THEN"-part (or right-hand side) is the rule consequent. In the rule antecedent, the condition consists of one or more *attribute tests* (such as *age = youth*, and *student = yes*) that are logically ANDed. The rule's consequent contains a class prediction (in this case, we are predicting whether a customer will buy a computer). *R*1 can also be written as

R1: (*age = youth*) ^ (*student = yes*))(*buys computer = yes*).

If the condition (that is, all of the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule antecedent is satisfied (or simply, that the rule is satisfied) and that the rule covers the tuple.

A rule *R* can be assessed by its coverage and accuracy. Given a tuple, ***X***, from a class labeled data set,*D*, let *ncovers* be the number of tuples covered by *R*; *ncorrect* be the number of tuples correctly classified by *R*; and j*D*j be the number of tuples in *D*. We can define the coverage and accuracy of *R* as

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

That is, a rule's coverage is the percentage of tuples that are covered by the rule (i.e., whose attribute values hold true for the rule's antecedent). For a rule's accuracy, we look at the tuples that it covers and see what percentage of them the rule can correctly classify.

Rule Extraction from a Decision Tree

Decision tree classifiers are a popular method of classification—it is easy to understand how decision trees work and they are known for their accuracy. Decision trees can become large and difficult to interpret. In this subsection, we look at how to build a rule based classifier by extracting IF-THEN rules from a decision tree. In comparison with a decision tree, the IF-THEN rules may be easier for humans to understand, particularly if the decision tree is very large.

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).

Example 3.4 Extracting classification rules from a decision tree. The decision tree of Figure 6.2 can be converted to classification IF-THEN rules by tracing the path from the root node to each leaf node in the tree.

*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*

A disjunction (logical OR) is implied between each of the extracted rules. Because the rules are extracted directly from the tree, they are mutually exclusive and exhaustive. By *mutually exclusive*, this means that we cannot have rule conflicts here because no two rules will be triggered for the same tuple. (We have one rule per leaf, and any tuple can map to only one leaf.) By *exhaustive*, there is one rule for each possible attribute-value combination, so that this set of rules does not require a default rule. Therefore, the order of the rules does not matter—they are *unordered*.

Since we end up with one rule per leaf, the set of extracted rules is not much simpler than the corresponding decision tree! The extracted rules may be even more difficult to interpret than the original trees in some cases. As an example, Figure 6.7 showed decision trees that suffer from subtree repetition and replication. The resulting set of rules extracted can be large and difficult to follow, because some of the attribute tests may be irrelevant or redundant. So, the plot thickens. Although it is easy to extract rules from a decision tree, we may need to do some more work by pruning the resulting rule set.

Rule Induction Using a Sequential Covering Algorithm

IF-THEN rules can be extracted directly from the training data (i.e., without having to generate a decision tree first) using a sequential covering algorithm. The name comes from the notion that the rules are learned *sequentially* (one

at a time), where each rule for a given class will ideally *cover* many of the tuples of that class (and hopefully none of the tuples of other classes). Sequential covering algorithms are the most widely used approach to mining disjunctive sets of classification rules, and form the topic of this subsection. Note that in a newer alternative approach, classification rules can be generated using *associative classification algorithms*, which search for attribute-value pairs that occur frequently in the data. These pairs may form association rules, which can be analyzed and used in classification. Since this latter approach is based on association rule mining (Chapter 5), we prefer to defer its treatment until later, in Section 6.8. There are many sequential covering algorithms. Popular variations include AQ, CN2, and the more recent, RIPPER. The general strategy is as follows. Rules are learned one at a time. Each time a rule is learned, the tuples covered by the rule are removed, and the process repeats on the remaining tuples. This sequential learning of rules is in contrast to decision tree induction. Because the path to each leaf in a decision tree corresponds to a rule, we can consider decision tree induction as learning a set of rules *simultaneously*.

A basic sequential covering algorithm is shown in Figure 6.12. Here, rules are learned for one class at a time. Ideally, when learning a rule for a class, *Ci*, we would like the rule to cover all (or many) of the training tuples of class *C* and none (or few) of the tuples from other classes. In this way, the rules learned should be of high accuracy. The rules need not necessarily be of high coverage.

