Data Warehousing and Data Mining

Study Material for **MS-31**

Directorate of Distance Education
Guru Jambheshwar University of Science &Technology, Hisar

Study Material Prepared by

Varun Kumar

Published by Excel Books**,** A-45, Naraina, Phase-I, New Delhi-110 028

Published by Anurag Jain for Excel Books, A-45, Naraina, Phase-I, New Delhi-110 028 and printed by him at Excel Printers, C-206, Naraina, Phase-I, New Delhi-110 028.

CONTENTS

Data Warehousing

UNIT

1

DATA WAREHOUSING INTRODUCTION

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Define Data Warehouse.
- Describe a Common way of Introducting Data Warehousing.
- Illustrate OLTP and Data Warehousing Environment
- Develop a Data Warehouse Architecture.
- Understand about Meta Data.
- Know about Technical Architecture.

U N I T S T R U C T U R E

1.1 INTRODUCTION

A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. It contains historical data derived from transaction data, but it can include data from other sources.

A system for monitoring data warehouse use is valuable for capturing queries and tracking usage, and performance tuning is also helpful. Data marts are systems designed for a particular line of business. Complete data warehouse architecture includes data and technical elements.

1.2 WHAT IS A DATA WAREHOUSE?

A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but it can include data from other sources. It separates analysis workload from transaction workload and enables an organization to consolidate data from several sources.

In addition to a relational database, a data warehouse environment includes an extraction, transportation, transformation, and loading (ETL) solution, an online analytical processing (OLAP) engine, client analysis tools, and other applications that manage the process of gathering data and delivering it to business users.

A common way of introducing data warehousing is to refer to the characteristics of a data warehouse as set forth by William Inmon:

- Subject Oriented
- Integrated
- Nonvolatile
- Time Variant

Subject Oriented

Data warehouses are designed to help you analyze data. For example, to learn more about your company's sales data, you can build a warehouse that concentrates on sales. Using this warehouse, you can answer questions like "Who was our best customer for this item last year?" This ability to define a data warehouse by subject matter, sales in this case, makes the data warehouse subject oriented.

Integrated

Integration is closely related to subject orientation. Data warehouses must put data from disparate sources into a consistent format. They must resolve such problems as naming conflicts and inconsistencies among units of measure. When they achieve this, they are said to be integrated.

Nonvolatile

Nonvolatile means that, once entered into the warehouse, data should not change. This is logical because the purpose of a warehouse is to enable you to analyze what has occurred.

Time Variant

In order to discover trends in business, analysts need large amounts of data. This is very much in contrast to online transaction processing (OLTP) systems, where performance requirements demand that historical data be moved to an archive. A data warehouse's focus on change over time is what is meant by the term time variant.

Contrasting OLTP and Data Warehousing Environments

Figure 1.1 illustrates key differences between an OLTP system and a data warehouse.

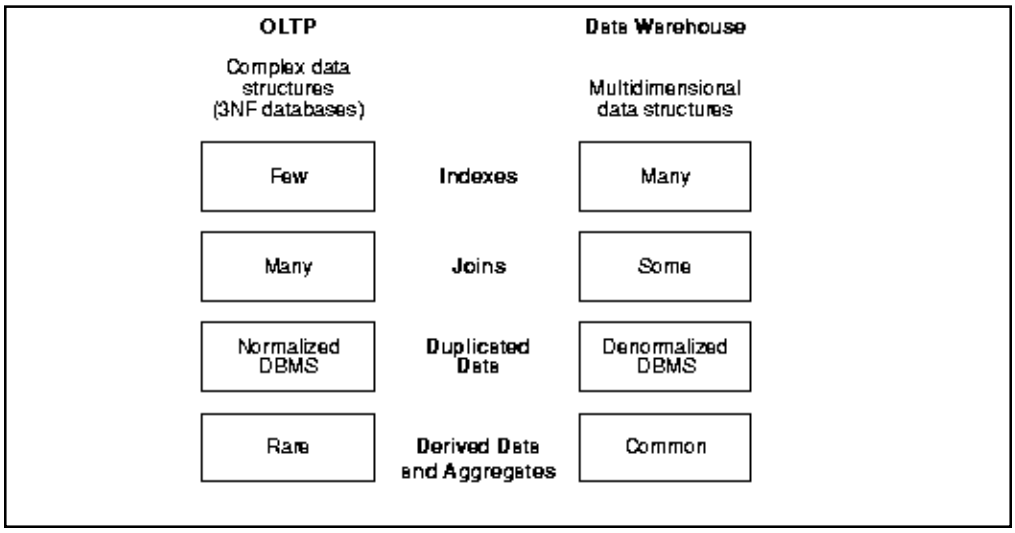


Figure 1.1 Constrasting OLTP and Data Warehousing Environments

One major difference between the types of system is that data warehouses are not usually in third normal form (3NF), a type of data normalization common in OLTP environments.

Data warehouses and OLTP systems have very different requirements. Here are some examples of differences between typical data warehouses and OLTP systems:

- ***Workload:*** Data warehouses are designed to accommodate *ad hoc* queries. You might not

know the workload of your data warehouse in advance, so a data warehouse should be optimized to perform well for a wide variety of possible query operations.

OLTP systems support only predefined operations. Your applications might be specifically tuned or designed to support only these operations.

- ***Data modifications:*** A data warehouse is updated on a regular basis by the ETL process (run nightly or weekly) using bulk data modification techniques. The end users of a data warehouse do not directly update the data warehouse.

  In OLTP systems, end users routinely issue individual data modification statements to the database. The OLTP database is always up to date, and reflects the current state of each business transaction.

- ***Schema design:*** Data warehouses often use denormalized or partially denormalized schemas (such as a star schema) to optimize query performance.

  OLTP systems often use fully normalized schemas to optimize update/insert/delete performance, and to guarantee data consistency.

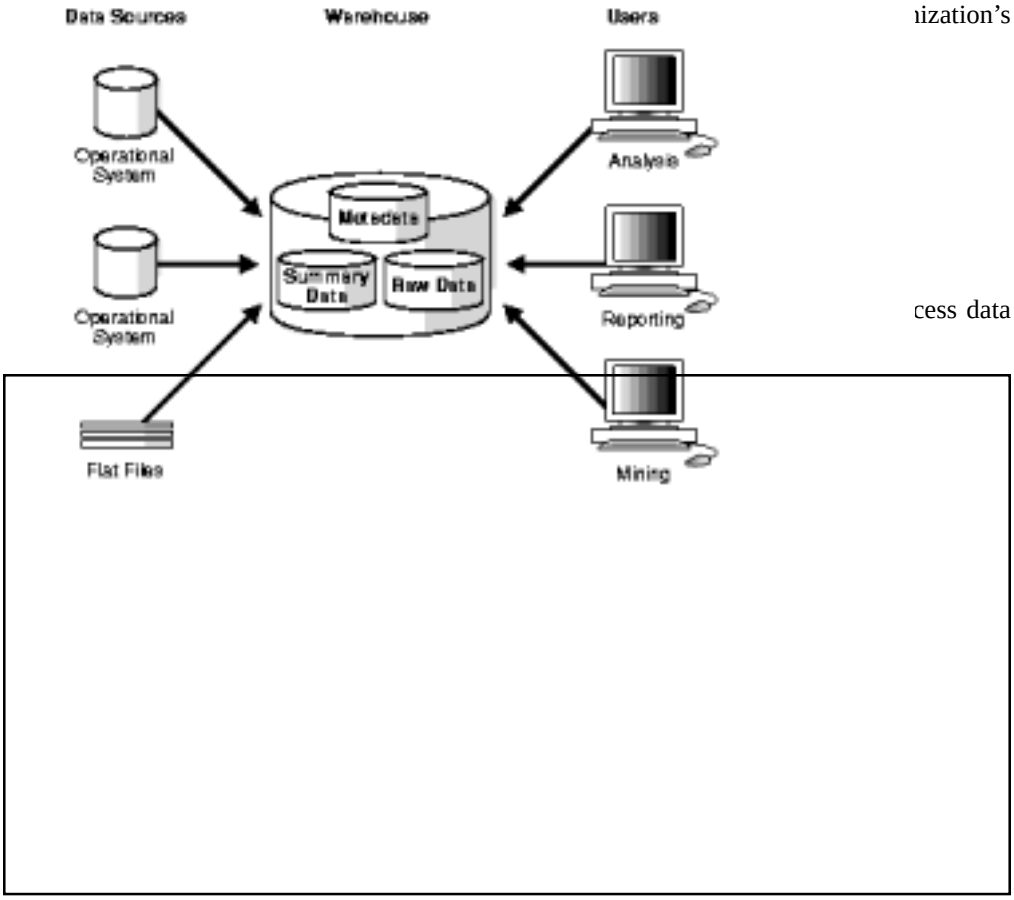
- ***Typical operations:*** A typical data warehouse query scans thousands or millions of rows. For example, "Find the total sales for all customers last month."

  A typical OLTP operation accesses only a handful of records. For example, "Retrieve the current order for this customer."

- ***Historical data:*** Data warehouses usually store many months or years of data. This is to support historical analysis.

  OLTP systems usually store data from only a few weeks or months. The OLTP system stores only historical data as needed to successfully meet the requirements of the current transaction.

1.3 DATA WAREHOUSE ARCHITECTURES



ization's

cess data

Figure 1.2 Architecture of a Data Warehouse

In Figure 1.2, the metadata and raw data of a traditional OLTP system is present, as is an additional type of data, summary data. Summaries are very valuable in data warehouses because they pre-compute long operations in advance. For example, a typical data warehouse query is to retrieve something like August sales. A summary in Oracle is called a materialized view.

Data Warehouse Architecture (with a Staging Area)

In Figure 1.2, you need to clean and process your operational data before putting it into the warehouse. You can do this programmatically, although most data warehouses use a staging area instead. A staging area simplifies building summaries and general warehouse management. Figure 1.3 illustrates this typical architecture.

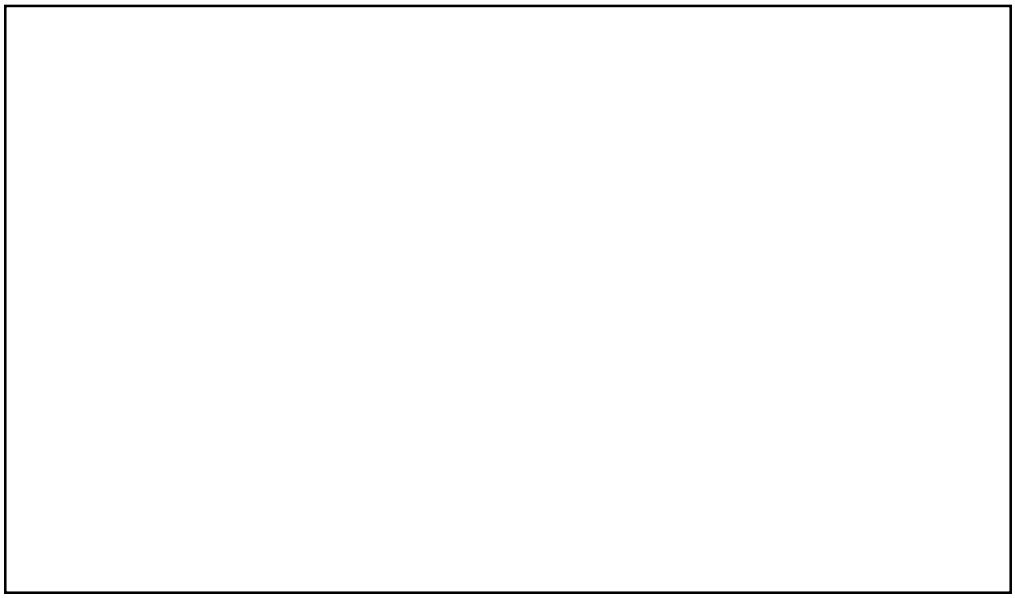


Figure 1.3 Architecture of a Data Warehouse with a staging Area

Data Warehouse Architecture (with a Staging Area and Data Marts)

Although the architecture in Figure 1.3 is quite common, you may want to customize your warehouse's architecture for different groups within your organization. You can do this by adding **data marts**, which are systems designed for a particular line of business. Figure 1.4 illustrates an example where purchasing, sales, and inventories are separated. In this example, a financial analyst might want to analyze historical data for purchases and sales.

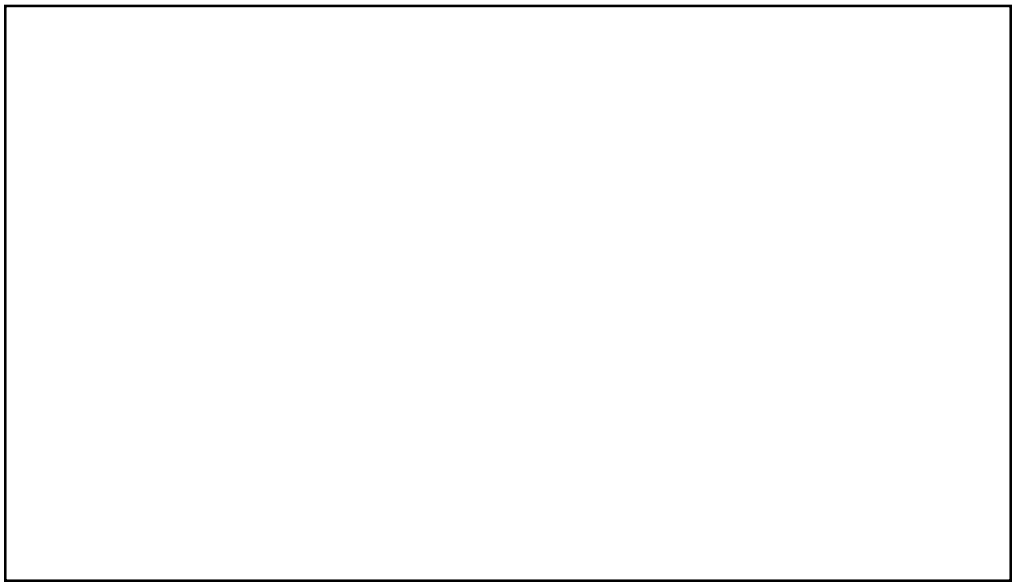


Figure 1.4 Architecture of a Data Warehouse with a Staging Area and Data Marts

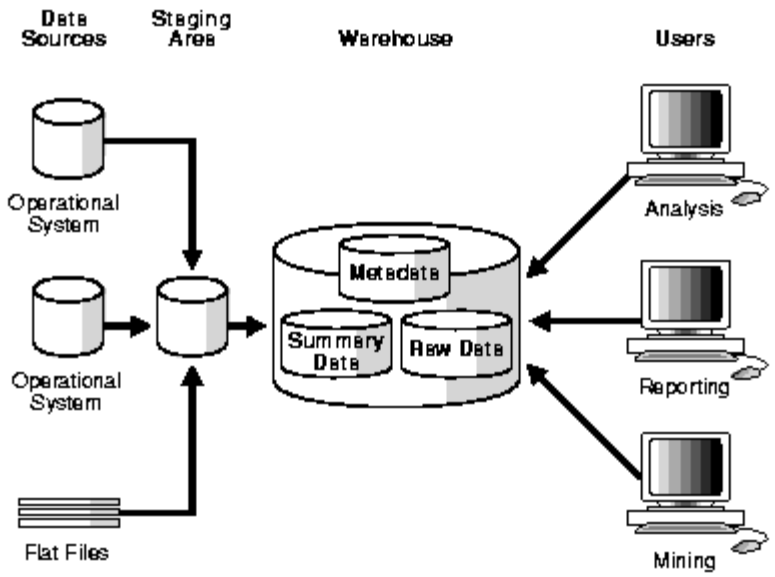
Criteria for a Data Warehouse DBMS Selection

Based on the realities of data warehousing today, a selected technical architecture for data warehousing should be:

- ***Manageable:*** Through minimal support tasks requiring DBA/System Administrator intervention. It should provide a single point of control to simplify system administration. You should be able to create and implement new tables and indexes at will.
- ***Complete and Integrated:*** The toolset should be comprehensive across the spectrum of eventual requirements for data and its access.
- ***Interoperable:*** Integrated access to the web, Microsoft Office, internal networks, and corporate mainframes.
- ***Scalable:*** The ability to handle increasing data sizes and workloads with simple additions to the architecture, as opposed to the increases requiring a rearchitecture
- ***Affordable:*** Proposed solution (hardware, software, services, required customer support) providing a low total cost of ownership (TCO) over a multi-year period.
- ***Proven and Supported:*** You don't want to risk a critical decision regarding a fundamental underpinning of the data warehouse environment on an unproven solution.
- ***Flexible:*** Provides optimal performance across the full range of models with large numbers of tables. Look for proven ability to support multiple applications from different business units, leveraging data that is integrated across business functions and subject areas.
- ***User Accessible:*** Compatibilities and interoperability with data access tools that provide a wide range of user-friendly access options.

Student Activity 1

1. Define the term "Data Warehouse". List the various characteristics of data



models, drawings, and specifications, with separate sections for each key component area and enough detail to allow their implementation by skilled professionals.

It's also not easy, he says, because we've only been developing data warehouse systems for 15 years, versus 5,000 years for building homes. Therefore we have fewer standards, the tools and techniques are rapidly evolving, there is little documentation of what systems we already have, and data warehouse terminology is extremely loose.

So while developing an architecture is difficult, it is possible—and it's critical. First and foremost, he says, the architecture has to be driven by the business. If your requirement is to have nightly updates, this has implications for the architecture, and you must understand the technical requirements to achieve what you want to have. A few business requirement examples, and the

general technical considerations for each:

- ***Nightly updates:*** Adequate staging horsepower.
- ***Worldwide availability:*** Parallel or distributed servers.
- ***Customer-level analysis:*** [large] server size.
- ***New data sources:*** Flexible tools with support for meta data.
- ***Reliability*** - Job control features.

Key Component Areas

A complete data warehouse architecture includes data and technical elements. The three broad areas of data warehouse architecture are, first, data architecture, is centered on business processes. The next area, infrastructure, includes hardware, networking, operating systems, and desktop machines. Finally, the technical area encompasses the decision-making technologies that will be needed by the users, as well as their supporting structures. These areas are detailed in the sub-sections below.

Data Architecture

As stated above, the data architecture portion of the overall data warehouse architecture is driven by business processes. For example, in a manufacturing environment the data model might include orders, shipping, and billing. Each area draws on a different set of dimensions. But where dimensions intersect in the data model the definitions have to be the same—the same customer who buys is the same that builds. So data items should have a common structure and content, and involve a single process to create and maintain.

As you work through the architecture and present data to your users, tool choices will be made, but many choices will disappear as the requirements are set. For example, he explains that product capabilities are beginning to merge, like MOLAP and ROLAP. MOLAP is okay if you stay within the cube you've built. It's fast and allows for flexible querying—within the confines of the cube. Its weaknesses are size (overall and within a dimension), design constraints (limited by the cube structure), and the need for a proprietary data base.

Infrastructure Architecture

With the required hardware platform and boxes, sometimes the data warehouse becomes its own IS shop. Indeed, there are lots of "boxes" in data warehousing, mostly used for data bases and application servers.

The issues with hardware and DBMS choices are size, scalability, and flexibility. In about 80 percent of data warehousing projects this isn't difficult; most businesses can get enough power to handle their needs.

In terms of the network, check the data sources, the warehouse staging area, and everything in between to ensure there's enough bandwidth to move data around. On the desktop, run the tools and actually get some data through them to determine if there's enough power for retrieval. Sometimes the problem is simply with the machine, and the desktops must be powerful enough to run current-generation access tools. Also, don't forget to implement a software distribution mechanism.

Technical Architecture

An important component of technical architecture is the data staging process, which covers five major areas:

- ***Extract:*** Data comes from multiple sources and is of multiple types. Data compression and encryption handling must be considered at this area, if it applies.
- ***Transform:*** Data transformation includes surrogate key management, integration, de-normalization, cleansing, conversion, aggregation, and auditing.

- ***Load:*** Loading is often done to multiple targets, with load optimization and support for the entire load cycle.
- ***Security:*** Administrator access and data encryption policies.
- ***Job control:*** This includes job definition, job scheduling (time and event), monitoring, logging, exception handling, error handling, and notification.

The staging box needs to be able to extract data from multiple sources, like MVS, Oracle, VM, and others, so be specific when you choose your products. It must handle data compression and encryption, transformation, loading (possibly to multiple targets), and security . In addition, the staging activities need to be automated. Many vendors' offerings do different things, so he advises that most organizations will need to use multiple products.

A system for monitoring data warehouse use is valuable for capturing queries and tracking usage, and performance tuning is also helpful. Performance optimization includes cost estimation through a "governor" tool, and should include ad hoc query scheduling. Middleware can provide query management services. Tools for all of these and other related tasks are available for the front end, for server-based query management, and for data from multiple sources. Tools are also available for reporting, connectivity, and infrastructure management. Finally, the data access piece should include reporting services (such as publish and subscribe), a report library, a scheduler, and a distribution manager.

A Word About Meta Data

The creation and management of data has the following "steps" in the data warehouse process:

1. warehouse model
2. source definitions
3. table definitions
4. source-to-target maps
5. map and transformation information
6. physical information (table spaces, etc.)
7. extracted data
8. transformed data
9. load statistics
10. business descriptions
11. query requests
12. the data itself
13. query statistics

To show how important meta data is, of the steps listed above only three involve "real" data—7, 8, and 12. Everything else is meta data and the whole data warehouse process relies on it. The major technical elements of a meta data catalog include:

- **Business rules:** Includes definitions, derivations, related items, validation, and hierarchy information (versions, dates, etc.).
- ***Movement/transformation information:*** Source/destination information, as well as DDL (data types, names, etc.).
- ***Operations information:*** Data load job schedules, dependencies, notification, and reliability information (such as host redirects and load balancing).
- ***Tool-specific information:*** Graphic display information and special function support.
- ***Security rules:*** Authentication and authorization.

Developing an Architecture

When you develop the technical architecture model, draft the architecture requirements document first. Next to each business requirement write down its architecture implications. Group these implications according to architecture areas (remote access, staging, data access tools, etc.) Understand how it fits in with the other areas. Capture the definition of the area and its contents. Then refine and document the model.

Developing a data warehouse architecture is difficult, and thus warns against using a "just do it" approach, which he also calls "architecture lite." But the Zachman framework is more than what most organizations need for data warehousing, so he recommends a reasonable compromise consisting of a four-layer process: business requirements, technical architecture, standards, and products.

Business requirements essentially drive the architecture, so talk to business managers, analysts, and power users. From your interviews look for major business issues, as well as indicators of business strategy, direction, frustrations, business processes, timing, availability, and performance expectations. Document everything well.

From an IT perspective, talk to existing data warehouse/DSS support staff, OLTP application groups, and DBAs; as well as networking, OS, and desktop support staff. Also speak with architecture and planning professionals. Here you want to get their opinions on data warehousing considerations from the IT viewpoint. Learn if there are existing architecture documents, IT principles, standards statements, organizational power centers, etc.

Not many standards exist for data warehousing, but there are standards for a lot of the components. The following are some to keep in mind:

Middleware: ODBC, OLE, OLE DB, DCE, ORBs, and JDBC.

Data base connectivity: ODBC, JDBC, OLE DB, and others.

Data management: ANSI SQL and FTP.

Network access: DCE, DNS, and LDAP.

Regardless of what standards they support, major data warehousing tools are meta data-driven. However, they don't often share meta data with each other and vary in terms of openness.

How detailed does a data warehouse architecture need to be? The question to ask is this: Is this enough information to allow a competent team to build a warehouse that meets the needs of the business? As for how long it will take, the architecture effort will grow exponentially as more people are added for its development (i.e., it becomes "techno-politically complex"), and more complex the resulting system needs to be (i.e., "functionally complex").

Benefits of having a data warehouse architecture

- ***Provides an organizing framework:*** The architecture draws the lines on the map in terms of what the individual components are, how they fit together, who owns what parts, and priorities.
- ***Improved flexibility and maintenance:*** Allows you to quickly add new data sources, interface standards allow plug and play, and the model and meta data allow impact analysis and single-point changes.
- ***Faster development and reuse:*** Warehouse developers are better able to understand the data warehouse process, data base contents, and business rules more quickly.
- ***Management and communications tool:*** Define and communicate direction and scope to set expectations, identify roles and responsibilities, and communicate requirements to vendors.
- ***Coordinate parallel efforts:*** Multiple, relatively independent efforts have a chance to converge successfully. Also, data marts without architecture become the stovepipes of tomorrow.

Student Activity 2

1. Write down the main benefits of having a data warehouse architecture.

1.5 SUMMARY

1. A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing.
2. A data warehouse contains historical data derived from transaction data, but it can include data from other sources.
3. Data warehouses must put data from disparate sources into a consistent format.
4. One major difference between the types of system is that data warehouses are not usually in third normal form (3NF), a type of data normalization common in OLTP environments.
5. Data warehouses often use denormalized or partially denormalized schemas (such as a star schema) to optimize query performance.
6. Data warehouses and their architectures vary depending upon the specifics of an organization's situation. Three common architectures are:
   - Data Warehouse Architecture (Basic)
   - Data Warehouse Architecture (with a Staging Area)
   - Data Warehouse Architecture (with a Staging Area and Data Marts)
7. A system for monitoring data warehouse use is valuable for capturing queries and tracking usage, and performance tuning is also helpful.
8. Data marts are systems designed for a particular line of business.
9. Complete data warehouse architecture includes data and technical elements.
10. An important component of technical architecture is the data staging process, which covers five major areas: *Extract, Transform, Load, Security and Job control*

1.6 KEYWORDS

A data warehouse: It is a relational database that is designed for query and analysis rather than for transaction processing.

A data warehouse architecture: It is a description of the elements and services of the warehouse, with details showing how the components will fit together and how the system will grow over time.

Job control: This includes job definition, job scheduling (time and event), monitoring, logging, exception handling, error handling, and notification.

1.7 REVIEW QUESTIONS

1. How a date warehouse is different from data base? Explain with the help of suitable example.
2. Highlights the main differences between OLTP and Data warehouse.
3. Explain the architecture of Data Warehouse in details.
4. Describe the criteria for a Data Warehouse DBMS Selection.
5. What could be the strategy to design a data warehouse architecture. Discuss.

1.8 FURTHER READINGS

Usama Fayyad, "Mining Databases: Towards Algorithms for Knowledge Discovery", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998. Christopher Matheus, Philip Chan, and Gregory Piatetsky-Shapiro, "Systems for Knowledge Discovery in Databases", IEEE Transactions on Knowledge and Data Engineering, 5(6):903-913, December 1993.

Rakesh Agrawal and Tomasz Imielinski, "Database Mining: A Performance Perspective", IEEE Transactions on Knowledge and Data Engineering, 5(6):914-925, December 1993.

Usama Fayyad, David Haussler, and Paul Stolorz, "Mining Scientific Data", Communications of the ACM, vol. 39, no. 11, pp. 51-57, November 1996.

David J. Hand, "Data Mining: Statistics and more?", The American Statistician, vol. 52, no. 2, pp 112-118, May 1998.

Tom M. Mitchell, "Does machine learning really work?", AI Magazine, vol. 18, no. 3, pp. 11-20, Fall 1997.

Clark Glymour, David Madigan, Daryl Pregibon, and Padhraic Smyth, "Statistical Inference and Data Mining", Communications of the ACM, vol. 39, no. 11, pp. 35-41, November 1996.

Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford, "Scalable, Distributed Data Mining using An Agent Based Architecture", Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), August 1997, Newport Beach, California, USA.

M-S. Chen, Jiawei Han, and Philip S. Yu, "Data Mining: An Overview from a Database Perspective", IEEE Transactions on Knowledge and Data Engineering, 8(6): 866-883, 1996. Surajit Chaudhuri, "Data Mining and Database Systems: Where is the Intersection?", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998.

UNIT

2

TYPES OF END-USER

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Know about end user.
- Develop a Data Warehouse Strategy.
- Design Data Warehouse.
- Manage Data Warehouse.
- Understand about IT Management.
- Future Scop of Data Warehousing

U N I T S T R U C T U R E

2.1 INTRODUCTION

In the same sense that there are lots of different ways to organize a data warehouse, it is important to note that there are an increasingly wide range of end-users as well. In general we tend to think in terms of three broad categories of end-users:

- Executives and managers
- "Power" users (business and financial analysts, engineers, etc.)
- Support users (clerical, administrative, etc.)

Each of these different categories of users has its own set of requirements for data, access, flexibility and ease of use.

2.2 DEVELOPING DATA WAREHOUSES

Developing a good data warehouse is no different from any other IT project; it requires careful planning, requirements definition, design, prototyping and implementation. The first and most important element is a planning process that determines what kind of data warehouse strategy the organization is going to start with.

Developing a Data Warehouse Strategy

Before developing a data warehouse, it is critical to develop a balanced data warehousing strategy that is appropriate for its needs and its user population. Who is the audience? What is the scope? What type of data warehouse should we build?

There are a number of strategies by which organizations can get into data warehousing. One way is to establish a "Virtual Data Warehouse" environment. A Virtual Data Warehouse is created by: (1) installing a set of data access, data directory and process management facilities, (2) training the end-users (3) monitoring how the data warehouse facilities are actually used and then (4) based on actual usage, create a physical data warehouse to support the high-frequency requests.

A second strategy is simply to build a copy of the operational data from a single operational system and enable the data warehouse from a series of information access tools. This strategy has the advantage of being both simple and fast. Unfortunately, if the existing data is of poor quality and/or the access to the data has not been thought out, then this approach can create a number of significant problems.

Ultimately, the optimal data warehousing strategy is to select a user population based on value to the enterprise and do an analysis of their issues, questions and data access needs. Based on these needs, prototype data warehouses are built and populated so the end-users can experiment and modify their requirements. Once there is general agreement on the needs, then the data can be acquired from existing operational systems across the enterprise and/or from external data sources and loaded into the data warehouse. If it is required, information access tools can also be enabled to allow end-users to have access to required data using their own favorite tools or to allow for the creation of high-performance multi-dimensional information access systems using the core data warehouse as the basis.

In the final analysis, there is no one approach to building a data warehouse that will fit the needs of every enterprise. Each enterprise's needs are different as is each enterprise's context. In addition, since data warehouse technology is evolving as we learn more about developing data warehouses, it turns out that the only practical approach to data warehousing is an evolutionary one.

2.3 EVOLVING A DATA WAREHOUSE ARCHITECTURE

The Data Warehouse Architecture in given figure on the next page, is simply a framework for understanding data warehousing and how the components of data warehousing fit together. Only the most sophisticated organizations will be able to put together such an architecture the first time out. What the Data Warehouse Architecture provides then is a kind of roadmap that can be used in designing coupled with an understanding of the options at hand, the Data Warehouse Architecture provides a useful way of determining if the organization is moving towards a reasonable data warehousing framework.

One of the keys to data warehousing is flexibility. It is critical to keep in mind that the more successful a data warehouse strategy is, the more end-users are going to want to add to it.

2.4 DESIGNING DATA WAREHOUSES

Designing data warehouses is very different from designing traditional operational systems. For one thing, data warehouse users typically don't know nearly as much about their wants and needs as operational users. Second, designing a data warehouse often involves thinking in terms of much broader, and more difficult to define, business concepts than does designing an operational system. In this respect, data warehousing is quite close to Business Process Reengineering (BPR). Finally, the ideal design strategy for a data warehouse is often outside-in as opposed to top-down.

But while data warehouse design is different from what we have been used to, it is no less important. The fact that end-users have difficulty defining what they need as a bare minimum is no less necessary. In practice, data warehouse designers find that they have to use every trick in the book to help their users "visualize" their requirements. In this respect, robust working prototypes are essential.

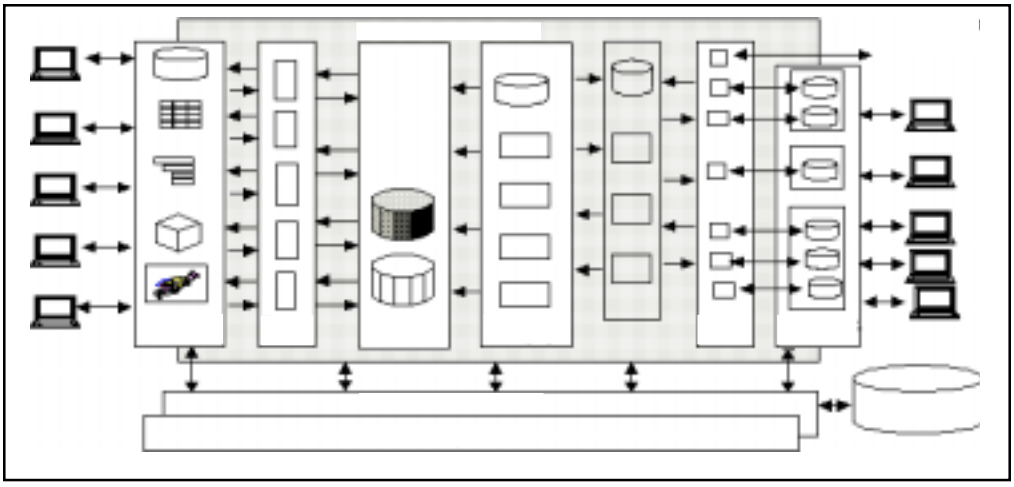


Figure 2.1

2.5 MANAGING DATA WAREHOUSES

Data Warehouses are not magic-they require a great deal of very hard work. In many cases data warehouse projects are viewed as a stopgap measure to get users off our backs or to provide something for nothing. But data warehouses require careful management and marketing. A data warehouse is a good investment only if end-users actually can get vital information faster and cheaper than they can using current technology. As a consequence, management has to think seriously about how they want their warehouses to perform and how they are going to get the word out to the end-user community. And management has to recognize that the maintenance of the data warehouse structure is as critical as the maintenance of any other mission-critical application. In fact, experience has shown that data warehouses quickly become one of the most used systems in any organization.

Management, especially IT management, must also understand that if they embark on a data warehousing program, they are going to create new demands upon their operational systems: demands for better data, demands for consistent data, demands for different kinds of data.

2.6 FUTURE DEVELOPMENTS

Data Warehousing is such a new field that it is difficult to estimate what new developments are likely to most affect it. Clearly, the development of parallel DB servers with improved query engines is likely to be one of the most important. Parallel servers will make it possible to access huge data bases in much less time.

Another new technology is data warehouses that allow for the mixing of traditional numbers, text and multimedia. The availability of improved tools for data visualization (business intelligence) will allow users to see things that could never be seen before.

2.7 SUMMARY

1. The three categories of user are executives and managers, power users and support users.
2. Developing a good data warehouse requires careful planning, requirement definitions, design, prototyping and implementation.
3. Organizations can get into data warehousing by a virtual data warehouse environment.
4. A virtual data warehouse is created by installing a data access, data directory and process management facilities, training the end users, monitoring how the data warehouse facilities are used and on the actual usage.
5. Data warehousing is close to business process reengineering as it is difficult to define the business concepts than designing an operational system.

6. A data ware house requires careful management and marketing.

7. A data warehouse is a good investment only if end users actually get vital information faster and cheaper than using current technology.

8. Data warehouses in this initial stage are developed by simply copying the database of an operational system to an off-line server where the processing load of reporting does not impact on the operational system's performance.

9. OLAP is an acronym for On Line Analytical Processing. It is an approach to quickly provide the answer to analytical queries that are dimensional in nature.

10. Integration of data within a warehouse is accomplished by making the data consistent in format, naming, and other aspects.

2.8 KEYWORDS

OLAP (On Line Analytical Processing): An acronym for On Line Analytical Processing.

Data Warehouse Architecture: A framework for understanding data warehousing and how the components of data warehousing fit together.

2.9 REVIEW QUESTIONS

1. Explain the various categories of users in context with data warehousing.

2. Write down the steps to create a virtual data warehouse environment.

3. List down the various method to store data in data warehouse.

4. Why should the data warehousing be flexible?

5. Why designing of data warehousing is different from designing traditional operational systems?

6. Define & Discuss the framework to manage data warehouses.

2.10 FURTHER READINGS

Adriaans, P. and Zantige, D. *Data Mining*, Harlow, UK: Addison-Wesley, 1996

Berry, M.J.A. and Linoff, G., *Data Mining Techniques for Marketing, Sales, and Customer Support*, New York, NY: John Wiley and Sons, 1997

Bishop, C. *Neural Networks for Pattern Recognition*, Oxford, UK: Clarendon Press, 1995

Breiman, L., Fredman, J., Olshen, R.A., and Stone, C.J., *Classification and Regression Trees*, Monterey, CA: Wadsworth & Brooks, 1984

Chester, M., *Neural Networks: A Tutorial*, Englwood cliffs, NJ, Prentice Hall, 1993

Cressie, N., *Statistics for Spatial Data*, Revised Edition, Wiley, New York, 1993

Fayyad, U.M.; Piatestsky-Shapiro G.; Smyth D.; and Uthurusamy R., *Advances in Knowledge Discovery and Data Mining*, Cambridge, MA: AAAI Press/MIT Press, 1996

Fukunaga, K., *Introduction to Statistical Pattern Recognition*, Academic Press, 1990

Groth, R., *Data Mining, A Hands-On Approach for business Professionals*, Prentice Hall, 1998 [with demo software]

Hand, D.J., Mannila, H., Smyth, P., *Principles of Data Mining*, MIT Press, [late] 2000

Jain, A. and Dubes, R., *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988

Jordan, M.I., *Learning in Graphical Models*, MIT Press, 1999

Kargupta, Hillol, and Chan, Philip, *Advances in Distributed and Parallel Knowledge Discovery*, MIT/AAAI Press, 2000

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*," New York: Springer-Verlag, 1994

Mitchell, T.M., *Machine Learning*, New York, NY: McGrawl Hill, 1997

Ripley, B.D., *Pattern Recognition and Neural Networks*, Cambridge, UK: Cambridge University Press, 1996

Shekhar, S. and Chawla, S., *Spatial Databases: Issues, Implementations, and Trends*, Prentice Hall, 2000

Tucker, A.B., Ed, *The Computer Science and Engineering Handbook*, CRC Press, 1997

Tukey, J., *Exploratory Data Analysis*, Reading, MA, Addison-Wesley, 1977

Weiss, S.M. and Indurkhya, N., *Predictive Data Mining: A Practical Guide*, San Francisco, CA: Morgan Kaufmann Publishers, 1998

Welstead, S.T., *Neural Network and Fuzzy Logic Applications in C/C++*, New York, John Wiley & Sons, 1994

Witten, I.H. and Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques with JAVA Implementations*, San Francisco, CA: Morgan Kaufmann, 1999

UNIT

3

DATA WAREHOUSING TECHNOLOGY

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Define Operations System.
- Illustrate the Framwork of Data Warehouse.
- Know about various Data base Data Warehouse Options.
- Explain Various Type of Warehoused.
- Evolve a Data Warehouse Architecture.
- Know about Various Data Warehouse Layers.

U N I T S T R U C T U R E

3.1 INTRODUCTION

Information technology itself has made possible revolutions in the way that organizations today operate throughout the world. But the sad truth is that in many organizations despite the availability of more and more powerful computers on everyone's desks and communication networks that span the globe, large numbers of executives and decision makers can't get their hands on critical information that already exists in the organization.

The Data Access Layer of the Data Warehouse Architecture is involved with allowing the Information Access Layer to talk to the Operational Layer.