Algorithm: Sequential covering. Learn a set of IF-THEN rules for classification.
Input: D, a data set class-labeled tuples;
Att vals, the set of all attributes and their possible values.
Output: A set of IF-THEN rules.
Method:
(1) Rule set = fg; // initial set of rules learned is empty
(2) for each class c do
(3) repeat
(4) Rule = Learn One Rule(D, Att vals, c);
(5) remove tuples covered by Rule from D;
(6) until terminating condition;
(7) Rule set = Rule set +Rule; // add new rule to rule set
(8) endfor
(9) return Rule Set;

This is because we can have more than one rule for a class, so that different rules may cover different tuples within the same class. The process continues until the terminating condition is met, such as when there are no more training tuples or the quality of a rule returned is below a user-specified threshold. The Learn One Rule procedure finds the "best" rule for the current class, given the current set of training tuples. "How are rules learned?" Typically, rules are

grown in a general-to-specific manner .We can think of this as a beam search, where we start off with an empty rule and then gradually keep appending attribute tests to it. We append by adding the attribute test as a logical conjunct to the existing condition of the rule antecedent. Suppose our training set, D, consists of loan application data. Attributes regarding each applicant include their age, income, education level, residence, credit rating, and the term of the loan. The classifying attribute is loan decision, which indicates whether a

loan is accepted (considered safe) or rejected (considered risky). To learn a rule for the class "accept," we start off with the most general rule possible, that is, the condition of the rule antecedent is empty. The rule is:

IF THEN loan decision = accept.

We then consider each possible attribute test that may be added to the rule. These can be derived from the parameter Att vals, which contains a list of attributes with their associated values. For example, for an attribute-value pair (att, val), we can consider attribute tests such as att = val, att _ val, att > val, and so on. Typically, the training data will contain many attributes, each of which may have several possible values. Finding an optimal rule set becomes computationally explosive. Instead, Learn One Rule adopts a greedy depth-first strategy. Each time it is faced with adding a new attribute test (conjunct) to the current rule, it picks the one that most improves the rule quality,

based on the training samples. We will say more about rule quality measures in a minute. For the moment, let's say we use rule accuracy as our quality measure. Getting back to our example with Figure 6.13, suppose Learn One Rule finds that the attribute test income = high best improves the accuracy of our current (empty) rule. We append it to the condition, so that the current rule becomes

IF income = high THEN loan decision = accept. Each time we add an attribute test to a rule, the resulting rule should cover more of the "accept" tuples. During the next iteration, we again consider the possible attribute tests and end up selecting credit rating = excellent. Our current rule grows to become

IF income = high AND credit rating = excellent THEN loan decision = accept.

The process repeats, where at each step, we continue to greedily grow rules until the resulting rule meets an acceptable quality level.

Rule Pruning

Learn One Rule does not employ a test set when evaluating rules. Assessments of rule quality as described above are made with tuples from the original training data. Such assessment is optimistic because the rules will likely overfit the data. That is, the rules may perform well on the training data, but less well on subsequent data. To compensate for this, we can prune the rules. A rule is pruned by removing a conjunct (attribute test). We choose to prune a rule, R, if the pruned version of R has greater quality, as assessed on an independent set of tuples. As in decision tree pruning, we

refer to this set as a pruning set. Various pruning strategies can be used, such as the pessimistic pruning approach described in the previous section. FOIL uses a simple yet effective method. Given a rule, R,

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg},$$

where pos and neg are the number of positive and negative tuples covered by R, respectively. This value will increase with the accuracy of R on a pruning set. Therefore, if the FOIL Prune value is higher for the pruned version of R, then we prune R. By convention, RIPPER starts with the most recently added conjunct when considering pruning. Conjuncts are pruned one at a time as long as this results in an improvement.

Rule based classifier

Classify records by using a collection of "if…then…" rules

Rule: (*Condition*) → *y*

– Where *Condition* is a conjunction of attributes *y* is the class label

– *LHS*: rule antecedent or condition

– *RHS*: rule consequent

– Examples of classification rules:

(Blood Type=Warm) I (Lay Eggs=Yes) → Birds

(Taxable Income < 50K) I (Refund=Yes) → Evade=No

R1: (Give Birth = no) I (Can Fly = yes) → Birds

R2: (Give Birth = no) I (Live in Water = yes) → Fishes

R3: (Give Birth = yes) I (Blood Type = warm) → Mammals

R4: (Give Birth = no) I (Can Fly = no) → Reptiles

R5: (Live in Water = sometimes) → Amphibians

Application of Rule-Based Classifier

A rule *r* covers an instance **x** if the attributes of the instance satisfy the condition of the rule.

R1: (Give Birth = no) I (Can Fly = yes) → Birds

R2: (Give Birth = no) I (Live in Water = yes) → Fishes

R3: (Give Birth = yes) I (Blood Type = warm) → Mammals

R4: (Give Birth = no) I (Can Fly = no) → Reptiles

R The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

Rule Coverage and Accuracy

Coverage of a rule: Fraction of records that satisfy the antecedent of a rule

Accuracy of a rule: Fraction of records satisfy both the antecedent and consequent of a rule table

Characteristics of Rule-Based Classifier

Mutually exclusive rules: Classifier contains mutually exclusive rules if the rules are independent of each other every record is covered by at most one rule.

Exhaustive rules: Classifier has exhaustive coverage if accounts for every possible combination of attribute values each record is covered by at least one rule.

5.5 Nearest-Neighbor Classifiers

Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuplewith training tuples that are similar to it. The training tuples are described by *n* attributes. Each tuple represents a point in an *n*-dimensional space. In this way all of the training tuples are stored in an *n*-dimensional pattern space. When given an unknown tuple, a ***k***-nearest-neighbor classifier searches the pattern space for the *k* training tuples that are closest to the unknown tuple. These *k* training tuples are the *k* "nearest neighbors" of the unknown tuple. "Closeness" is defined in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points or tuples, say, $\boldsymbol{X}1 = (x11, x12, : : : , x1n)$ and $\boldsymbol{X}2 = (x21, x22, : : : , x2n)$, is

In other words, for each numeric attribute, we take the difference between the corresponding values of that attribute in tuple X1 and in tuple X2, square this difference, and accumulate it. The square root is taken of the total accumulated distance count. Typically, we normalize the values of each attribute before using Euclid's Equation. This helps prevent attributes with initially large ranges (such as income) from outweighing attributes with initially smaller ranges (such as binary attributes). Min-max normalization, for example, can be used to transform a value v of a numeric attribute A to v0 in the range [0, 1] by computing

$$v' = \frac{v - min_A}{max_A - min_A},$$

where minA and maxA are the minimum and maximum values of attribute A. Chapter 2 describes other methods for data normalization as a form of data transformation. For k-nearest-neighbor classification, the unknown tuple is assigned the most common class among its k nearest neighbors. When k = 1, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space. Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple. In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown tuple. "But how can distance be computed for attributes that not numeric, but categorical, such as color?" The above discussion assumes that the attributes used to describe the tuples are all numeric. For categorical attributes, a simple method is to compare the corresponding value of the attribute in tuple X1 with that in tuple X2. If the two are identical (e.g., tuples X1 and X2 both have the color blue), then the difference between the two is taken as 0. If the two are different (e.g., tuple X1 is blue but tuple X2 is red), then the difference is considered to be 1. Other methods may incorporate more sophisticated schemes for differential grading (e.g., where a larger difference score is assigned, say, for blue and white than for blue and black).

"What about missing values?" In general, if the value of a given attribute A is missing in tuple X1 and/or in tuple X2, we assume the maximum possible difference. Suppose that each of the attributes have been mapped to the range [0, 1]. For categorical attributes, we take the difference value to be 1 if either one or both of the corresponding values of A are missing. If A is numeric and missing fromboth tuples X1 and X2, then the difference is also taken to be 1.