3.2 "DATA IN JAIL" - THE DATA ACCESS CRISIS

If there is a single key to survival in the 1990s and beyond, it is being able to analyze, plan and react to changing business conditions in a much more rapid fashion. To do this, top managers, analysts and knowledge workers in our enterprises need more and better information.

Every day organizations large and small create billions of bytes of data about all aspects of their business, millions of individual facts about their customers, products, operations and people.

But for the most part, this data is locked up in a myriad of computer systems and is exceedingly difficult to get at. This phenomenon has been described as “data in jail”.

Experts have estimated that only a small fraction of the data that is captured, processed and stored in the enterprise is actually available to executives and decision makers. While technologies for the manipulation and presentation of data have literally exploded, it is only recently that those involved in developing IT strategies for large enterprises have concluded that large segments of the enterprise are “data poor.”

Data Warehousing - Providing Data Access to the Enterprise

Recently, a set of significant new concepts and tools have evolved into a new technology that makes it possible to attack the problem of providing all the key people in the enterprise with access to whatever level of information needed for the enterprise to survive and prosper in an increasingly competitive world.

The term that has come to characterize this new technology is “data warehousing.” Data Warehousing has grown out of the repeated attempts on the part of various researchers and organizations to provide their organizations flexible, effective and efficient means of getting at the sets of data that have come to represent one of the organization’s most critical and valuable assets.

Data Warehousing is a field that has grown out of the integration of a number of different technologies and experiences over the last two decades. These experiences have allowed the IT industry to identify the key problems that have to be solved.

Operational vs. Informational Systems

Perhaps the most important concept that has come out of the Data Warehouse movement is the recognition that there are two fundamentally different types of information systems in all organizations: operational systems and informational systems.

“Operational systems” are just what their name implies; they are the systems that help us run the enterprise operation day-to-day. These are the backbone systems of any enterprise, our “order entry’, “inventory”, “manufacturing”, “payroll” and “accounting” systems. Because of their importance to the organization, operational systems were almost always the first parts of the enterprise to be computerized. Over the years, these operational systems have been extended and rewritten, enhanced and maintained to the point that they are completely integrated into the organization. Indeed, most large organizations around the world today couldn’t operate without their operational systems and the data that these systems maintain.

On the other hand, there are other functions that go on within the enterprise that have to do with planning, forecasting and managing the organization. These functions are also critical to the survival of the organization, especially in our current fast-paced world. Functions like “marketing planning”, “engineering planning” and “financial analysis” also require information systems to support them. But these functions are different from operational ones, and the types of systems and information required are also different. The knowledge-based functions are informational systems.

“Informational systems” have to do with analyzing data and making decisions, often major decisions, about how the enterprise will operate, now and in the future. And not only do informational systems have a different focus from operational ones, they often have a different scope. Where operational data needs are normally focused upon a single area, informational data needs often span a number of different areas and need large amounts of related operational data.

In the last few years, Data Warehousing has grown rapidly from a set of related ideas into an architecture for data delivery for enterprise end-user computing.

3.3 UNDERSTANDING THE FRAMEWORK OF THE DATA WAREHOUSE

One of the reasons that data warehousing has taken such a long time to develop is that it is actually a very comprehensive technology. In fact, data warehousing can be best represented as an enterprise-wide framework for managing informational data within the organization. In order to understand how all the components involved in a data warehousing strategy are related, it is essential to have a Data Warehouse Architecture.

A Data Warehouse Architecture

A Data Warehouse Architecture (DWA) is a way of representing the overall structure of data, communication, processing and presentation that exists for end-user computing within the enterprise. The architecture is made up of a number of interconnected parts:

- Operational Database / External Database Layer
- Information Access Layer
- Data Access Layer
- Data Directory (Metadata) Layer
- Process Management Layer
- Application Messaging Layer
- Data Warehouse Layer
- Data Staging Layer

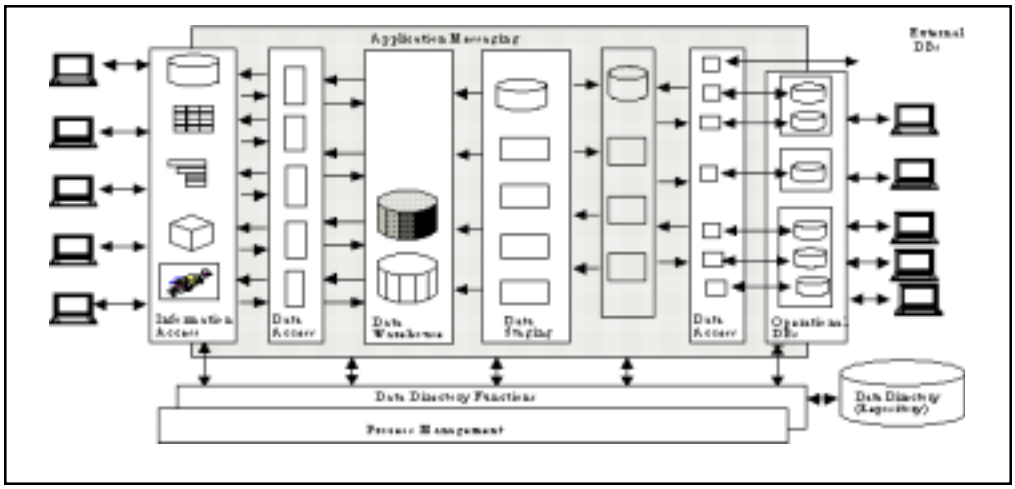


Figure3.1 - Data Warehouse Architecture

Operational Database / External Database Layer

Operational systems process data to support critical operational needs. In order to do that, operational databases have been historically created to provide an efficient processing structure for a relatively small number of well-defined business transactions. However, because of the limited focus of operational systems, the databases designed to support operational systems have difficulty accessing the data for other management or informational purposes. This difficulty in accessing operational data is amplified by the fact that many operational systems are often 10 to 15 years old. The age of some of these systems means that the data access technology available to obtain operational data is itself dated.

Clearly, the goal of data warehousing is to free the information that is locked up in the operational databases and to mix it with information from other, often external, sources of data. Increasingly,

large organizations are acquiring additional data from outside databases. This information includes demographic, econometric, competitive and purchasing trends. The so-called "information superhighway" is providing access to more data resources every day.

Information Access Layer

The Information Access layer of the Data Warehouse Architecture is the layer that the end-user deals with directly. In particular, it represents the tools that the end-user normally uses day to day, e.g., Excel, Lotus 1-2-3, Focus, Access, SAS, etc. This layer also includes the hardware and software involved in displaying and printing reports, spreadsheets, graphs and charts for analysis and presentation. Over the past two decades, the Information Access layer has expanded enormously, especially as end-users have moved to PCs and PC/LANs.

Today, more and more sophisticated tools exist on the desktop for manipulating, analyzing and presenting data; however, there are significant problems in making the raw data contained in operational systems available easily and seamlessly to end-user tools. One of the keys to this is to find a common data language that can be used throughout the enterprise.

Data Access Layer

The Data Access Layer of the Data Warehouse Architecture is involved with allowing the Information Access Layer to talk to the Operational Layer. In the network world today, the common data language that has emerged is SQL. Originally, SQL was developed by IBM as a query language, but over the last twenty years has become the de facto standard for data interchange.

One of the key breakthroughs of the last few years has been the development of a series of data access "filters" such as EDA/SQL that make it possible for SQL to access nearly all DBMSs and data file systems, relational or nonrelational. These filters make it possible for state-of-the-art Information Access tools to access data stored on database management systems that are twenty years old.

The Data Access Layer not only spans different DBMSs and file systems on the same hardware, it spans manufacturers and network protocols as well. One of the keys to a Data Warehousing strategy is to provide end-users with "universal data access". Universal data access means that, theoretically at least, end-users, regardless of location or Information Access tool, should be able to access any or all of the data in the enterprise that is necessary for them to do their job.

The Data Access Layer then is responsible for interfacing between Information Access tools and Operational Databases. In some cases, this is all that certain end-users need. However, in general, organizations are developing a much more sophisticated scheme to support Data Warehousing.

Data Directory (Metadata) Layer

In order to provide for universal data access, it is absolutely necessary to maintain some form of data directory or repository of meta-data information. Meta-data is the data about data within the enterprise. Record descriptions in a COBOL program are meta-data. So are DIMENSION statements in a FORTRAN program, or SQL Create statements. The information in an ERA diagram is also meta-data.

In order to have a fully functional warehouse, it is necessary to have a variety of meta-data available, data about the end-user views of data and data about the operational databases. Ideally, end-users should be able to access data from the data warehouse (or from the operational databases) without having to know where that data resides or the form in which it is stored.

Student Activity 1

1. What's metadata? What's a data dictionary?

Process Management Layer

The Process Management Layer is involved in scheduling the various tasks that must be accomplished to build and maintain the data warehouse and data directory information. The Process Management Layer can be thought of as the scheduler or the high-level job control for the many processes (procedures) that must occur to keep the Data Warehouse up-to-date.

Application Messaging Layer

The Application Message Layer has to do with transporting information around the enterprise computing network. Application Messaging is also referred to as “middleware”, but it can involve more that just networking protocols. Application Messaging for example can be used to isolate applications, operational or informational, from the exact data format on either end. Application Messaging can also be used to collect transactions or messages and deliver them to a certain location at a certain time. Application Messaging in the transport system underlying the Data Warehouse.

Data Warehouse (Physical) Layer

The (core) Data Warehouse is where the actual data used primarily for informational uses occurs. In some cases, one can think of the Data Warehouse simply as a logical or virtual view of data. In many instances, the data warehouse may not actually involve storing data.

In a Physical Data Warehouse, copies, in some cases many copies, of operational and or external data are actually stored in a form that is easy to access and is highly flexible. Increasingly, Data Warehouses are stored on client/server platforms, but they are often stored on main frames as well.

Data Staging Layer

The final component of the Data Warehouse Architecture is Data Staging. Data Staging is also called copy management or replication management, but in fact, it includes all of the processes necessary to select, edit, summarize, combine and load data warehouse and information access data from operational and/or external databases.

Data Staging often involves complex programming, but increasingly data warehousing tools are being created that help in this process. Data Staging may also involve data quality analysis programs and filters that identify patterns and data structures within existing operational data.

Student Activity 2

1. Explain the various layers of data warehouse architecture.

3.4 DATA WAREHOUSE OPTIONS

There are perhaps as many ways to develop data warehouses as there are organizations. Moreover, there are a number of different dimensions that need to be considered:

- Scope of the data warehouse
- Data redundancy
- Type of end-user

Figure3.2 shows a two-dimensional grid for analyzing the basic options, with the horizontal dimension indicating the scope of the warehouse, and the vertical dimension showing the amount of redundant data that must be stored and maintained.

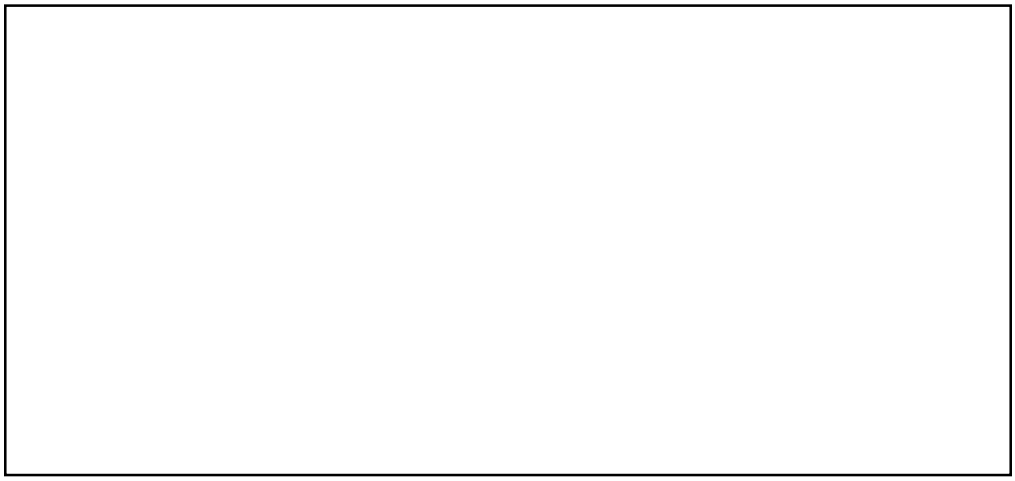


Figure3.2 - Data Warehouse Options

Data Warehouse Scope

The scope of a data warehouse may be as broad as all the informational data for the entire enterprise from the beginning of time, or it may be as narrow as a personal data warehouse for a single manager for a single year. There is nothing that makes one of these more of a data warehouse than another.

In practice, the broader the scope, the more value the data warehouse is to the enterprise and the more expensive and time consuming it is to create and maintain. As a consequence, most organizations seem to start out with functional, departmental or divisional data warehouses and then expand them as users provide feedback.

Data Redundancy

There are essentially three levels of data redundancy that enterprises should think about when considering their data warehouse options:

- "Virtual" or "Point-to-Point" Data Warehouses
- Central Data Warehouses
- Distributed Data Warehouses

There is no one best approach. Each option fits a specific set of requirements, and a data warehousing strategy may ultimately include all three options

"Virtual" or "Point-to-Point" Data Warehouses

A virtual or point-to-point data warehousing strategy means that end-users are allowed to get at operational databases directly using whatever tools are enabled to the "data access network" . This approach provides the ultimate in flexibility as well as the minimum amount of redundant data that must be loaded and maintained. This approach can also put the largest unplanned query load on operational systems.

As we will see, virtual warehousing is often an initial strategy in organizations where there is a broad but largely undefined need to get at operational data from a relatively large class of end-users and where the likely frequency of requests is low. Virtual data warehouses often provide a starting point for organizations to learn what end-users are really looking for. Figure 3 below shows a Virtual Data Warehouse within the Data Warehouse Architecture.

Central Data Warehouses

Central Data Warehouses are what most people think of when they first are introduced to the concept of data warehouse. The central data warehouse is a single physical database that contains all of the data for a specific functional area, department, division, or enterprise. Central Data Warehouses are often selected where there is a common need for informational data and there are large numbers of end-users already connected to a central computer or network. A Central Data Warehouse may contain data for any specific period of time. Usually, Central Data Warehouses contain data from multiple operational systems.

Central Data Warehouses are real. The data stored in the data warehouse is accessible from one place and must be loaded and maintained on a regular basis. Normally, data warehouses are built around advanced RDBMs or some form of multi-dimensional informational database server.

Distributed Data Warehouses

Distributed Data Warehouses are just what their name implies. They are data warehouses in which the certain components of the data warehouse are distributed across a number of different physical databases. Increasingly, large organizations are pushing decision-making down to lower and lower levels of the organization and in turn pushing the data needed for decision making down (or out) to the LAN or local computer serving the local decision-maker.

Distributed Data Warehouses usually involve the most redundant data and, as a consequence, most complex loading and updating processes.

Type of End-user

In the same sense that there are lots of different ways to organize a data warehouse, it is important to note that there are an increasingly wide range of end-users as well. In general we tend to think in terms of three broad categories of end-users:

- Executives and managers
- "Power" users (business and financial analysts, engineers, etc.)
- Support users (clerical, administrative, etc.)

Each of these different categories of user has its own set of requirements for data, access, flexibility and ease of use.

3.5 DEVELOPING DATA WAREHOUSES

Developing a good data warehouse is no different from any other IT project; it requires careful planning, requirements definition, design, prototyping and implementation. The first and most important element is a planning process that determines what kind of data warehouse strategy the organization is going to start with.

Developing a Data Warehouse Strategy

Before developing a data warehouse, it is critical to develop a balanced data warehousing strategy that is appropriate for its needs and its user population. Who is the audience? What is the scope? What type of data warehouse should we build?

There are a number of strategies by which organizations can get into data warehousing. One way is to establish a "Virtual Data Warehouse" environment. A Virtual Data Warehouse is created by: (1) installing a set of data access, data directory and process management facilities, (2) training the end-users (3) monitoring how the data warehouse facilities are actually used and then (4) based on actual usage, create a physical data warehouse to support the high-frequency requests.

A second strategy is simply to build a copy of the operational data from a single operational system and enable the data warehouse from a series of information access tools. This strategy has the advantage of being both simple and fast. Unfortunately, if the existing data is of poor quality and/or the access to the data has not been thought through, then this approach can create a number of significant problems.

Ultimately, the optimal data warehousing strategy is to select a user population based on value to the enterprise and do an analysis of their issues, questions and data access needs. Based on these needs, prototype data warehouses are built and populated so the end-users can experiment and modify their requirements. Once there is general agreement on the needs, then the data can be acquired from existing operational systems across the enterprise and/or from external data sources and loaded into the data warehouse. If it is required, information access tools can also be enabled to allow end-users to have access to required data using their own favorite tools or to allow for the creation of high-performance multi-dimensional information access systems using the core data warehouse as the basis.

In the final analysis, there is no one approach to building a data warehouse that will fit the needs of every enterprise. Each enterprise's needs are different as is each enterprise's context. In addition, since data warehouse technology is evolving as we learn more about developing data warehouses, it turns out that the only practical approach to data warehousing is an evolutionary one.

Evolving a Data Warehouse Architecture

The Data Warehouse Architecture in Figure 1 is simply a framework for understanding data warehousing and how the components of data warehousing fit together. Only the most sophisticated organizations will be able to put together such an architecture the first time out. What the Data Warehouse Architecture provides then is a kind of roadmap that can be used to design toward. Coupled with an understanding of the options at hand, the Data Warehouse

Architecture provides a useful way of determining if the organization is moving toward a reasonable data warehousing framework.

One of the keys to data warehousing is flexibility. It is critical to keep in mind that the more successful a data warehouse strategy is, the more end-users are going to want to add to it.

Designing Data Warehouses

Designing data warehouses is very different from designing traditional operational systems. For one thing, data warehouse users typically don't know nearly as much about their wants and needs as operational users. Second, designing a data warehouse often involves thinking in terms of much broader, and more difficult to define, business concepts than does designing an operational system. In this respect, data warehousing is quite close to Business Process Reengineering (BPR). Finally, the ideal design strategy for a data warehouse is often outside-in as opposed to top-down.

But while data warehouse design is different from what we have been used to, it is no less important. The fact that end-users have difficulty defining what they need as a bare minimum is no less necessary. In practice, data warehouse designers find that they have to use every trick in the book to help their users "visualize" their requirements. In this respect, robust working prototypes are essential.

3.6 MANAGING DATA WAREHOUSES

Data Warehouses are not magic-they take a great deal of very hard work. In many cases data warehouse projects are viewed as a stopgap measure to get users off our backs or to provide something for nothing. But data warehouses require careful management and marketing. A data warehouse is a good investment only if end-users actually can get at vital information faster and cheaper than they can using current technology. As a consequence, management has to think seriously about how they want their warehouses to perform and how they are going to get the word out to the end-user community. And management has to recognize that the maintenance of the data warehouse structure is as critical as the maintenance of any other mission-critical application. In fact, experience has shown that data warehouses quickly become one of the most used systems in any organization.

Management, especially IT management, must also understand that if they embark on a data warehousing program, they are going to create new demands upon their operational systems: demands for better data, demands for consistent data, demands for different kinds of data.

3.7 FUTURE DEVELOPMENTS

Data Warehousing is such a new field that it is difficult to estimate what new developments are likely to most affect it. Clearly, the development of parallel DB servers with improved query engines is likely to be one of the most important. Parallel servers will make it possible to access huge data bases in much less time.

Another new technology is data warehouses that allow for the mixing of traditional numbers, text and multi-media. The availability of improved tools for data visualization (business intelligence) will allow users to see things that could never be seen before.