"How can I determine a good value for k, the number of neighbors?" This can be determined experimentally. Starting with k = 1, we use a test set to estimate the error rate of the classifier. This process can be repeated each time by incrementing k to allow for one more neighbor. The k value that gives the minimum error rate may be selected. In general, the larger the number of training tuples is, the larger the value of k will be (so that classification and prediction decisions can be based on a larger portion of the stored tuples). As the number of training tuples approaches infinity and k =1, the error rate can be no worse then twice the Bayes error rate (the latter being the theoretical minimum).

5.6. *Introduction to Bayesian Classification*

The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. Assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems. This Classification is named after Thomas Bayes (1702-1761), who proposed the ayesTheorem.Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian

Classification provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities forhypothesis and it is robust to noise in input data.Uses of Naive Bayes classification: 1. Naive Bayes text classification) The Bayesian classification is used as a probabilistic learning method (Naive Bayes text classification). Naive Bayes classifiers are among the most successful known algorithms forlearning to classify text documents.

2. Spam filtering (http://en.wikipedia.org/wiki/Bayesian_spam_filtering)Spam filtering is the best known use of Naive Bayesian text classification. It makes use of anaive Bayes classifier to identify spam e-mail.Bayesian spam filtering has become a popular mechanism to distinguish illegitimate spamemail from legitimate email (sometimes called "ham" or "bacn").[4] Many modern mail clientsimplement Bayesian spam filtering. Users can also install separate email filtering programs.

UNIT – 7

Clustering Techniques

7.1Overview

Clustering and classification are both fundamental tasks in Data Mining. Classification is used mostly as a supervised learning method, clustering for unsupervised learning (some clustering models are for both). The goal of clus- tering is descriptive, that of classification is predictive (Veyssieres and Plant, 1998). Since the goal of clustering is to discover a new set of categories, the new groups are of interest in themselves, and their assessment is intrinsic. In classification tasks, however, an important part of the assessment is extrinsic, since the groups must reflect some reference set of classes. *"Understanding our world requires conceptualizing the similarities and differences between the entities that compose it"* (Tyron and Bailey, 1970).

Clustering groups data instances into subsets in such a manner that similar instances are grouped together, while different instances belong to differ-ent groups. The instances are thereby organized into an efficient representation that characterizes the population being sampled. Formally, the clustering structure is represented as a set of subsets $C = C_1, \ldots, C_k$ of S, such that: $\bigcup_{k} C_i = S$ and $C_i \cap C_j = \emptyset$ for $i = j$. Consequently, any instance in S belongs to exactly one and only one subset. Clustering of objects is as ancient as the human need for describing the salient characteristics of men and objects and identifying them with a type. Therefore, it embraces various scientific disciplines: from mathematics and statistics to biology and genetics, each of which uses different terms to describethe topologies formed using this analysis. From biological "taxonomies", to medical "syndromes" and genetic "genotypes" to manufacturing "group tech- nology" — the problem is identical: forming categories of entities and assigning individuals to the proper groups within it.

Distance Measures

Since clustering is the grouping of similar instances/objects, some sort of measure that can determine whether two objects are similar or dissimilar is required. There are two main type of measures used to estimate this relation: distance measures and similarity measures.

Many clustering methods use distance measures to determine the similarity or dissimilarity between any pair of objects. It is useful to denote the distance between two instances x_i and x_j as: $d(x_i,x_j)$. A valid distance measure should be symmetric and obtains its minimum value (usually zero) in case of identical vectors. The distance measure is called a metric distance measure if it also satisfies the following properties:

1. Triangle inequality $d(x_i,x_k) \leq d(x_i,x_j) + d(x_j,x_k) \quad \forall x_i,x_j,x_k \in S$.

2. $d(x_i, x_j) = 0 \; \mathbf{I} \; x_i = x_j \qquad Ix_i, x_j \; \mathbf{I} \; S.$

7.2 Features of cluster analysis

In this section we describe the most well-known clustering algorithms. The main reason for having many clustering methods is the fact that the notion of "cluster" is not precisely defined (Estivill-Castro, 2000). Consequently many

clustering methods have been developed, each of which uses a different in- diction principle. Farley and Raftery (1998) suggest dividing the clustering methods into two main groups: hierarchical and partitioning methods. Han

and Kamber (2001) suggest categorizing the methods into additional three main categories: *density-based methods*, *model-based clustering* and *grid-based methods*. An alternative categorization based on the induction principle of the various clustering methods is presented in (Estivill-Castro, 2000).

Types of Cluster Analysis Methods, Partitional Methods, Hierarchical Methods, Density Based Methods

Hierarchical Methods

These methods construct the clusters by recursively partitioning the instances in either a top-down or bottom-up fashion. These methods can be sub- divided as following:

Agglomerative hierarchical clustering — Each object initially represents a cluster of its own. Then clusters are successively merged until the desired cluster structure is obtained.

Divisive hierarchical clustering — All objects initially belong to one cluster. Then the cluster is divided into sub-clusters, which are successively divided into their own sub-clusters. This process continues until the desired cluster structure is obtained.

The result of the hierarchical methods is a dendrogram, representing the nested n grouping of objects and similarity levels at which groupings change. A clustering of the data objects is obtained by cutting the dendrogram at the desired similarity level.

The merging or division of clusters is performed according to some similar-ity measure, chosen so as to optimize some criterion (such as a sum of squares).The hierarchical clustering methods could be further divided according to the manner that the similarity measure is calculated (Jain *et al*., 1999):

Single-link clustering (also called the connectedness, the minimum method or the nearest neighbor method) — methods that consider the distance between two clusters to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, the similarity between a pair of clusters is considered to be equal to the greatest similarity from any member of one cluster to any member of the other cluster (Sneath and Sokal, 1973).

Complete-link clustering (also called the diameter, the maximum method or the furthest neighbor method) - methods that consider the distance between two clusters to be equal to the longest distance from any member of one cluster to any member of the other cluster (King,1967).

Average-link clustering (also called minimum variance method) - meth-ods that consider the distance between two clusters to be equal to the average distance from any member of one cluster to any member of the other cluster. Such clustering algorithms may be found in (Ward, 1963)and (Murtagh, 1984).The disadvantages of the single-link clustering and the average-link clustering can be summarized as follows (Guha *et al*., 1998):Single-link clustering has a drawback known as the "chaining effect": Afew points that form a bridge between two clusters cause the single-link clustering to unify these two clusters into one. Average-link clustering may cause elongated clusters to split and for portions of neighboring elongated clusters to merge.

The complete-link clustering methods usually produce more compact clusters and more useful hierarchies than the single-link clustering methods, yet the single-link methods are more versatile. Generally, hierarchical methods are

characterized with the following strengths: Versatility — The single-link methods, for example, maintain good performance on data sets containing non-isotropic clusters, including well-separated, chain-like and concentric clusters.

Multiple partitions — hierarchical methods produce not one partition, but multiple nested partitions, which allow different users to choose different partitions, according to the desired similarity level. The hierarchical partition is presented using the dendrogram.

The main disadvantages of the hierarchical methods are:

Inability to scale well — The time complexity of hierarchical algorithms is at least $O(m^2)$ (where m is the total number of instances), which is non-linear with the number of objects. Clustering a large number of objects using a hierarchical algorithm is also characterized by huge I/O costs. Hierarchical methods can never undo what was done previously. Namely there is no back-tracking capability.

PartitioningMethods

Partitioning methods relocate instances by moving them from one cluster to another, starting from an initial partitioning. Such methods typically require that the number of clusters will be pre-set by the user. To achieve global optimality in partitioned-based clustering, an exhaustive enumeration process of all possible partitions is required. Because this is not feasible, certain greedy heuristics are used in the form of iterative optimization. Namely, a relocation method iteratively relocates points between the *k* clusters. The following subsections present various types of partitioning methods.

7**.5 Quality and Validity of Cluster Analysis**.

UNIT – 8

Web Mining

Introduction

Web mining - is the application of data mining techniques to discover patterns from the Web. According to analysis targets, web mining can be divided into three different types, which are Web usage mining, Web content mining and Web structure mining.