Data Warehousing is not a new phenomenon. All large organizations already have data warehouses, but they are just not managing them. Over the next few years, the growth of data warehousing is going to be enormous with new products and technologies coming out frequently. In order to get the most out of this period, it is going to be important that data warehouse planners and developers have a clear idea of what they are looking for and then choose strategies and methods that will provide them with performance today and flexibility for tomorrow.

3.8 SUMMARY

1. Information technology itself has made possible revolutions in the way that organizations today operate throughout the world.

2. Every day organizations large and small create billions of bytes of data about all aspects of their business, millions of individual facts about their customers, products, operations and people.

3. Data Warehousing has grown out of the repeated attempts on the part of various researchers and organizations to provide their organizations flexible, effective and efficient means of getting at the sets of data that have come to represent one of the organization's most critical and valuable assets.

4. Operational systems are just what their name implies; they are the systems that help us run the enterprise operation day-to-day.

5. Informational systems have to do with analyzing data and making decisions, often major decisions, about how the enterprise will operate, now and in the future.

6. A Data Warehouse Architecture (DWA) is a way of representing the overall structure of data, communication, processing and presentation that exists for end-user computing within the enterprise.

7. The Information Access layer of the Data Warehouse Architecture is the layer that the end-user deals with directly.

8. The Data Access Layer of the Data Warehouse Architecture is involved with allowing the Information Access Layer to talk to the Operational Layer.

9. Meta-data is the data about data within the enterprise.

10. A virtual or point-to-point data warehousing strategy means that end-users are allowed to get at operational databases directly using whatever tools are enabled to the "data access network".

3.9 KEYWORDS

A Data Warehouse Architecture (DWA): is a way of representing the overall structure of data, communication, processing and presentation that exists for end-user computing within the enterprise.

Operating System: Operational systems process data to support critical operational needs.

Information Access layer: The Information Access layer of the Data Warehouse Architecture is the layer that the end-user deals with directly.

Data Access Layer: The Data Access Layer then is responsible for interfacing between Information Access tools and Operational Databases.

The Process Management Layer: It is involved in scheduling the various tasks that must be accomplished to build and maintain the data warehouse and data directory information.

Data Staging: The final component of the Data Warehouse Architecture is Data Staging. Data Staging is also called copy management or replication management, but in fact, it includes all of the processes necessary to select, edit, summarize, combine and load data warehouse and information access data from operational and/or external databases.

Scope of a data warehouse: The scope of a data warehouse may be as broad as all the informational data for the entire enterprise from the beginning of time, or it may be as narrow as a personal data warehouse for a single manager for a single year.

Distributed Data Warehouses: Data warehouses in which the certain components of the data warehouse are distributed across a number of different physical databases.

3.10 REVIEW QUESTIONS

1. What's metadata? What's a data dictionary?

2. How do we go about designing and building a DW architecture?

3. What is data analysis? What is data analysis used for?

4. What should be the features of a data ware house? Discuss.

5. Differentiate operational system and informational system with suitable example.

6. Explain the various layers of data warehouse architecture.

7. Write a short note on the futuristic direction of data warehousing.

3.11 FURTHER READINGS

Bezdek, J. C., & Pal, S. K. (1992). *Fuzzy models for pattern recognition: Methods that search for structures in data*. New York: IEEE Press

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (Eds.). (1996). *Advances in knowledge discovery and data mining.* AAAI/MIT Press.

Han, J., & Kamber, M. (2000). *Data mining: Concepts and techniques*: Morgan Kaufmann.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: Data mining, inference, and prediction:* New York: Springer.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. New Jersey: Prentice Hall.

Jensen, F. V. (1996). *An introduction to bayesian networks*. London: University College London Press.

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis*. New York: John Wiley.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine learning, neural and statistical classification*: Ellis Horwood.

Fukunaga, K., *Introduction to Statistical Pattern Recognition,* Academic Press, 1990

Groth, R., *Data Mining, A Hands-On Approach for business Professionals*, Prentice Hall, 1998 [with demo software]

Hand, D.J., Mannila, H., Smyth, P., *Principles of Data Mining*, MIT Press, [late] 2000

Jain, A. and Dubes, R., *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988

Jordan, M.I., *Learning in Graphical Models*, MIT Press, 1999

Kargupta, Hillol, and Chan, Philip, *Advances in Distributed and Parallel Knowledge Discovery*, MIT/AAAI Press, 2000

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*," New York: Springer-Verlag, 1994

Mitchell, T.M., *Machine Learning*, New York, NY: McGrawl Hill, 1997

Ripley, B.D., *Pattern Recognition and Neural Networks*, Cambridge, UK: Cambridge University Press, 1996

UNIT

4

DATA WAREHOUSE IMPLEMENTATION

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Know about major Tasks of Data Warehouse Implementation.
- Know about results and performance Targets of Data Warehouse Implementation.
- Illustrate Warehouse design.
- Know about various major Processes in Warehouse Design.
- Know about Various Problem areas.
- Know about different Custom Analysis Features.

U N I T S T R U C T U R E

4.1 INTRODUCTION

An objective for a Data Warehouse is to make all critical business data available in standard form for rapid inquiry, analysis and reporting. To achieve this, it is necessary to move data out of its non-standard form in existing applications by steps such as those outlined below. These steps were followed in a successful implementation which resulted in useful data being available within 3 months, and a full capability being available in 6. For most organizations, creating a Data Warehouse should not be a major undertaking.

In general most organizations have similar legacy applications needing to be converted. The more the existing applications adhere to some set of common data standards the easier the implementation should be.

For implementation, the three major tasks are:

1. Create Initial Extract of all data sources.
2. Standardize and clean the data.
3. Set up normalized Tables.

These major tasks are further detailed below.

1. Create Initial Extract of all data sources

1. ***Identify the critical data subject areas:*** Orders, Inventory, Invoices, Accounts payable, General Ledger, Customer, Vendor, Employee ... Identify the actual application files containing the related data.

2. Specify the Date, Flag, Number, Time, and Text formats to be used in converting the source file data to standard ASCII form. It should be noted that existing standards established by ANSI, ISO, and for EDI can be used as is, or readily adapted. As a result, this effort can be minor. See Neutral Form Data Standards.

3. ***Specify the target performance for creating extract files*** - Rate at which conversion must proceed to be acceptable. This is particularly important if the warehouse is to be for "near operational" data.

4. Document the field offsets, size, data types and short description or name of the fields in the extract files.

5. Create programs for each file or table to dump the source file contents to standard ASCII within conversion time targets. (Note that this is the only custom coding that may need to be performed by several groups. It is essentially a simple dump as if for a report, but with no headers, pagination or spacing. Because of language specific data types, it may need to be done in the programming language of the application and by staff familiar with the contents.)

2. Standardize and clean the data

1. Evaluate transfer times achieved and, if needed, specify acceptable deviations from standard transfer formats to meet time targets.

2. Set up source data for evaluation, field by field, character by character, to establish actual contents, mapping needed, acceptable data standards for the organization. A variety of tools can assist in this analysis, and for value mapping, normalization, etc., or they can be created as part of the project. See Data Warehouse oriented Data Analysis Tools

3. Establish organization-wide size and other standards for various common field types - name, address, city, postal code, ... Again consult documents from existing standards groups so as to minimize reinvention.

4. Establish content and format standards for critical Identifiers - Product, Employee, Customer, Vendor, Cost Center, Account, ...

5. Establish content, format, and size standards, for critical Codes and Flags - Order status, Employee type ... . (This is likely an on-going effort as more data needs are identified and more codes encountered in the extracts.)

6. Incorporate standards into conversion code on servers to change sizes, map content and formats, and create fully mapped, standard, versions of the data in the source record layouts.

7. Make clean, standard data accessible for analysis by loading to SQL or other warehouse specific data server. This provides access to existing data in standard form, but likely to differing levels of normalization. See normalization issues in The why of data standards - Do you really know your data?

3. Set up normalized Tables.

1. Define the normalized content of the required subject data sets.

2. Implement code to perform normalization of source files and create Tables in normalized form with common content, at least within the same subject areas.

3. Develop and load a common Data Dictionary of Elements and Code lists and Tables to define the new standard elements and support future data access by users.

4. Load data to SQL or other servers.

5. Evaluate results and implement further changes needed to complete cross-subject integration.

Resources Required

The actual numbers depend on the geographic distribution of the data sources, and the amount of data involved. In practice it has been found that for smaller organizations with a single location, it can all be done by two people part time for data volumes measure at less than a Gygabyte and less than 70 files. More resources may be needed for larger numbers of files and/or distributed data.

Schedule

On the basis of prior efforts, it is possible to reach the end of Step 2 within 3-6 months, even when creating all the required extract and mapping code from scratch. At this point, all the data is accessible but not necessarily normalized. This stage may also not include mapping of all values to common standards, largely because of the time and effort required to discuss and define new organization-wide standards for some of the critical elements. The constraints to reaching rapid agreement are usually organizational, rather than technical.

If applicable prior work has been done to define content of the normalized tables, parts of Step 3 can also be completed in this 3-6 month time period, resulting in normalized, cleaned data in standard form being made available from multiple sources.

Problem areas

The major problem areas encountered in following these steps to implementation have been as follows:

1. Processing Time to extract data from the source systems. This can prove a major constraint because of the need to use routines from the vendor packages for the extracts. The more proprietary the package and internal data structures and representation, the more likely it is that this constraint will be encountered. It may need to be resolved by redoing the code, or by transferring much of the standardization load to the servers after extract (or dumping) from the source files.
2. Inadequate or incorrect documentation of source data content. Often an unfortunate occurrence with the older packages and applications, it is solved by character by character analysis of all data extracted so as to determine exact content and nature of the invalid data..
3. Data mapping and Load times for creating clean normalized tables in an SQL server. To provide adequate processing speed, array processing was used for the data conversion involved in standardization, and high speed servers applied to the nightly load and re-index the data. See Data Warehouse Implementation and Array processing of commercial data - to drastically cut elapsed times

Results and performance targets

The result of completing this effort is that all applications data, organization-wide is available in standard form. In addition to allowing access to historical data, the achievable performance targets for data transfer and conversion are such that, for most organizations, their current data can be moved daily.

Because of the daily transfer of all data, the effect is to provide consistent access to what might be termed “near operational” data with the opportunity to replace existing applications reports by direct access to this consistent Warehouse data. A side benefit is reduced load on the “transaction” systems. In addition, the “standards” and content defined for the normalized Tables provides the basis for designing the replacement for the legacy applications - An alternative to just patching existing code to solve the year 2000 problem.

4.2 WAREHOUSE DESIGN

The Data Warehouse is designed to receive, clean, standardize and normalize data from a series of applications every night. In the process, it creates intermediate tables of the raw data as transferred, an additional version of the same data cleaned and standardized, and normalized versions in a flat file, ASCII delimited form, to load an SQL server. The processing is initiated by arrival of an ASCII file from the source VAX system, transferred over an FDDI link to a directory

on a dual Pentium server. Once the actual data file transfer is complete another empty file of the same name is transferred to a directory called "FIN". On the server, a small set of APL functions use the Windows timer to poll the directory at intervals the polling process on the server (DIRWATCH) detects receipt of files and adds a process request to a FIFO stack. One or more other EXEC processes monitor the stack (an APL component file) and invoke the required software to perform the cleanup and mapping of the data.

Running in parallel with the file transfers, the last file is converted within a few minutes of arrival. Should the transfer fail for any reason, the conversion can be re-initiated and all 560 MBytes of 50 files mapped and reloaded to the server in less than 1 hour. The APL process, which includes creation of the intermediate files, runs up to three processes on the dual Pentium systems under NT. to complete the process in some 20 minutes, faster than the SQL load despite doing more work.

The major programs invoked from the FIFO stack are UNMIX, which splits out the multiple record type files, and UNTXT, which performs the major clean up, data standardization and normalization of the source files, including those created by UNMIX.

The objective in running the Data Warehouse is to make all current operational data available to the users, and to allow for retention of historical snapshots at intervals. Due to the data sources being several purchased packages plus home developed applications, the clean up and standardization performed is essential to success as no data standards exist in the source data. In addition, the Data Warehouse supports manual clean up of the source data by providing a simple mechanism to identify problem fields which contain data that even the extensive automatic clean up rules can not correct.

The Query by Mail capability is initiated in a similar manner to the main processing. One (VBASIC) process monitors the mail system running under NT and transfers any received messages to a MAILIN directory. (A VBASIC program had to be used instead of APL because of the need to access the mail API in NT). The same DIRWATCH process monitors this directory and places requests on the stack. The same EXEC process monitoring the stack initiates the MAIL program as requested (rather than the clean up process). The MAIL program places the results of the data analysis as a text file in a MAILOUT directory to be picked up and mailed back by the VBASIC program. In all cases the polling is controlled by the Windows timer, placing a minimal load on the system during the "sleep" period.

The major processes then are:

- ***DIRWATCH:*** Polls directories FIN and MAILIN for any files. Based on the name and extension, adds an action request to the FIFO stack, a component file.
- ***EXEC:*** Polls the stack and copies in the functions to perform the UNMIX, UNTXT or MAIL processes, reloading EXEC when complete. One or more EXEC processes can run on one or more CPUs.
- ***UNMIX:*** Splits one file of many record types into individual flat files.
- ***UNTXT:*** Performs major cleanup, standardization and normalization of the data.
- ***MAIL:*** Supports analyze and query capabilities against the raw and cleaned data.

Some of the design objectives are that every process should be so fast that no extensive batch streams or checkpoint restart procedures are needed. At most, a failure entails re-initiating a process by point and click so as to re-run the whole process. Achieving this performance does entail limiting the standardization performed on the source systems, with most of it being done on the server. While this complicated the rules to be implemented in the clean up (dates and numbers appear in several formats, files have multi record types, the same field is various sizes, etc.,) it is still possible to meet the other objective of using data independent code for all the mapping and have one simple program process all files. The same program pointed at a Meta Data file (data about the data), and the associated data file, processes any one of the 50+ files and generates over 70 normalized files.

The initial implementation was done with Dyalog APL under Windows for Workgroups. Processing was performed using a PC farm of 486 class machines with the files on a server. The PC Farm had

several PCs on the network set up such that each could run one or more processes to monitor the FIFO stack and process requests. Due to inadequate reliability of the network at the time, the process was subsequently converted to Manugistics APL under NT, with all processes running on the dual Pentium NT server.

Student Activity 1

1. Explain architectural design of data warehouse implementation.

4.3 CUSTOM ANALYSIS FEATURES

The design of the data field analysis capability is customized for the type of data error or discrepancy encountered. In general, the data capture systems apply validation rules, but these are not consistent. A Flag value which should be 2-valued, “Yes” or “No”, preferably stored as a “1” or “0”, is often 3 or 4 valued, “Y” and “y” for “Yes’, “N” or “n” for “No” with a default or missing value of space, “ “ or period “.”. The next Flag field can have values of “*” or “x” or “T” for the “Yes” or “True” condition, and so on. Even within a single record with several Flag values, there can be several different representations.

Part of the analysis requirement therefore is to determine character content of fields and frequency of occurrence of each value. The latter assists in determining the size of the problem and whether the source is occasional errors or something more serious. The specific nature of any data errors encountered assists in defining the clean up rules to be applied.

An example of the output of the analysis is shown in Figure 4.1. It provides a summary of the field content, how many values are empty, the content of each character position in the field, and a sample of the frequency of occurrence of the content, alpha sorted and also sorted by frequency of occurrence. In the case of shorter Code or Flag fields, a list of all the unique values is given.

Type	Description		
PC	Zip Code		
Field	Size	Empty	PC Content
7 1262	9	131	10 -.0123456789ABCDEFGHJKLMOPQRSUVY[\deilovx

Ch	Empty	PC	Content
1	138	10	0123456789BCEHJLMOSUV[\
2	135	10	.0123456789DGeo
3	137	10	0123456789ABCDHJKRVYdl
4	145	11	-.0123456789ei
5	145	11	0123456789AEFGPRSvx
6	1212	96	-.012345689BMQRS
7	1232	97	0123456789G
8	1240	98	0123469
9	1240	98	01235678

Value	Nos		
131		131	
06107	1	10604	10
01920	1	06107	8
02072	2	60091	7
03818	1	02154	6
[D	3	V6H3G2	1
[D009	1	[D009	1
[D110	1	[D110	1
\1452	1	\1452	1

In the Figure 4.1, the data on:

Type	Description
PC	Zip Code

Contd..

Field	Size	Empty	PC	Content
7	1262	9	131	10 -.0123456789ABCDEFGHJKLMOPQRSUVY[\deilovx

Figure4.1 - Sample result of Analyze

Prints out some of the meta Data - it is a PC (Postal Code), called “Zip Code” it is the 7th item, with 1262 values, 9 characters wide of which 131 or 10 PC are empty. The actual range of characters found in the field is then shown.

The next section gives more details of the content for each character in the field and the number and percent of spaces. The final 4 columns provide a sample of the content and frequency of occurrence of the values in the field, first alpha sorted and then by frequency. This is especially useful in determining the nature of any “funnies” in the field.

The resulting data is sufficient to indicate the nature of many of the problems. The kind of irregularity often found is:

- Numbers in Name fields
- Numbers in City name fields (valid for some cities)
- Brackets or other symbols in Postal Codes
- Lower case letters in Code fields
- Invalid Codes in specific code fields
- Incorrect usage of fields
- Characters in number fields

And so on.

The analyzer output provides enough information to allow identification of the real “Type” code to be applied to the field, and of the clean up rules to be formulated. (The source dictionary, if any, often miss classified the data type). It also provides data to use in selecting records with invalid data, usually using a “HAS” selection. The “HAS” selection finds any value which HAS a given character anywhere in the field. In almost all cases the “bad” data is in a small fraction of the records and easily selected for correction. Fortunately, the automatic clean up can handle the higher volume transformations needed to standardize the data.

4.4 SUMMARY

1. An objective for a Data Warehouse is to make all critical business data available in standard form for rapid inquiry, analysis and reporting
2. For implementation, the three major tasks are:
   - ♦ Create Initial Extract of all data sources.
   - ♦ Standardize and clean the data.
   - ♦ Set up normalized Tables.
3. Make clean, standard data accessible for analysis by loading to SQL or other warehouse specific data server.
4. Develop and load a common Data Dictionary of Elements and Code lists and Tables to define the new standard elements and support future data access by users.
5. The more proprietary the package and internal data structures and representation, the more likely it is that this constraint will be encountered.
6. The objective in running the Data Warehouse is to make all current operational data available to the users, and to allow for retention of historical snapshots at intervals.

7. Some of the design objectives are that every process should be so fast that no extensive batch streams or checkpoint restart procedures are needed.

8. To provide adequate processing speed, array processing was used for the data conversion involved in standardization, and high speed servers applied to the nightly load and re-index the data.

9. Because of language specific data types, it may need to be done in the programming language of the application and by staff familiar with the contents.

10. In general, the data capture systems apply validation rules, but these are not consistent.

4.5 KEYWORDS

EXEC: Polls the stack and copies in the functions to perform the UNMIX, UNTXT or MAIL processes, reloading EXEC when complete. One or more EXEC processes can run on one or more CPUs.

UNMIX: Splits one file of many record types into individual flat files.

UNTXT: Performs major cleanup, standardization and normalization of the data.

MAIL: Supports analyze and query capabilities against the raw and cleaned data.

4.6 REVIEW QUESTIONS

1. What do you mean by data warehouse implementation?
2. What are the main steps to implements a data warehouse? Explain with the help of suitable example.
3. Describe the major problem areas encountered in the implementation of data warehouse.
4. Explain architectural design of data warehouse implementation.
5. List down some the main issues involved in the implementation of a data warehouse in an organization.
6. Write short note on:
    a. Managing the project and environment
    b. Organizational implications of data warehousing
    c. Data warehouse management
    d. Normalized tables

4.7 FURTHER READINGS

Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurasamy, “Advances in Knowledge Discovery and Data Mining”, AAAI Press/ The MIT Press, 1996.

J. Ross Quinlan, “C4.5: Programs for Machine Learning”, Morgan Kaufmann Publishers, 1993.

Michael Berry and Gordon Linoff, “Data Mining Techniques (For Marketing, Sales, and Customer Support), John Wiley & Sons, 1997.

Sholom M. Weiss and Nitin Indurkhya, “Predictive Data Mining: A Practical Guide”, Morgan Kaufmann Publishers, 1998.

Alex Freitas and Simon Lavington, “Mining Very Large Databases with Parallel Processing”, Kluwer Academic Publishers, 1998.

A. K. Jain and R. C. Dubes, “Algorithms for Clustering Data”, Prentice Hall, 1988.

V. Cherkassky and F. Mulier, “Learning From Data”, John Wiley & Sons, 1998.

Han, J., & Kamber, M. (2000). *Data mining: Concepts and techniques*: Morgan Kaufmann.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: Data mining, inference, and prediction:* New York: Springer.

UNIT 5

BUILDING DATA WAREHOUSING TEAM

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Know about building Data Warehouse System.
- Understand the need to cover to build a Warehouse.
- Have a look at a Data Warehouse roles Checklist.
- Know about different areas of responsibility.
- Know about the trainer and Technical writer.
- Correlate different Vendor relations person.

U N I T S T R U C T U R E

5.1 INTRODUCTION

Building data warehouse systems for an entire corporation is very different than building a data warehouse or data mart for a single department. In addition to the increased scale, number of participants, and integration complexities, there is also the challenge of your own team.

DW teams will usually go though three distinct organizational stages. The **first stage** could be classified as a pilot project within a small division or department, using a five-ten person team of existing IT staff that has little or no previous experience building DW systems. Success on that project then leads to larger and more ambitious DW projects, requiring a corresponding expansion of the team during the **second stage** of organizational development. Usually in Stage 2, more of the same type of staff are simply added to the Stage 1 team. When the team grows into a corporate DW team with enterprise-wide responsibilities, it enters its **third stage** of development. The evolution to Stage 3 responsibilities has profound staffing and organizational implications.

During the first stage, it is critical to have IT staff who know current systems, data sources, and users very well and who are quick learners that can work in an unstructured rapid application development environment. When a DW team is in Stages 1 and 2 of its development, the typical line-IT person, with experience developing systems for local users, is fine. Basically, they are developing a data warehouse from feeder systems that they know well, and they are able to work with local end users and IT contacts that have been established over a long period of time.

During the third stage, these traits remain an important element of the team composition, but need to be complemented with different types of individuals who excel at large-scale standardized systems development, data integration, and training. These individuals also have to develop the capabilities of a consulting organization, as opposed to continuing to operate as a local IT shop. In effect, while the technical steps necessary to build a DW have not changed, the environment in which this is to be done has changed radically.

Some of the new challenges that now have to be faced may include:

- The user can be 2000 miles away, requiring frequent out-of-town trips
- The DW team members are viewed as corporate “outsiders” by the local divisional IT shops
- A new network of personal contacts (user and IT) has to be developed quickly
- New legacy system environments have to be learned

When assuming corporate responsibilities, the assignment of roles and responsibilities between the corporate DW team and any local/divisional IT teams also becomes a critical new aspect of the organization. Clearly articulating who is responsible for what prevents misunderstandings and conflicts up front.

In closing, there have been many Stage 1 and 2 data warehouse teams that have become a victim of their own success when trying to take on Stage 3 responsibilities without the appropriate adjustments in team composition. Careful consideration of the evolving job responsibilities will allow you to maximize the potential of each person, while integrating vital new skills into a successful corporate DW team.

The following roles are typical for a data warehouse project:

- ***Project Manager:*** This person will oversee the progress and be responsible for the success of the data warehousing project.
- ***DBA:*** This role is responsible to keep the database running smoothly. Additional tasks for this role may be to plan and execute a backup/recovery plan, as well as performance tuning.
- ***Technical Architect:*** This role is responsible for developing and implementing the overall technical architecture of the data warehouse, from the backend hardware/software to the client desktop configurations.

- ***ETL Developer**:* This role is responsible for planning, developing, and deploying the extraction, transformation, and loading routine for the data warehouse.
- ***Front End Developer**:* This person is responsible for developing the front-end, whether it be client-server or over the web.
- ***OLAP Developer**:* This role is responsible for the development of OLAP cubes.
- ***Trainer**:* A significant role is the trainer. After the data warehouse is implemented, a person on the data warehouse team needs to work with the end users to get them familiar with how the front end is set up so that the end users can get the most benefit out of the data warehouse system.
- ***Data Modeler**:* This role is responsible for taking the data structure that exists in the enterprise and model it into a schema that is suitable for OLAP analysis.
- ***QA Group**:* This role is responsible for ensuring the correctness of the data in the data warehouse. This role is more important than it appears, because bad data quality turns away users more than any other reason, and often is the start of the downfall for the data warehousing project.

The above list is roles, and one person does not necessarily correspond to only one role. In fact, it is very common in a data warehousing team where a person takes on multiple roles. For a typical project, it is common to see teams of 5-8 people. Any data warehousing team that contains more than 10 people is definitely bloated.

5.2 THE DATA WAREHOUSE TEAM CHECKLIST

Let's take a high-level look at a data warehouse roles checklist; these are the areas of responsibility you need to cover to build a warehouse:

- Data Warehouse Project Manager
- Data Warehouse Architect
- Database Administrator
- System Administrator
- Data Migration Specialist (moving the Legacy data)
- Legacy System Specialist
- Data Transformation/Grooming Specialist
- Data Provision Specialist
- Mart Development Leader
- Quality Assurance/ Testing Specialist
- Infrastructure Specialist
- Power User
- Trainer
- Technical Writer
- Public Relations Person
- Corporate Sponsors
- Help Desk Functionality
- Tools Specialist
- Vendor Relations Person

Your particular situation will determine how these roles are allocated. Some may require an entire team, while others may be shared between one or two people.

As you can see from reviewing the data warehouse roles checklist, you must think through and account for many things. Your level of success will depend on how you manage the resources available to you to accomplish these tasks.

Roles Needed for Your Data Warehouse Team

As you read the detail of these roles, remember your particular situation and the personalities involved. Review your situation from two different perspectives: the function and the person. For example, you must decide who will be responsible for the function of a system administrator, who should be in charge of whether or not the backups are working, and if system patches must be installed. Also, you must determine if the people you are considering have the appropriate skills for the necessary tasks.

Student Activity 1

1. Explain the concept of Data Warehousing team with the help of suitable example.

5.3 DATA WAREHOUSE PROJECT DIRECTOR

Abilities/ Responsibilities

- The perspective of this person is that of a bird flying over a maze. This represents the project director's need to take a global view of the project to see where it is headed. Like the bird, a project director sees the paths of the maze and where they all head, yet he or she may not see every pitfall within the maze.
- Strong leadership skills are a prerequisite. The project director should constantly communicate the vision of the project. This ability to motivate and inspire will, in effect, energize all team members and enable them to overcome any barriers to change. These barriers will come in many forms, from political obstacles to resource problems.
- He should be able to establish a vision for the future and relate this vision to the needs of the business. Must develop strategies to produce all necessary changes to accomplish the vision.
- Must have strong political skills. The project director must know how to package and sell the efforts of the team to senior management. He or she must understand the importance of each business unit within the company and ensure the priorities of the warehouse project are in synch with the priorities of the business.
- Excellent communication skills, including written ability, oral ability, and the ability to listen effectively is also highly desirable.
- Must have strong financial skills. The project director will have ownership of the warehouse budget, as well as ownership of the preparation of the budget. He or she will also have to ensure this budget adheres to the standard corporate practices.
- Have command over the latest technology. The project director must be able to talk the talk, but not necessarily walk the walk.
- Be responsible to senior management for the status of the project. The buck stops with the project director. If anything goes wrong, the problem rests on his or her shoulders.
- Must be responsible for monitoring the major milestones of the project and for making certain the milestones are in tune with corporate needs.

5.4 DATA WAREHOUSE PROJECT MANAGER

The most important attribute of a great project manager is the ability to keep an eye on milestones constantly and never lose sight of the project commitments.

Abilities/Responsibilities

- This person is not a corporate politician but, instead, a person with strong team-building skills. The project manager understands how to make a group of people work together as one toward a common goal.
- Must have strong organizational and planning skills.
- He should have strong management skills and understanding of technology. On the job, the project manager will constantly be dealing with team members who have varied skill sets. He or she must understand technical and non-technical jargon with equal ease.
- Must be able to plan and allocate resources. People will quit a project, become burned out over time, or just want a vacation. A good project manager deals with these common issues so they do not reach crisis proportions.
- Must be a diplomat. The project manager will constantly deal with stakeholders who feel their needs are the most critical. A good project manager must assess each stakeholder's needs and see where those needs fit into the overall vision of the project.
- Must know how to deliver bad news and how to circulate good news.
- Must be able to set up and control a meeting.
- Must be in control of the project scope. The project manager must be able to control scope creep. Scope creep happens when requests come in that will affect the delivery date of a project. The project manager must know how to take this problem to the appropriate ears and then he or she must obtain the necessary documentation to support the approval of the scope creep.
- Must be able to do appropriate risk management. For example, when the project manager is working with new technology, he or she must understand who this new technology may affect the project plans. In a data warehouse environment, this is critical because you are constantly dealing with bleeding edge technology. Bleeding edge technology is state-of-the-art and introduces new ways to do something with new functionality. If you don't want your project to die from this, you must understand risk assessment.
- The project manager must have good listening ability and pay attention; he or she must know when to interrupt and when to listen.
- The ability to harmonize the team and to reconcile disagreements is a must.
- Must be able to play the role of gate keeper. The project manager helps facilitate participation by the team members. He or she must constantly monitor to see if everyone is satisfied with the approaches. The project manager should make certain the team understands the priorities.
- Must recognize the contribution of every team member.
- Must be an excellent negotiator- able to admit an error and to look for the compromise that will get the project back on track.
- Must determine the performance of the project.
- Must realize that sick projects will not get better on their own. The project manager knows sick projects require corrective action immediately.
- Must realize he or she cannot be a developer at the same time. The position of a data warehouse project manager is a full-time, dedicated position.

5.5 DATA PROVISION SPECIALIST

Abilities/Responsibilities

- Must know how to phrase questions to determine the end users needs.

- Must have excellent writing skills and the ability to convey the end users needs to the technical teams.
- Must take responsibility for defining the scope statements. Must work with the technical teams and the user community to obtain the necessary consensus so the project moves forward.
- Must have excellent people skills. Data provisions specialists not only must be diplomats, they must also be eloquent communicators. Much of the success of the data warehouse project hinges upon the ability of the data provisions specialist to work well with the user community.
- Must have enough political savvy to identify key people.

5.6 DATA WAREHOUSE ARCHITECT

Abilities/Responsibilities

- Must take ownership of the entry relationship diagram and all the corresponding documentation. An example of other documentation is a corporate data dictionary with all the attributes, table names, and constraints.
- Must possess a thorough knowledge of database design and associated tools.
- Must have good peer-to-peer communication skills. Does not require corporate political skill.
- Must have good business analysis skills.
- Must be responsible for managing the data warehouse metadata, which is data about data. The data warehouse architect is king or queen of the data.
- Must be the watchdog of data warehouse standards. It is his or her responsibility to maintain all the documentation about the warehouse database design. The data warehouse architect is responsible for adopting database design and naming standards; he or she must also make certain these are followed. These standards should be used in all future application development.

5.7 DATABASE ADMINISTRATOR

Abilities/Responsibilities

- Must have some political skills. For example, the database administrator might not want to upgrade the system on the busiest business day of the year. Common sense is a requirement.
- Must ensure appropriate backup and recovery procedures are in place to meet the business requirements. If a project is not backed up and a database is lost, a month of the project team's work could be lost.
- Must be responsible for putting security in place to make certain only the right people can look at the right data. In other words, the DBA uses Oracle mechanisms to control all access to the database. The database administrator works closely with the data architects to implement the database design.
- Must work closely with the technical team to ensure they adhere to corporate policies and procedures pertaining to the database. This includes development of policies to control the movement of applications onto a production database.
- Must monitor the growth of the database to ensure the smooth running of daily activities.
- Must monitor performance. This is a critical function of the DBA. He or she must establish baselines and compare the database performance against the baselines to ensure that it is performing adequately.

- Must tend to daily administration of the database.
- Must be able to tackle issues as soon as they spring up. The database administrator position is one of the most technically challenging roles that exists within all the teams.
- Must have minimum writing skills, although e-mail etiquette is a plus.
- Must be available always.
- Must work closely with the system administrator to install all database software and patches.

5.8 SYSTEM ADMINISTRATOR

Abilities/Responsibilities

- Must have some political skills. For example, the system administrator may not want to upgrade the system on the busiest business day of the year. Common sense is a critical requirement.
- Must be available all the time so that; when things happen, SA can respond immediately.
- Must ensure appropriate backup and recovery procedures are in place to meet the business requirements.
- Must take responsibility for putting security in place to make certain only the right people can look at the system. In today's environment, the SA typically gets involved in firewall creation.
- Must work closely with the technical team to make certain they adhere closely to corporate policies and procedure pertaining to the system. This includes development of policies to control the movement of application onto a production system.
- Must be able to do performance monitoring; this is another critical function of the system administrator. He or she must establish baselines and compare the performance of the system against the base to ensure it is performing adequately.
- Must have prior experience in working in a technically challenging role; the SA role is the most technically challenging within all the teams.
- Must have minimum writing skills, although e-mail etiquette is a plus.
- Must be able to access the system remotely; when something happens after hours, the SA must be able to respond.

5.9 DATA MIGRATION SPECIALIST

Abilities/Responsibilities

- Must have intimate knowledge of current legacy system and understand the data stored within the system. This knowledge must include the internal structure of the existing legacy systems or the ability to analyze the legacy system to learn the data structures. This is a highly technical area, where strong programming skills are a plus.
- Need not be political. Like the rat mentioned earlier, the data migration specialist must know how to navigate through the maze to find the cheese.
- Must be competent with legacy tools. Communication or writing skills are not important. This is a highly technical slot.
- Must work closely with the legacy technology specialist to learn necessary information.

5.10 DATA TRANSFORMATION/GROOMING SPECIALIST

Abilities/Responsibilities

- Must be highly technical. The data transformation/ grooming specialist requires a strong working knowledge of SQL* Loader and PL/SQL.
- Must be a perfectionist. The data transformation/ grooming specialist or team contains the "bean counters" of the technical world.
- Must develop the code needed to groom the data into the new data structure, meeting all the new standards.
- Must work closely with the data architect and data provision specialist to make certain all business needs are met.

5.11 DATAMART DEVELOPMENT LEADER

A datamart, by definition, is a subset of the data warehouse. The required skills for the datamart development leader closely mirror those of the data provision specialist. The only difference is this effort concentrates on a business area. For example, the datamart development leader might be working on the marketing datamart. His or her effort is focused on how to present the key components of the warehouse in which the marketing group is interested. The abilities and responsibilities are the same as the data provision specialist.

5.12 QUALITY ASSURANCE/TESTING SPECIALIST

Abilities/Responsibilities

- Must be responsible for creating and reviewing all test plans. This includes identifying the appropriate warehouse team members and members of the user community who should participate in the quality/ testing process.
- Must be aware this is a cooperative effort of all the shareholders. To obtain a quality product, the quality assurance testing specialist or team and all the stakeholders must make a concentrated effort throughout the project. A quality product will not be obtained if the effort occurs only at the end.
- Need not be political, except when dealing with team members who always have good reasons as to why they do not have to comply with the rules. We suggest you have the quality team report to the project manager. Otherwise, they will not be effective.
- Need not be highly technical role. The position of quality assurance/ testing specialist is primarily one of user testing.
- For this process to work, a plan must exist. Without proper preparation you will not achieve the results you want.

5.13 INFRASTRUCTURE SPECIALIST

Abilities/Responsibilities

- Must be able to see the bigger picture. For example, don't order a personal computer (PC) for the field a year ahead of time. And you don't give end users a PC without making certain they know how to use it.
- Must be moderately technical. It helps if the infrastructure specialist can talk the terminology.
- Must have excellent organization skills. He or she must be able to hold people's feet to the fire. If the network specialist promises to have network software installed on Thursday, then the infrastructure person needs to make certain it happens.

5.14 POWER USER (LEGACY SYSTEM SPECIALIST)

Abilities/Responsibilities

- Must have intimate working knowledge of the current legacy system.
- Must have strong working knowledge of application, but technical ability is not a requirement.
- Must have strong communication skills because you want them to help teach the new system to other end users.
- Must have political savvy and build connections. You want them to spread the good news. A classic trait of a power user is that he is someone the rest of the organization uses for information and he or she becomes the answer person within the department.
- May or may not be technical, yet power users are adept at making the system work for them.

5.15 TRAINER

Abilities/Responsibilities

- Must have excellent communication skills and infinite patience.
- Must have excellent user's knowledge of the warehouse and its tools.
- Must have excellent writing skills. A difference exists between good technical documentation and good training materials.
- Must have the ability to laugh. A good smile is a plus.

5.16 TECHNICAL WRITER

Abilities/Responsibilities

- Must have excellent communication skills and infinite patience because the technical writer is dealing with developers with little to no program documentation.
- Must have good working knowledge of the warehouse.
- Must write clearly and concisely; must employ good standards within the documentation.
- Need not be political.
- Must have a working vocabulary of data warehousing.
- Must be responsible. This is a professional skill set and you want it done right.

5.17 PUBLIC RELATIONS PERSON

Abilities/Responsibilities

- Must know how to let senior management win the golf game.
- Must have excellent communication and presentation skills.
- Should have attended a ‘dress for success’ seminar.
- Should know how to turn a challenge into an opportunity.
- Must know how to turn a project delay into a respite.
- Must be a great politician, with an ear constantly against the wall.
- Must be loyal to the cause.

- Must like to travel, to get the good word spread.
- Must never talk technical as this frightens the non-technical community.

5.18 HELP DESK FUNCTIONALITY

Abilities/Responsibilities

- Must be a trained person who understands how to use the warehouse properly.
- Must have infinite patience and understanding.
- Should have good technical knowledge so he or she can communicate the problem to the team.
- Must be organized.
- Must have excellent communication skills and know the terminology of the community of users.

5.19 TOOLS SPECIALIST

Abilities/Responsibilities

- Must be highly technical.
- Should be aware and have a good working knowledge of the core third-party tools in the market place.
- Should have the ability to choose the right tool for the right job. For example, in the Oracle arena, this person would know when it makes sense to use PL/SQL compared to SQL* Plus.
- Must be a worker rat. The tools specialist builds the maze and, thus, needs no political skills.

5.20 VENDOR RELATIONS PERSON

Abilities/Responsibilities

- Must have good technical skills and understand the warehouse requirements.
- Must be able to develop criteria that can be used to determine if a vendor product makes sense. From Day One, the vendor relations person should be aware of the core data warehouse products.
- Must have good communication skills.
- Must have good writing skills.
- Must have good negotiation skills, which includes the ability to review the contracts.
- Must understand the political relation between the ventures and your company.

As you can see, many roles and individuals are necessary for a successful data warehouse project.

5.21 HOW TO SUCCEED IN DATA WAREHOUSING

More and more companies are using data warehousing as a strategic tool to help them win new customers, develop new products, and lower costs. Searching through mountains of data generated by corporate transaction systems can provide insights and highlight critical facts that can significantly improve business performance. Until recently, data warehousing has been an option mostly for large companies, but the reduction in the cost of warehousing technology makes it practical, often even a competitive, requirement for smaller companies as well. Turnkey integrated analytical solutions are reducing the cost, time, and risk involved in implementation. While

access to the warehouse was previously limited to highly trained analytical specialists, today corporate portals are making it possible to grant access to hundreds, even thousands of employees.