Web Mining

Web content mining is the mining, extraction and integration of useful data, information and knowledge from Web page content. The heterogeneity and the lack of structure that permeates much of the ever-expanding information sources on the World Wide Web, such as hypertext documents, makes automated discovery, organization, and search and indexing tools of the Internet and the World Wide Web such as Lycos, Alta Vista, WebCrawler, ALIWEB [6], MetaCrawler, and others provide some comfort to users, but they do not generally provide structural information nor categorize, filter, or interpret documents. In recent years these factors have prompted researchers to develop more intelligent tools for information retrieval, such as intelligent web agents, as well as to extend database and data mining techniques to provide a higher level of organization for semi-structured data available on the web. The agent-based approach to web mining involves the development of sophisticated AI systems that can act autonomously or semi-autonomously on behalf of a particular user, to discover and organize web-based information.

Web content mining is differentiated from two different points of view:[1] Information Retrieval View and Database View. R. Kosala et al.[2] summarized the research works done for unstructured data and semi-structured data from information retrieval view. It shows that most of the researches use bag of words, which is based on the statistics about single words in isolation, to represent unstructured text and take single word found in the training corpus as features. For the semi-structured data, all the works utilize the HTML structures inside the documents and some utilized the hyperlink structure between the documents for document representation. As for the database view, in order to have the better information management and querying on the web, the mining always tries to infer the structure of the web site to transform a web site to become a database.

There are several ways to represent documents; vector space model is typically used. The documents constitute the whole vector space. If a term t occurs n(D, t) in document D, the t-th coordinate of D is n(D, t) . When the length of the words in a document goes to **I,** D maxt n(D, t) **I** . This representation does not realize the importance of words in a document. To resolve this, tf-idf (Term Frequency Times Inverse Document Frequency) is introduced.

By multi-scanning the document, we can implement feature selection. Under the condition that the category result is rarely affected, the extraction of feature subset is needed. The general algorithm is to construct an evaluating function to evaluate the features. As feature set, Information Gain, Cross Entropy, Mutual Information, and Odds Ratio are usually used. The classifier and pattern analysis methods of text data mining are very similar to traditional data mining techniques. The usual evaluative merits are Classification Accuracy, Precision, Recall and Information Score.

Text mining

Text mining, also referred to as *text data mining*, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interestingness. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (*i.e.*, learning relations between named entities).Text analysis involves information retrieval, lexical analysis to study word frequency distributions, pattern recognition, tagging/annotation, information extraction, data mining techniques including link and association analysis, visualization, and predictive analytics. The overarching goal is, essentially, to turn text into data for analysis, via application of natural language processing (NLP) and analytical methods.A typical application is to scan a set of documents written in a natural language and either model the document set for predictive classification purposes or populate a database or search index with the information extracted.

Generalization of Structured Data

An important feature of object-relational and object-oriented databases is their capabilityof storing, accessing, and modeling complex structure-valued data, such as set- and list-valued data and data with nested structures.

"How can generalization be performed on such data?" Let's start by looking at thegeneralization of set-valued, list-valued, and sequence-valued attributes.

A set-valued attribute may be of homogeneous or heterogeneous type. Typically, set-valued data can be generalized by (1) *generalization of each value in the set to its corresponding higher-level concept*, or (2) *derivation of the general behavior of the set*, such as the number of elements in the set, the types or value ranges in the set, the

weighted average for numerical data, or the major clusters formed by the set. Moreover, generalization can be performed by *applying different generalization operators to explore alternative generalization paths*. In this case, the result of generalization is a heterogeneous set.

Spatial Data Mining

A spatial database stores a large amount of space-related data, such as maps, preprocessed remote sensing or medical imaging data, and VLSI chip layout data. Spatial databases have many features distinguishing them from relational databases. They carry topological and/or distance information, usually organized by sophisticated, multidimensional spatial indexing structures that are accessed by spatial data access methods and often require spatial reasoning, geometric computation, and spatial knowledge representation techniques.