1. ***Recognize that the job is probably harder than you expect:*** Experts frequently report that 30 to 50 percent of the information in a typical database is missing or incorrect. This situation may not be noticeable or may even be acceptable in an operational system that focuses on swiftly and accurately processing current transactions. But it's totally unacceptable in a data warehousing system designed to sort through millions of historical records in order to identify trends or select potential customers for a new product. And, even when the data is correct, it may not be usable in a data warehouse environment. For example, legacy system programmers often use shortcuts to save disk space or CPU cycles, such as using numbers instead of names of cities, which makes the data meaningless in a generic environment. Another challenge is that database schema often change over the lifecycle of a project, yet few companies take the time to rebuild historical databases to account for these changes.

2. ***Understand the data in your existing systems***: The first step in any data warehousing project should be to perform a detailed analysis of the status of all databases that will potentially contribute to the data warehouse. An important part of understanding the existing data is determining interrelationships between various systems. The importance of this step lies in the fact that interrelationships must be maintained as the data is moved into the warehouse. In addition, the implementation of the data warehouse often involves making changes to database schema. A clear understanding of data relationships among heterogeneous systems is required to determine in advance the impact of any such change. Otherwise, it's possible for changes to create inconsistencies in other areas that ripple across the entire enterprise, creating enormous headaches.

3. ***Be sure to recognize equivalent entities***: One of the most important aspects of preparing for a data warehousing project is identifying equivalent entities and heterogeneous systems. The problem arises from the fact that the same essential piece of information may appear under different field names in different parts of the organization. For example, two different divisions may be servicing the same customer yet have the name entered in a slightly different manner in their records. A data transformation product capable of fuzzy matching can be used to identify and correct this and similar problems. More complicated issues arise when corporate entities take a conceptually different approach in the way that they manage data. This situation frequently occurs in cases of a merger. The database schema of the two organizations was established under the influence of two entirely different corporate cultures. Establishing a common database structure can be just as important as merging the corporate cultures and crucial to obtaining the full effects of synergy.

4. ***Emphasize early wins to build support throughout the organization***: The availability of a wide range of off-the-shelf solutions has made it possible to drastically reduce cost and lead-time requirements for data warehousing applications. Off-the-shelf solutions won't usually complete project objectives, but they often can be used to provide point solutions in a short time that serve as a training and demonstration platform and most importantly, build momentum for full-scale implementation. Even for the largest scale applications, technology surveys should be performed in order to maximize the use of pre-built technology.

5. ***Consider outsourcing your data warehouse development and maintenance***: A large percentage of medium and large companies use outsourcing to avoid the difficulty of locating and the high cost of retaining skilled IT staff members. Most data warehousing applications fit the main criteria for a good outsourcing project — a large project that has been defined to the point that it does not require day-to-day interaction between business and development teams. There have been many cases where an outsourcing team is able to make dramatic improvements in a new or existing data warehouse. Typically, these improvements do not necessarily stem from an increased level of skill on the part of the outsourcing team, but rather flow from the nature of outsourcing. The outsourcing team brings fresh ideas and a new perspective to their assignment and is often able to bring to bear methods and solutions that they have developed on previous assignments. The outsourcing team also does not have to deal with manpower shortages and conflicting priorities faced by the previous internal team.

Building a data warehouse is no easy task. Reliance on technology is only a small part in realizing the true business value buried within the multitude of data collected within an organization's business systems. The right people, methodology, and experience are critical. Data warehouses touch the organization at all levels, and the people that design and build the data warehouse must be capable of working across the organization as well. The industry and product experience of a diverse team coupled with a business focus and proven methodology may be the difference between a functional system and true success.

Student Activity 2

1. List down main abilities/responsibilities of the various roles needed to implement the concept of data warehousing in an organization.

5.22 SUMMARY

- Building data warehouse systems for an entire corporation is very different than building a data warehouse or data mart for a single department.
- DW teams will usually go though three distinct organizational stages.
- The first stage could be classified as a pilot project within a small division or department, using a five-ten person team of existing IT staff that has little or no previous experience building DW systems.
- Success on that project then leads to larger and more ambitious DW projects, requiring a corresponding expansion of the team during the second stage of organizational development.
- The evolution to Stage 3 responsibilities has profound staffing and organizational implications.
- OLAP Developer is the role responsible for the development of OLAP cubes.
- More and more companies are using data warehousing as a strategic tool to help them win new customers, develop new products, and lower costs.
- The first step in any data warehousing project should be to perform a detailed analysis of the status of all databases that will potentially contribute to the data warehouse.
- A data transformation product capable of fuzzy matching can be used to identify and correct this and similar problems.
- Building a data warehouse is no easy task. Reliance on technology is only a small part in realizing the true business value buried within the multitude of data collected within an organization's business systems.

5.23 KEYWORDS

Project Manager: This person will oversee the progress and be responsible for the success of the data warehousing project.

DBA*:* This role is responsible to keep the database running smoothly. Additional tasks for this role may be to plan and execute a backup/recovery plan, as well as performance tuning.

Technical Architect*:* This role is responsible for developing and implementing the overall technical architecture of the data warehouse, from the backend hardware/software to the client desktop configurations.

ETL Developer*:* This role is responsible for planning, developing, and deploying the extraction, transformation, and loading routine for the data warehouse.

Front End Developer*:* This person is responsible for developing the front-end, whether it be client-server or over the web.

OLAP Developer*:* This role is responsible for the development of OLAP cubes.

***Trainer**:* A significant role is the trainer. After the data warehouse is implemented, a person on the data warehouse team needs to work with the end users to get them familiar with how the front end is set up so that the end users can get the most benefit out of the data warehouse system.

***Data Modeler**:* This role is responsible for taking the data structure that exists in the enterprise and model it into a schema that is suitable for OLAP analysis.

***QA Group**:* This role is responsible for ensuring the correctness of the data in the data warehouse. This role is more important than it appears, because bad data quality turns away users more than any other reason, and often is the start of the downfall for the data warehousing project.

5.24 REVIEW QUESTIONS

1. Justify with suitable example the need of a data warehousing team in an organization.
2. Describe the various task oriented stages of data warehousing team.
3. List down some of the main roles needed in an organization to implement data warehouse project.
4. What do you mean by data warehouse team checklist? Draw a tentative format of data warehouse team checklist.

5.25 FURTHER READINGS

R. Agrawal, T. Imielinski, A. Swami, "Mining Associations between Sets of Items in Massive Databases'', Proc. of the ACM SIGMOD Int'l Conference on Management of Data, Washington D.C., May 1993,207-216.

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo, "Fast Discovery of Association Rules", Advances in Knowledge Discovery and Data Mining, Chapter 12, AAAI/MIT Press, 1995.

R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", Proc. of the 20th Int'l Conference on Very Large Databases, Santiago,

R. Srikant, Q. Vu, R. Agrawal, "Mining Association Rules with Item Constraints", Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, California, August 1997.

R. Srikant, R. Agrawal: "Mining Generalized Association Rules", Proc. of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland,

Jong Soo Park, Ming-Syan Chen, Philip S. Yu, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules", IEEE Transactions on Knowledge and Data Engineering, vol. 9, no. 5, pp. 813-825, Sept/Oct 1997.

Mohammed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, "New Algorithms for Fast Discovery of Association Rules", 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97), pp 283-286, Newport Beach, California, August, 1997. Mohammed Zaki, Mitsunori Ogihara, Srinivasan Parthasarathy, and Wei Li, "Parallel Data Mining for Association Rules on Shared-memory Multi- processors", Supercomputing'96, Pittsburg, PA, Nov 17-22, 1996.

Mohammed Zaki, Srinivasan Parthasarathy, Wei Li, "A Localized Algorithm for Parallel Association Mining", 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp 321-330, Newport, Rhode Island,

A. Amir, R. Feldman, and R. Kashi, "A New and Versatile Method for Association Generation", Principles of Data Mining and Knowledge Discovery, First European Symposium, PKDD'97, Komorowski and Zytkow (eds.), Springer LNAI 1263, pp. 221-231, Trondheim, Norway, 1997.

Charu Aggarwal and Philip Yu, "Mining Large Itemsets for Association Rules", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998.

R. Ng, L. V. S. Lakshmanan, J. Han and A. Pang, “Exploratory Mining and Pruning Optimizations of Constrained Associations Rules”, Proc. of 1998 ACM-SIGMOD Conf. on Management of Data, Seattle, Washington, June 1998.

Sergey Brin, Rajeev Motwani, Craig Silverstein, “Beyond Market Baskets: Generalizing Association Rules to Correlations”, Proceedings of 1997 ACM SIGMOD, Montreal, Canada, June 1997.

Sergey Brin, Rajeev Motwani, Dick Tsur, Jeffrey Ullman, “Dynamic Itemset Counting and Implication Rules for Market Basket Data”, Proceedings of 1997 ACM SIGMOD, Montreal, Canada, June 1997.

Dick Tsur, Jeffrey Ullman, Chris Clifton, Serge Abiteboul, Rajeev Motwani, Svetlozar Nestorov, Arnie Rosenthal, “Query Flocks: a Generalization of Association-Rule Mining”, Proceedings of 1998 ACM SIGMOD, Seattle,

K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, “Computing Optimized Rectilinear Regions for Association Rules”, Proceedings of the Third Conference on Knowledge Discovery and Data Mining (KDD’97), pages 96-103, Los Angels, August 1997

K. Wang, W. Tay, B. Liu, “Interestingness-based Interval Merger for Numeric Association Rules” , Proc. International Conference on Knowledge Discovery and Data Mining (KDD-98), August 1998, New York City.

K. Alsabti, S. Ranka and V. Singh. “A One-Pass Algorithm for Accurately Estimating Quantiles for Disk-Resident Data”, In Proc. of VLDB’97

R. Agrawal and J. C. Shafer, “Parallel Mining of Association Rules: Design, Implementation and Experience”, IEEE Transactions on Knowledge Data and Engg., 8(6):962-969, December 1996.

Eui-Hong (Sam) Han, George Karypis and Vipin Kumar, “Scalable Parallel Data Mining for Association Rules”, Proc. of 1997 ACM-SIGMOD International Conference on Management of Data, May 1997.

T. Shintani and M. Kitsuregawa, “Hash Based Parallel Algorithm for Mining Association Rules”, Proceedings of IEEE Fourth International Conference on Parallel and Distributed Information Systems, pp.19-30, 1996.

H. Mannila and H. Toivonen, “Multiple uses of frequent sets and condensed representations”, Proc. Second International Conference on Knowledge Discovery and Data Mining (KDD’96), 189-194, Portland, Oregon, August 1996.

H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hdtvnen, and H. Mannila, “Pruning and grouping discovered association rules”, In MLnet Workshop on Statistics, Machine Learning, and Discovery in Databases, pp. 47-52, Heraklion, Crete, Greece, April 1995.

Jose Borges and Mark Levene, “Mining Association Rules in Hypertext Databases”, Proc. International Conference on Knowledge Discovery and Data Mining (KDD-98), August 1998, New York City.

Mohammed J. Zaki and Mitsunori Ogihara, “Theoretical Foundations of Association Rules”, 3rd SIGMOD’98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), Seattle, WA, June 1998.

Data Mining

UNIT

6

FUNDAMENTALS OF DATA MINING

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Define data, information and knowledge.
- Explain Data Mining and its significance.
- Describe historical revolution of Data Mining.
- Know about different Data Mining Techniques.
- Different type of Data Bases.
- Know about Applications of Data Mining.

U N I T S T R U C T U R E

6.1 INTRODUCTION

Over the past decade or so, businesses have accumulated huge amounts of data in large databases. These stockpiles mainly contain customer data, but the data's hidden value—the potential to predict business trends and customer behavior—has largely gone untapped.

To convert this potential value into strategic business information, many companies are turning to data mining, a growing technology based on a new generation of hardware and software. Data mining combines techniques including statistical analysis, visualization, decision trees, and neural networks to explore large amounts of data and discover relationships and patterns that shed light on business problems. In turn, companies can use these findings for more profitable, proactive decision making and competitive advantage. Although data mining tools have been around for many years, data mining became feasible in business only after advances in hardware and software technology came about.

Hardware advances—reduced storage costs and increased processor speed—paved the way for data mining's large-scale, intensive analyses. Inexpensive storage also encouraged businesses to collect data at a high level of detail, consolidated into records at the customer level.

Software advances continued data mining's evolution. With the advent of the data warehouse, companies could successfully analyze their massive databases as a coherent, standardized whole. To exploit these vast stores of data in the data warehouse, new exploratory and modeling tools—including data visualization and neural networks—were developed. Finally, data mining incorporated these tools into a systematic, iterative process.

Although data mining is a relatively new term, the technology is not. Companies have used powerful computers to sift through volumes of supermarket scanner data and analyze market research reports for years. However, continuous innovations in computer processing power, disk storage, and statistical software are dramatically increasing the accuracy of analysis while driving down the cost. For example, one Midwest grocery chain used the data mining capacity of Oracle software to analyze local buying patterns. They discovered that when men bought diapers on Thursdays and Saturdays, they also tended to buy beer. Further analysis showed that these shoppers typically did their weekly grocery shopping on Saturdays. On Thursdays, however, they only bought a few items. The retailer concluded that they purchased the beer to have it available for the upcoming weekend. The grocery chain could use this newly discovered information in various ways to increase revenue. For example, they could move the beer display closer to the diaper display. And, they could make sure beer and diapers were sold at full price on Thursdays.

Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

6.2 DATA, INFORMATION AND KNOWLEDGE

Data

Data are any facts, numbers, or text that can be processed by a computer. Today, organizations are accumulating vast and growing amounts of data in different formats and different databases. This includes:

- operational or transactional data such as, sales, cost, inventory, payroll, and accounting
- nonoperational data, such as industry sales, forecast data, and macro economic data
- meta data - data about the data itself, such as logical database design or data dictionary definitions

Information

The patterns, associations, or relationships among all this *data* can provide *information*. For example, analysis of retail point of sale transaction data can yield information on which products are selling and when.

Knowledge

Information can be converted into *knowledge* about historical patterns and future trends. For example, summary information on retail supermarket sales can be analyzed in light of promotional efforts to provide knowledge of consumer buying behavior. Thus, a manufacturer or retailer could determine which items are most susceptible to promotional efforts.

Data Warehouses

Dramatic advances in data capture, processing power, data transmission, and storage capabilities are enabling organizations to integrate their various databases into *data warehouses*. Data

warehousing is defined as a process of centralized data management and retrieval. Data warehousing, like data mining, is a relatively new term although the concept itself has been around for years. Data warehousing represents an ideal vision of maintaining a central repository of all organizational data. Centralization of data is needed to maximize user access and analysis. Dramatic technological advances are making this vision a reality for many companies. And, equally dramatic advances in data analysis software are allowing users to access this data freely. The data analysis software is what supports data mining.

Student Activity 1

1. Define the term Data, Information and Knowledge with suitable examples.

6.3 DATA MINING

Data Mining can be defined as the process of selecting, exploring, and modeling large amounts of data to uncover previously unknown patterns for a business advantage. As a sophisticated decision support tool, data mining is a natural outgrowth of a business' investment in data warehousing. The data warehouse provides a stable, easily accessible repository of information to support dynamic business intelligence applications.

What can data mining do?

Data mining is primarily used today by companies with a strong consumer focus - retail, financial, communication, and marketing organizations. It enables these companies to determine relationships among "internal" factors such as price, product positioning, or staff skills, and "external" factors such as economic indicators, competition, and customer demographics. And, it enables them to determine the impact on sales, customer satisfaction, and corporate profits. Finally, it enables them to "drill down" into summary information to view detail transactional data.

With data mining, a retailer could use point-of-sale records of customer purchases to send targeted promotions based on an individual's purchase history. By mining demographic data from comment or warranty cards, the retailer could develop products and promotions to appeal to specific customer segments.

For example, Blockbuster Entertainment mines its video rental history database to recommend rentals to individual customers. American Express can suggest products to its cardholders based on analysis of their monthly expenditures.

What is data mining good for?

Data mining software allows users to analyze large databases to solve business decision problems. Data mining is, in some ways, an extension of statistics, with a few artificial intelligence and machine learning twists thrown in. Like statistics, data mining is not a business solution, it is just a technology. For example, consider a catalog retailer who needs to decide who should receive information about a new product. The information operated on by the data mining process is contained in a historical database of previous interactions with customers and the features associated with the customers, such as age, zip code, their responses. The data mining software would use this historical information to build a model of customer behavior that could be used to predict which customers would be likely to respond to the new product. By using this information a marketing manager can select only the customers who are most likely to respond. The operational business software can then feed the results of the decision to the appropriate touch point systems (call centers, direct mail, web servers, e-mail systems, etc.) so that the right customers receive the right offers.

6.4 THE HISTORICAL EVOLUTION OF DATA MINING

Data mining techniques are the result of a long process of research and product development. This evolution began when business data was first stored on computers, continued with improvements in data access, and more recently, generated technologies that allow users to navigate through their data in real time. Data mining takes this evolutionary process beyond retrospective data access and navigation to prospective and proactive information delivery. Data mining is ready for application in the business community because it is supported by three

technologies that are now sufficiently mature:

- Massive data collection
- Powerful multiprocessor computers
- Data mining algorithms

Commercial databases are growing at unprecedented rates. A recent META Group survey of data warehouse projects found that 19% of respondents are beyond the 50 gigabyte level, while 59% expect to be there by second quarter of 1996.1 In some industries, such as retail, these numbers can be much larger. The accompanying need for improved computational engines can now be met in a cost-effective manner with parallel multiprocessor computer technology. Data mining algorithms embody techniques that have existed for at least 10 years, but have only recently been implemented as mature, reliable, understandable tools that consistently outperform older statistical methods.

In the evolution from business data to business information, each new step has built upon the previous one. For example, dynamic data access is critical for drill-through in data navigation applications, and the ability to store large databases is critical to data mining. From the user's point of view, the four steps listed in Table 6.1 were revolutionary because they allowed new business questions to be answered accurately and quickly.

Table 6.1: Steps in the Evolution of Data Mining

Evolutionary Step	Business Question	Enabling Technologies	Product Providers	Characteristics
Data Collection (1960s)	"What was my total revenue in the last five years?"	Computers, tapes, disks	IBM, CDC	Retrospective, static data delivery
Data Access (1980s)	"What were unit sales in New England last March?"	Relational databases (RDBMS), Structured Query Language (SQL), ODBC	Oracle, Sybase, Informix, IBM, Microsoft	Retrospective, dynamic data delivery at record level
Data Warehousing & Decision Support (1990s)	"What were unit sales in New England last March? Drill down to Boston."	On-line analytic processing (OLAP), multidimensional databases, data warehouses	Pilot, Comshare, Arbor, Cognos, Microstrategy	Retrospective, dynamic data delivery at multiple levels
Data Mining (Emerging Today)	"What's likely to happen to Boston unit sales next month? Why?"	Advanced algorithms, multiprocessor computers, massive databases	Pilot, Lockheed, IBM, SGI, numerous startups (nascent industry)	Prospective, proactive information delivery

The core components of data mining technology have been under development for decades, in research areas such as statistics, artificial intelligence, and machine learning. Today, the maturity of these techniques, coupled with high-performance relational database engines and broad data integration efforts, make these technologies practical for current data warehouse environments.

6.5 HOW DATA MINING WORKS?

How exactly is data mining able to tell you important things that you didn't know or what is going to happen next? The technique that is used to perform these feats in data mining is called modeling. Modeling is simply the act of building a model in one situation where you know the

answer and then applying it to another situation that you don't. For instance, if you were looking for a sunken Spanish galleon on the high seas the first thing you might do is to research the times when Spanish treasure had been found by others in the past. You might note that these ships often tend to be found off the coast of Bermuda and that there are certain characteristics to the ocean currents, and certain routes that have likely been taken by the ship's captains in that era. You note these similarities and build a model that includes the characteristics that are common to the locations of these sunken treasures. With these models in hand you sail off looking for treasure where your model indicates it most likely might be given a similar situation in the past. Hopefully, if you've got a good model, you find your treasure.

This act of model building is thus something that people have been doing for a long time, certainly before the advent of computers or data mining technology. What happens on computers, however, is not much different than the way people build models. Computers are loaded up with lots of information about a variety of situations where an answer is known and then the data mining software on the computer must run through that data and distill the characteristics of the data that should go into the model. Once the model is built it can then be used in similar situations where you don't know the answer. For example, say that you are the director of marketing for a telecommunications company and you'd like to acquire some new long distance phone customers. You could just randomly go out and mail coupons to the general population - just as you could randomly sail the seas looking for sunken treasure. In neither case would you achieve the results you desired and of course you have the opportunity to do much better than random - you could use your business experience stored in your database to build a model.

As the marketing director you have access to a lot of information about all of your customers: their age, sex, credit history and long distance calling usage. The good news is that you also have a lot of information about your prospective customers: their age, sex, credit history etc. Your problem is that you don't know the long distance calling usage of these prospects (since they are most likely now customers of your competition). You'd like to concentrate on those prospects who have large amounts of long distance usage. You can accomplish this by building a model. Table 6.2 illustrates the data used for building a model for new customer prospecting in a data warehouse.

Table 6.2: Data Mining for Prospecting

	Customers	Prospects
General information (e.g. demographic data)	Known	Known
Proprietary information (e.g. customer transactions)	Known	Target

The goal in prospecting is to make some calculated guesses about the information in the lower right hand quadrant based on the model that we build going from Customer General Information to Customer Proprietary Information. For instance, a simple model for a telecommunications company might be:

98% of my customers who make more than $60,000/year spend more than $80/month on long distance

This model could then be applied to the prospect data to try to tell something about the proprietary information that this telecommunications company does not currently have access to. With this model in hand new customers can be selectively targeted.

Test marketing is an excellent source of data for this kind of modeling. Mining the results of a test market representing a broad but relatively small sample of prospects can provide a foundation for identifying good prospects in the overall market.

If someone told you that he had a model that could predict customer usage how would you know if he really had a good model? The first thing you might try would be to ask him to apply his model to your customer base - where you already knew the answer. With data mining, the best way to accomplish this is by setting aside some of your data in a vault to isolate it from the mining

process. Once the mining is complete, the results can be tested against the data held in the vault to confirm the model's validity. If the model works, its observations should hold for the vaulted data.

6.6 KINDS OF DATA

Data mining is performed on a data repository. A data repository is simply a large storage of electronic data. There are a number of different types of data storage. Data mining should be equally applicable to all every type of data repository, in principle at least. The techniques of data mining, however, do depend on the type of data repository being mined. Therefore, a summary of different types of data storage is given below.

Relational Databases

A database is a repository of data. A database management system consists of a database and a set of procedures/programs through which a user may access and manage the database. There are a variety of databases - Relational, Network, Hierarchical etc.

A relational database is, by and large, the most popular database system. It stores its data in the form of collection of tables having unique names. Each table consists of attributes (columns or fields) and tuples (rows or records). Each record or tuple in a relational table represents information regarding a single object identified by a key. The data model that is used to model relational database is ER (Entity-Relationship). The readers are expected to have a fair knowledge of relational databases.

Consider the following relational database example. It will be used in subsequent chapters for illustration purposes.

The database of a trading company, say Haryana Trading Company or HTC, has the following tables (relations):

Table 6.3

Relation	Attribute	Attribute description
Cust	CustId	Unique customer identification number
	CustName	Name of the customer
	CustAddress	Address of the customer
	CustAge	Age of the customer
	CustIncome	Annual income of the customer
Item	ItemId	Unique item identification number
	ItemName	Name of the item
	ItemBrand	Brand name of the item
	ItemMake	Make of the item
	ItemPrice	Price of this item
Emp	EmpId	Unique employee identification number
	EmpName	Employee name
	EmpSal	Employee salary
Bra	BraId	Unique branch identification number
	BraName	Branch name
	BraAddress	Branch address
Pur	PurId	Unique purchase identification number
	CustId	Id of the customer making this purchase

Contd...

	EmpId	Id of the employee who made this sale
	PurDate	Date of this purchase
	PurTime	Time of the purchase
	PurPayMode	Mode of payment of this purchase
	PurAmt	Amount of this purchase
Sol	PurId	Unique purchase identification number
	ItemId	Id of the item sold in this purchase
	SolQty	Quantity of this item sold in this purchase
EmpBra	EmpId	Unique employee identification number
	BraId	Branch id where this employee works

The names of fields of each table are self-explanatory. An instance of this database is presented below:

Item					
ItemId	**ItemName**	**ItemBrand**	**ItemMake**		**ItemPrice**
1102	Cellphone	Samsung	Japan		8000
1130	Computer	Compaq	USA		48000

Cust			
CustId	**CustName**	**CustAddress**	**CustAge**	**CustIncome**
0001	Mohan Singh	23, P.B. Colony, Hisar	45	40000
0132	Ram Prasad	109, GT Road, Karnal	36	75000

Emp	
EmpId	**EmpName**	**EmpSalary**
9999	Jivan Yadav	5000
9998	Renu Sharma	4500
9997	Amit Rahi	6000
9996	Amita Malik	5500

Bra		
BraId	**BraName**	**BraAddress**
0001	HP Electronics	78, AB Complex, Hisar
0002	Raghav Traders	2, West End, Chandigarh
0003	Tulip Traders	34, R & B Street, Karnal

Pur						
PurId	**CustId**	**EmpId**	**PurDate**	**PurTime**	**PurPayMode**	**PurAmt**
0001	0001	9997	12/11/2002	1340	Cash	20000
0002	0132	9996	15/11/2002	1015	Credit Card	5000
0003	0001	9996	20/11/2002	1500	Cash	4800

Sol			EmpBra	
PurId	**ItemId**	**SolQty**	**EmpId**	**BraId**
0001	0001	4	9999	0001
0002	0002	2	9998	0002
0002	0004	1	9997	0002
0003	0001	1	9996	0003

Data Warehouses

Suppose that HTC is a successful international company, with branches around the world and that each branch has its own set of databases. It is required to analyze the company's sales per item type per branch for the current year. This is a difficult task, particularly since the relevant data are spread out over several databases, physically located at numerous sites. If HTC had a data warehouse, this task would be easy.

As stated earlier, a data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and which usually resides at a single site. Data warehouses are constructed via a process of data cleaning, data transformation, data integration, data loading, and periodic data refreshing.

In order to facilitate decision making, the data in a data warehouse are organized around major subjects, such as customer, item, supplier, and activity. The data are stored to provide information from a historical perspective (such as from the past 5-10 years) and are typically summarized. For example, rather than storing the details of each sales transaction, the data warehouse may store a summary of the transactions per item type for each store or, summarized to a higher level, for each sales region.

A data warehouse is usually modeled by a multidimensional database structure, where each dimension corresponds to an attribute or a set of attributes in the schema, and each cell stores the value of some aggregate measure, such as count or sales amount. The actual physical structure of a data warehouse may be a relational data store or a multidimensional data cube. It provides a multidimensional view of data and allows the pre-computation and fast accessing of summarized data.

A data mart is a department subset of a data warehouse. It focuses on selected subjects, and thus its scope is department-wide.

By providing multidimensional data views and the pre-computation of summarized data, data warehouse systems are well suited for On-Line Analytical Processing, or OLAP. OLAP operations make use of background knowledge regarding the domain of the data being studied in order to allow the presentation of data at different levels of abstraction. Such operations accommodate different user viewpoints. Examples of OLAP operations include drill-down and roll-up, which allow the user to view the data at differing degrees of summarization. For instance, one may drill down on sales data summarized by quarter to see the data summarized by month. Similarly, one may roll up on sales data summarized by city to view the data summarized by country.

Though data warehouse tools help support data analysis, additional tools for data mining are required to allow more in-depth and automated analysis.

Transactional Databases

A database that contains information about transactions either in a flat-file format or in tables. The essential property of a transactional database is that it keeps on changing frequently. New records are entered for each new transaction. It is less stable in terms of its contents unlike a data warehouse.

In our ongoing HTC example, relation Pur is a table of transactional database.

Pur						
PurId	**CustId**	**EmpId**	**PurDate**	**PurTime**	**PurPayMode**	**PurAmt**
0001	0001	9997	12/11/2002	1340	Cash	20000
0002	0132	9996	15/11/2002	1015	Credit Card	5000
0003	0001	9996	20/11/2002	1500	Cash	4800

Object Oriented Databases

These databases are based on object-oriented programming paradigm. The unit of storage is object instances rather than simple data item. Every entity is modeled as an object having:

- ***Properties:*** A set of variables describing various attributes of the object.
- ***Methods:*** A set of functions to manipulate this object.
- ***Messages:*** A set of signals to establish communication between objects.

The database stores instances of these objects. In our HTC example, there are many objects - Employee object, Customer object, Item object etc.

Object-Relational Databases

This type of database employs both the technologies offered by object-oriented approach and relational approach. The data is modeled in object manner and the objects are stored in a relational manner. It, thus provides advantages of both the technologies.

Spatial Databases

These databases contain space-related information mostly in the form of maps. For instance information about a country, its states and districts etc.; multidimensional VLSI chip design etc.; two dimensional satellite images etc.; geographical information etc.

These databases are of great use in applications ranging from forestry and ecology-planning, city-planning, tourist-information systems, navigation and dispatch systems etc.

In such databases, data mining may uncover patterns that describe the characteristics of houses in a particular locality such as market place, parks etc. Spatial data-cubes may be constructed to on which OAP operations (such as drill-down and roll-up) may be performed to uncover interesting patterns.

Temporal Databases

Temporal databases contain information in which at least one attribute is related to time. For instance information on sales of a particular commodity over a period of time in a year.

One variation of the temporal database is known as time-series database, in which information regarding some quantity and a sequence of its values/observation over a period of time described in some interval. For example, data collected regarding stock-exchange per day over last ten years.

Data mining can be performed on these databases to uncover evolution of a particular object through time as well as the trend of changes in the object. The stock-exchange database may be mined to uncover trends which might help a user to make decisions regarding when to buy, when to sell, how much to buy etc.

Text Databases

Text databases are databases that store information in textual form without any coding-scheme implementation. The information stored in form of memo, paragraphs, reports etc. rather than a

simple keyword. They may be highly unstructured such as web pages or semi-structured using HTML/XML documents. Highly structured and regular information may be stored in the form of relational databases.

Data mining on this type of database may be integrated with data retrieval tools to uncover general description of object classes, keyword and content associations etc.

Multimedia Databases

These databases store images, audio information and video data. They are heavily used in applications like voice-mail systems, video-on-demand systems, speech-based user interfaces etc. These databases are designed to store object of very large sizes usually of several giga-bytes. Since data retrieval from these databases need to be real-time, they are also referred to as continuous or stream databases.

In case of multi-media database data mining, storage and search techniques are required to be integrated with the standard data mining tools. Some approaches use construction of multi-media data cubes, similarity-based pattern matching etc.

Heterogeneous Databases

A heterogeneous database contains database objects coming from various different databases. Therefore, it is difficult to assimilate their individual semantics into the overall heterogeneous database. Information exchange also becomes difficult because exact transformation rules are required to be worked out.

Thus, a database system that stores some of the information in unstructured textual form and other information in tabular form, constitutes a heterogeneous database.

Legacy Databases

A database that has existed for a long time and resembles no contemporary database systems is known as a legacy database. To use such a database generally two options are available. First option is to continue working with the old database system with whatever limitations it has. Another option is to migrate this legacy database to a modern database system. In either case a lot of efforts are needed.

The World Wide Web

This is another repository that contains unimaginable amount of information scattered all over the world. The information objects are linked together and can be accessed via the links provided. These information exist in form of web-pages. Web-pages, though carry tremendous amount of information, can be highly unstructured. Each web-site is free to present and store the information in their own individual format and manner.

Data mining the WWW may provide insight into the path-traversal patterns. This information may be further used in improving the web-site design and in making better marketing decisions

6.7 AN ARCHITECTURE FOR DATA MINING

To best apply these advanced techniques, they must be fully integrated with a data warehouse as well as flexible interactive business analysis tools. Many data mining tools currently operate outside of the warehouse, requiring extra steps for extracting, importing, and analyzing the data. Furthermore, when new insights require operational implementation, integration with the warehouse simplifies the application of results from data mining. The resulting analytic data warehouse can be applied to improve business processes throughout the organization, in areas such as promotional campaign management, fraud detection, new product rollout, and so on. Figure 6.1 illustrates an architecture for advanced analysis in a large data warehouse.

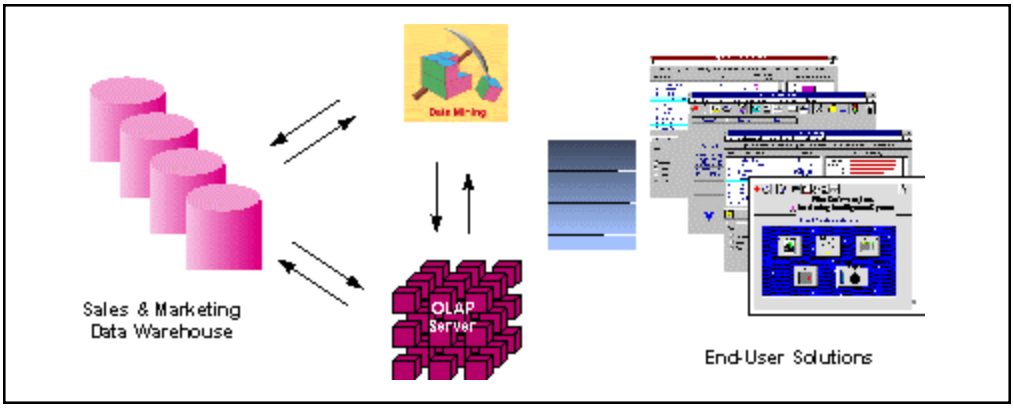


Figure 6.1: Integrated data Mining Architecture

The ideal starting point is a data warehouse containing a combination of internal data tracking all customer contact coupled with external market data about competitor activity. Background information on potential customers also provides an excellent basis for prospecting. This warehouse can be implemented in a variety of relational database systems: Sybase, Oracle, Redbrick, and so on, and should be optimized for flexible and fast data access.

An OLAP (On-Line Analytical Processing) server enables a more sophisticated end-user business model to be applied when navigating the data warehouse. The multidimensional structures allow the user to analyze the data as they want to view their business – summarizing by product line, region, and other key perspectives of their business. The Data Mining Server must be integrated with the data warehouse and the OLAP server to embed ROI-focused business analysis directly into this infrastructure. An advanced, process-centric metadata template defines the data mining objectives for specific business issues like campaign management, prospecting, and promotion optimization. Integration with the data warehouse enables operational decisions to be directly implemented and tracked. As the warehouse grows with new decisions and results, the organization can continually mine the best practices and apply them to future decisions.

This design represents a fundamental shift from conventional decision support systems. Rather than simply delivering data to the end user through query and reporting software, the Advanced Analysis Server applies users' business models directly to the warehouse and returns a proactive analysis of the most relevant information. These results enhance the metadata in the OLAP Server by providing a dynamic metadata layer that represents a distilled view of the data. Reporting, visualization, and other analysis tools can then be applied to plan future actions and confirm the impact of those plans.

6.8 APPLICATIONS OF DATA MINING

Risk Analysis

Insurance companies and banks use data mining for risk analysis. An insurance company searches in its own insurants and claims databases for relationships between personal characteristics and claim behavior. The company is especially interested in the characteristics of insurants with a highly deviating claim behavior. With data mining, these so-called risk-profiles can be discovered. The company can use this information to adapt its premium policy.

Direct Marketing

Data mining can also be used to discover the relationship between one's personal characteristics, e.g. age, gender, hometown, and the probability that one will respond to a mailing. Such relationships can be used to select those customers from the mailing database that have the highest probability of responding to a mailing. This allows the company to mail its prospects selectively, thus maximizing the response. In more detail, this works as shown on next page.

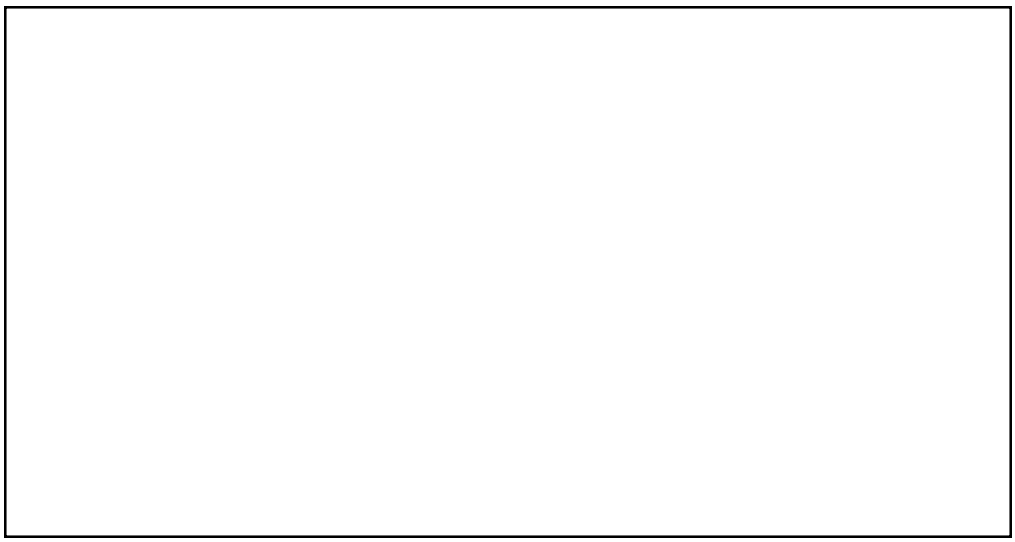


Figure 6.2

Company X sends a mailing (1) to a number of prospects. The response (2) is e.g. 2%. The response is analyzed using data mining techniques (3), discovering differences between the customers that did respond, and those that did not respond. The result consists of database subgroups that have a significantly higher response probability (4), e.g. of all young couples with double incomes, 24% replied to the last mailing. The groups with the highest response-probability are selected as targets for the next mailing (5). Data mining thus increases the response considerably.

Production Quality Control

Data Mining can also be used to determine those combinations of production factors that influence the quality of the end-product. This information allows the process engineers to explain why certain products fail the final test and to increase the quality of the production process.

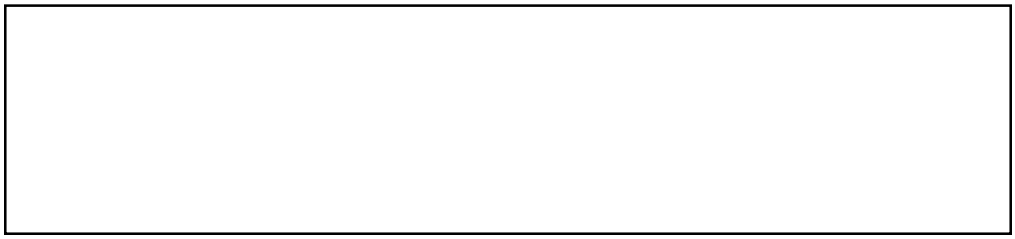


Figure 6.3

The above production process consists of steps A, B and C. During each step, machine settings and environmental factors influence the quality of the product. A quality test takes place at the end of the process. With data mining, the company can discover which combinations of machine adjustments and other factors result in faulty products. Using this information, the number of failing products can be decreased.