Spatial data mining refers to the extraction of knowledge, spatial relationships, or other interesting patterns not explicitly stored in spatial databases. Such mining demands an integration of data miningwith spatial database technologies. It can be used for understanding spatial data, discovering spatial relationships and relationships between spatial and nonspatial data, constructing spatial knowledge bases, reorganizing spatial databases, and optimizing spatial queries. It is expected to have wide applications in geographic information systems, geo marketing, remote sensing, image database exploration, medical imaging, navigation, traffic control, environmental studies, and many other areas where spatial data are used. A crucial challenge to spatial data mining is the exploration of *efficient* spatial data mining techniques due to the huge amount of spatial data and the complexity of spatial data types and spatial access methods. *"What about using statistical techniques for spatial data mining?"* Statistical spatial data analysis has been a popular approach to analyzing spatial data and exploring geographic information. The term geo *statistics* is often associated with continuous geographic space, whereas the term *spatial statistics* is often associated with discrete space. In a statistical model that handles non spatial data, one usually assumes statistical independence among different portions of data. However, different from traditional data sets, there is no such independence among spatially distributed data because in reality, spatial objects are often interrelated, or more exactly spatially *co-located*, in the sense that *the closer the two objects are located, the more likely they share similar properties*. For example, nature resource, climate, temperature, and economic situations are likely to be similar in geographically

closely located regions. People even consider this as the first law of geography: "*Everything is related to everything else, but nearby things are more related than distant things.*" Such a property of close interdependency across nearby space leads to the notion of spatial autocorrelation. Based on this notion, spatial statistical modeling methods have been developed with good success. Spatial data mining will further develop spatial statistical analysis methods

and extend them for huge amounts of spatial data, with more emphasis on efficiency, scalability, cooperation with database and data warehouse systems, improved user interaction, and the discovery of new types of knowledge.

Mining spatio-temporal data

Spatio-temporal data mining is an emerging research area dedicated to the development and application of novel computational techniques for the analysis of large spatio-temporal databases. The main impulse to research in this subfield of data mining comes from the large amount of & spatial data made available by GIS, CAD, robotics and computer vision applications, computational biology, and mobile computing applications; & temporal data obtained by registering events (e.g., telecommunication or web traffic data) and monitoring processes and workflows.Both the temporal and spatial dimensions add substantial complexity to data mining tasks.

First of all, the spatial relations, both metric (such as distance) and non-metric (such as topology, direction, shape, etc.) and the temporal relations (such as before and after) are information bearing and therefore need to be considered in the data mining methods.

Secondly, some spatial and temporal relations are implicitly defined, that is, they are not explicitly encoded in a database. These relations must be extracted from the data and there is a trade-off between pre computing them before the actual mining process starts (eager approach) and computing them on-the-fly when they are actually needed (lazy approach). Moreover, despite much formalization of space

and time relations available in spatio-temporal reasoning, the extraction of spatial/ temporal relations implicitly defined in the data introduces some degree of fuzziness that may have a large impact on the results of the data mining process.

Thirdly, working at the level of stored data, that is, geometric representations (points, lines and regions) for spatial data or time stamps for temporal data, is often undesirable. For instance, urban planning researchers are interested in possible relations between two roads, which either cross each other, or run parallel, or can be confluent, independently of the fact that the two roads are represented by one or

more tuples of a relational table of Blines'' or Bregions''. Therefore, complex transformations are required to describe the units of analysis at higher conceptual levels, where human-interpretable properties and relations are expressed. Fourthly, spatial resolution or temporal granularity can have direct impact on the strength of patterns that can be discovered in the datasets. Interesting patterns are more likely to be discovered at the lowest resolution/granularity level. On the other hand, large support is more likely to exist at higher levels. Fifthly, many rules of qualitative reasoning on spatial and temporal data (e.g., transitive properties for temporal relations after and

before), as well as spatiotemporal ontologies, provide a valuable source of domain independent knowledge that should be taken into account when generating patterns. How to express these rules and how to integrate spatio-temporal reasoning mechanisms in data mining systems are still open problems.