6.9 DATA MINING FUNCTIONS

Data mining methods may be classified by the function they perform or according to the class of application they can be used in. Some of the main techniques used in data mining are described in this section.

Classification

Data mine tools have to infer a model from the database, and in the case of supervised learning this requires the user to define one or more classes. The database contains one or more attributes that denote the class of a tuple and these are known as predicted attributes whereas the remaining

attributes are called predicting attributes. A combination of values for the predicted attributes defines a class.

When learning classification rules the system has to find the rules that predict the class from the predicting attributes so firstly the user has to define conditions for each class, the data mine system then constructs descriptions for the classes. Basically the system should given a case or tuple with certain known attribute values be able to predict what class this case belongs to.

Once classes are defined the system should infer rules that govern the classification therefore the system should be able to find the description of each class. The descriptions should only refer to the predicting attributes of the training set so that the positive examples should satisfy the description and none of the negative. A rule said to be correct if its description covers all the positive examples and none of the negative examples of a class.

A rule is generally presented as, if the left hand side (LHS) then the right hand side (RHS), so that in all instances where LHS is true then RHS is also true, are very probable. The categories of rules are:

- ***Exact rule:*** Permits no exceptions so each object of LHS must be an element of RHS
- ***Strong rule:*** Allows some exceptions, but the exceptions have a given limit
- ***Probablistic rule:*** Relates the conditional probability P(RHS|LHS) to the probability P(RHS)

Other types of rules are classification rules where LHS is a sufficient condition to classify objects as belonging to the concept referred to in the RHS.

Associations

Given a collection of items and a set of records, each of which contain some number of items from the given collection, an association function is an operation against this set of records which return affinities or patterns that exist among the collection of items. These patterns can be expressed by rules such as "72% of all the records that contain items A, B and C also contain items D and E." The specific percentage of occurrences (in this case 72) is called the confidence factor of the rule. Also, in this rule, A,B and C are said to be on an opposite side of the rule to D and E. Associations can involve any number of items on either side of the rule.

A typical application, identified by IBM, that can be built using an association function is Market Basket Analysis. This is where a retailer run an association operator over the point of sales transaction log, which contains among other information, transaction identifiers and product identifiers. The set of products identifiers listed under the same transaction identifier constitutes a record. The output of the association function is, in this case, a list of product affinities. Thus, by invoking an association function, the market basket analysis application can determine affinities such as "20% of the time that a specific brand toaster is sold, customers also buy a set of kitchen gloves and matching cover sets."

Another example of the use of associations is the analysis of the claim forms submitted by patients to a medical insurance company. Every claim form contains a set of medical procedures that were performed on a given patient during one visit. By defining the set of items to be the collection of all medical procedures that can be performed on a patient and the records to correspond to each claim form, the application can find, using the association function, relationships among medical procedures that are often performed together.

Sequential/Temporal patterns

Sequential/temporal pattern functions analyse a collection of records over a period of time for example to identify trends. Where the identity of a customer who made a purchase is known an analysis can be made of the collection of related records of the same structure (i.e. consisting of a number of items drawn from a given collection of items). The records are related by the identity of the customer who did the repeated purchases. Such a situation is typical of a direct mail application where for example a catalogue merchant has the information, for each customer, of the sets of products that the customer buys in every purchase order. A sequential pattern function will analyse such collections of related records and will detect frequently occurring patterns of

products bought over time. A sequential pattern operator could also be used to discover for example the set of purchases that frequently precedes the purchase of a microwave oven.

Sequential pattern mining functions are quite powerful and can be used to detect the set of customers associated with some frequent buying patterns. Use of these functions on for example a set of insurance claims can lead to the identification of frequently occurring sequences of medical procedures applied to patients which can help identify good medical practices as well as to potentially detect some medical insurance fraud.

Clustering/Segmentation

Clustering and segmentation are the processes of creating a partition so that all the members of each set of the partition are similar according to some metric. A cluster is a set of objects grouped together because of their similarity or proximity. Objects are often decomposed into an exhaustive and/or mutually exclusive set of clusters.

Clustering according to similarity is a very powerful technique, the key to it being to translate some intuitive measure of similarity into a quantitative measure. When learning is unsupervised then the system has to discover its own classes i.e. the system clusters the data in the database. The system has to discover subsets of related objects in the training set and then it has to find descriptions that describe each of these subsets.

There are a number of approaches for forming clusters. One approach is to form rules which dictate membership in the same group based on the level of similarity between members. Another approach is to build set functions that measure some property of partitions as functions of some parameter of the partition.

IBM - Market Basket Analysis example

IBM have used segmentation techniques in their Market Basket Analysis on POS transactions where they separate a set of untagged input records into reasonable groups according to product revenue by market basket i.e. the market baskets were segmented based on the number and type of products in the individual baskets.

Each segment reports total revenue and number of baskets and using a neural network 275,000 transaction records were divided into 16 segments. The following types of analysis were also available, revenue by segment, baskets by segment, average revenue by segment etc.

6.10 DATA MINING: ISSUES

Issues Presented

One of the key issues raised by data mining technology is not a business or technological one, but a social one. It is the issue of individual privacy. Data mining makes it possible to analyze routine business transactions and glean a significant amount of information about individuals buying habits and preferences.

Another issue is that of data integrity. Clearly, data analysis can only be as good as the data that is being analyzed. A key implementation challenge is integrating conflicting or redundant data from different sources. For example, a bank may maintain credit cards accounts on several different databases. The addresses (or even the names) of a single cardholder may be different in each. Software must translate data from one system to another and select the address most recently entered.

A hotly debated technical issue is whether it is better to set up a relational database structure or a multidimensional one. In a relational structure, data is stored in tables, permitting ad hoc queries. In a multidimensional structure, on the other hand, sets of cubes are arranged in arrays, with subsets created according to category. While multidimensional structures facilitate multidimensional data mining, relational structures thus far have performed better in client/server environments. And, with the explosion of the Internet, the world is becoming one big client/server environment.

Finally, there is the issue of cost. While system hardware costs have dropped dramatically within

the past five years, data mining and data warehousing tend to be self-reinforcing. The more powerful the data mining queries, the greater the utility of the information being gleaned from the data, and the greater the pressure to increase the amount of data being collected and maintained, which increases the pressure for faster, more powerful data mining queries. This increases pressure for larger, faster systems, which are more expensive.

Student Activity 2

1. Write a short note on the futuristic development in the field of data mining.

6.11 SUMMARY

- Data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.
- Data are any facts, numbers, or text that can be processed by a computer
- The patterns, associations, or relationships among all this *data* can provide *information*.
- Information can be converted into *knowledge* about historical patterns and future trends.
- Data warehousing is defined as a process of centralized data management and retrieval
- Data warehousing represents an ideal vision of maintaining a central repository of all organizational data.
- Data mining software allows users to analyze large databases to solve business decision problems
- Test marketing is an excellent source of data for this kind of modeling. Mining the results of a test market representing a broad but relatively small sample of prospects can provide a foundation for identifying good prospects in the overall market .
- An OLAP (On-Line Analytical Processing) server enables a more sophisticated end-user business model to be applied when navigating the data warehouse.
- Risk analysis, Direct Marketing, Production Quality Control are the applications of Data mining.
- A rule is generally presented as, if the left hand side (LHS) then the right hand side (RHS), so that in all instances where LHS is true then RHS is also true, are very probable. The categories of rules are:
- Exact rule - permits no exceptions so each object of LHS must be an element of RHS
- Strong rule - allows some exceptions, but the exceptions have a given limit
- Probabilistic rule - relates the conditional probability P(RHS|LHS) to the probability P(RHS)
- an association function is an operation against this set of records which return affinities or patterns that exist among the collection of items.
- The specific percentage of occurrences (in this case 72) is called the confidence factor of the rule.

6.12 KEYWORDS

Data: Any facts, numbers, or text that can be processed by a computer.

Data Mining: It can be defined as the process of selecting, exploring, and modeling large amounts of data to uncover previously unknown patterns for a business advantage.

Clustering and segmentation: These are the processes of creating a partition so that all the members of each set of the partition are similar according to some metric.

6.13 REVIEW QUESTIONS

1. Explain how the evolution of database technology led to data mining.
2. Explain various steps involved in data mining.
3. Describe the architecture of data mining. Also mention what data mining can do.
4. List down the various data mining functionalities.
5. Describe three challenges to data mining regarding data mining methodology and user interaction issues.
6. What is the difference between:
   a. Discrimination and Classification
   b. Characterization and Clustering
   c. Classification and Prediction
7. List down some of the applications of data mining.
8. How data mining is different from data warehousing? Explain its working.

6.14 FURTHER READINGS

Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurasamy, "Advances in Knowledge Discovery and Data Mining", AAAI Press/ The MIT Press, 1996.

J. Ross Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1993.

Michael Berry and Gordon Linoff, "Data Mining Techniques (For Marketing, Sales, and Customer Support), John Wiley & Sons, 1997.

Sholom M. Weiss and Nitin Indurkhya, "Predictive Data Mining: A Practical Guide", Morgan Kaufmann Publishers, 1998.

Alex Freitas and Simon Lavington, "Mining Very Large Databases with Parallel Processing", Kluwer Academic Publishers, 1998.

A. K. Jain and R. C. Dubes, "Algorithms for Clustering Data", Prentice Hall, 1988.

V. Cherkassky and F. Mulier, "Learning From Data", John Wiley & Sons, 1998.

UNIT

7

DATA WAREHOUSE OLAP AND DATA MINING

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Define Data Warehouse
- Know about different stages of Data Warehousing
- Know about Metadata and Optional Components
- Know about OLAP

U N I T S T R U C T U R E

7.1 INTRODUCTION

A **data warehouse** is a computer system designed for archiving and analyzing an organisation's historical data, such as sales, salaries, or other information from day-to-day operations. Normally, an organisation copies information from its operational systems (such as sales and human resources) to the data warehouse on a regular schedule, such as every night or every weekend; after that, management can perform complex queries and analysis (such as data mining) on the information without slowing down the operational systems.

A data warehouse is the main repository of the organization's historical data, its *corporate memory*. For example, an organization would use the information that's stored in its data warehouse to find out what day of the week they sold the most widgets in May 1992, or how employee sick

leave the week before Christmas differed between California and Quebec from 2001-2005. In other words, the data warehouse contains the raw material for management's decision support system.

While operational systems are optimized for simplicity and speed of modification (online transaction processing, or *OLTP*) through heavy use of database normalization and an entity-relationship model, the data warehouse is optimized for reporting and analysis (online analytical processing, or *OLAP*). Frequently data in Data Warehouses is heavily denormalised, summarised and/or stored in a dimension-based model but this is not always required to achieve acceptable query response times.

More formally, Bill Inmon (one of the earliest and most influential practitioners) defined a data warehouse as follows:

- **Subject-oriented**, meaning that the data in the database is organized so that all the data elements relating to the same real-world event or object are linked together;
- **Time-variant**, meaning that the changes to the data in the database are tracked and recorded so that reports can be produced showing changes over time;
- **Non-volatile**, meaning that data in the database is never over-written or deleted, but retained for future reporting; and,
- **Integrated**, meaning that the database contains data from most or all of an organization's operational applications, and that this data is made consistent.

7.2 HISTORY OF DATA WAREHOUSING

Data Warehouses became a distinct type of computer database during the late 1980s and early 1990s. They were developed to meet a growing demand for management information and analysis that could not be met by operational systems. Operational systems were unable to meet this need for a range of reasons:

- The processing load of reporting reduced the response time of the operational systems,
- The database designs of operational systems were not optimised for information analysis and reporting,
- Most organizations had more than one operational system, so company-wide reporting could not be supported from a single system, and
- Development of reports in operational systems often required writing specific computer programs which was slow and expensive.

As a result, separate computer databases began to be built that were specifically designed to support management information and analysis purposes. These data warehouses were able to bring in data from a range of different data sources, such as mainframe computers, minicomputers, as well as personal computers and office automation software such as spreadsheet, and integrate this information in a single place. This capability, coupled with user-friendly reporting tools and freedom from operational impacts, has led to a growth of this type of computer system.

As technology improved (lower cost for more performance) and user requirements increased (faster data load cycle times and more features), data warehouses have evolved through several fundamental stages:

- ***Offline Operational Databases****:* Data warehouses in this initial stage are developed by simply copying the database of an operational system to an off-line server where the processing load of reporting does not impact on the operational system's performance.
- ***Offline Data Warehouse****:* Data warehouses in this stage of evolution are updated on a regular time cycle (usually daily, weekly or monthly) from the operational systems and the data is stored in an integrated reporting-oriented data structure.
- **Real Time Data Warehouse**: Data warehouses at this stage are updated on a transaction or event basis, every time an operational system performs a transaction (e.g. an order or a delivery or a booking etc.)

- ***Integrated Data Warehouse:*** Data warehouses at this stage are used to generate activity or transactions that are passed back into the operational systems for use in the daily activity of the organization.

Student Activity 1

1. Write in details about data warehouse. How it is different from data base?

7.3 COMPONENTS OF A DATA WAREHOUSE

The primary components of the majority of data warehouses are shown in the attached diagram and described in more detail below:

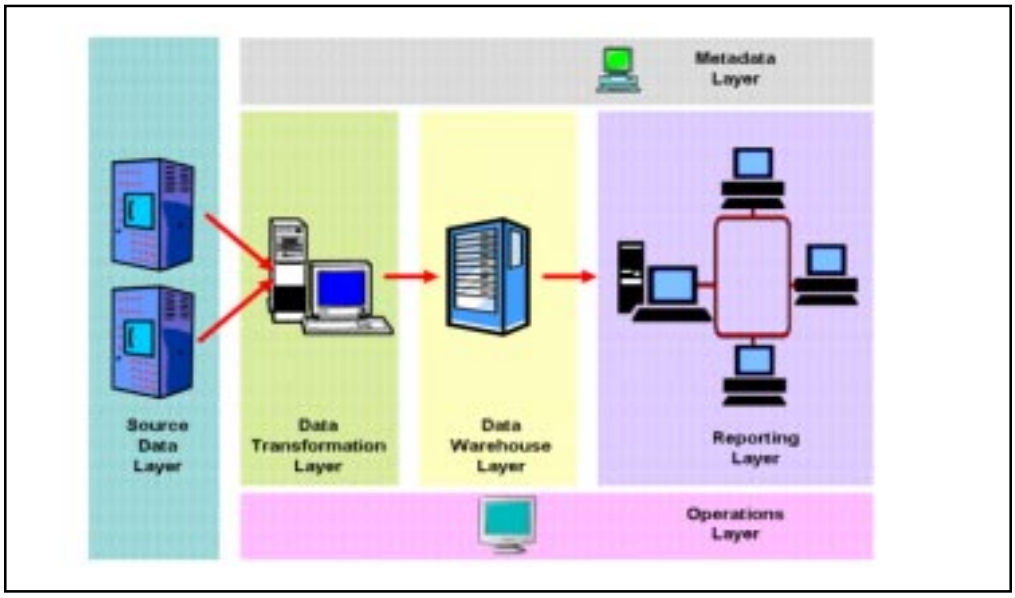


Figure 7.1

Data Sources

Data sources refers to any electronic repository of information that contains data of interest for management use or analytics. This definition covers mainframe databases (e.g. IBM DB2, ISAM, Adabas, Teradata, etc.), client-server databases (e.g. Teradata, IBM DB2, Oracle database, Informix, Microsoft SQL Server, etc.), PC databases (e.g. Microsoft Access, Alpha Five), spreadsheets (e.g. Microsoft Excel) and any other electronic store of data. Data needs to be passed from these systems to the data warehouse either on a transaction-by-transaction basis for real-time data warehouses or on a regular cycle (e.g.daily or weekly) for offline data warehouses.

Data Warehouse

The data warehouse is normally (but does not have to be) a relational database. It must be organized to hold information in a structure that best supports not only query and reporting, but also advanced analysis techniques, like data mining. Most data warehouses hold information for at least 1 year and sometimes can reach half century, depending on the business/operations data retention requirement. As a result these databases can become very large.

Reporting

The data in the data warehouse must be available to the organisation's staff if the data warehouse is to be useful. There are a very large number of software applications that perform this function, or reporting can be custom-developed. Examples of types of reporting tools include:

- ***Business intelligence tools:*** These are software applications that simplify the process of development and production of business reports based on data warehouse data.
- ***Executive information systems (known more widely as Dashboard (business):*** These are software applications that are used to display complex business metrics and information in a graphical way to allow rapid understanding.

- ***OLAP Tools:*** OLAP tools form data into logical multi-dimensional structures and allow users to select which dimensions to view data by.
- ***Data Mining:*** Data mining tools are software that allow users to perform detailed mathematical and statistical calculations on detailed data warehouse data to detect trends, identify patterns and analyse data.

Metadata

Metadata, or "data about data", is used not only to inform operators and users of the data warehouse about its status and the information held within the data warehouse, but also as a means of integration of incoming data and a tool to update and refine the underlying DW model.

Examples of data warehouse metadata include table and column names, their detailed descriptions, their connection to business meaningful names, the most recent data load date, the business meaning of a data item and the number of users that are logged in currently.

Operations

Data warehouse operations is comprised of the processes of loading, manipulating and extracting data from the data warehouse. Operations also cover user management, security, capacity management and related functions.

Optional Components

In addition, the following components exist in some data warehouses:

1. ***Dependent Data Marts****:* A dependent data mart is a physical database (either on the same hardware as the data warehouse or on a separate hardware platform) that receives all its information from the data warehouse. The purpose of a Data Mart is to provide a sub-set of the data warehouse's data for a specific purpose or to a specific sub-group of the organization. A data mart is exactly like a data warehouse technically, but it serves a different business purpose: it either holds information for only part of a company (such as a division), or it holds a small selection of information for the entire company (to support extra analysis without slowing down the main system). In either case, however, it is not the organization's official repository, the way a data warehouse is.
2. ***Logical Data Marts****:* A logical data mart is a filtered view of the main data warehouse but does not physically exist as a separate data copy. This approach to data marts delivers the same benefits but has the additional advantages of not requiring additional (costly) disk space and it is always as current with data as the main data warehouse. The downside is that Logical Data Marts can have slower response times than physicalized ones.
3. ***Operational Data Store****:* An ODS is an integrated database of operational data. Its sources include legacy systems, and it contains current or near-term data. An ODS may contain 30 to 60 days of information, while a data warehouse typically contains years of data. ODSs are used in some data warehouse architectures to provide near-real-time reporting capability in the event that the Data Warehouse's loading time or architecture prevents it from being able to provide near-real-time reporting capability.

7.4 DIFFERENT METHODS OF STORING DATA IN A DATA WAREHOUSE

All data warehouses store their data grouped together by **subject areas** that reflect the general usage of the data (Customer, Product, Finance etc.). The general principle used in the majority of data warehouses is that data is stored at its most elemental level for use in reporting and information analysis.

Within this generic intent, there are two primary approaches to organising the data in a data warehouse.

The first is using a "dimensional" approach. In this style, information is stored as "facts" which are numeric or text data that capture specific data about a single transaction or event, and

"dimensions" which contain reference information that allows each transaction or event to be classified in various ways. As an example, a sales transaction would be broken up into facts such as the number of products ordered, and the price paid, and dimensions such as date, customer, product, geographical location and salesperson. The main advantages of a dimensional approach is that the Data Warehouse is easy for business staff with limited information technology experience to understand and use. Also, because the data is pre-processed into the dimensional form, the Data Warehouse tends to operate very quickly. The main disadvantage of the dimensional approach is that it is quite difficult to add or change later if the company changes the way in which it does business.

The second approach uses database normalization. In this style, the data in the data warehouse is stored in third normal form. The main advantage of this approach is that it is quite straightforward to add new information into the database — the primary disadvantage of this approach is that it can be rather slow to produce information and reports.

7.5 ADVANTAGES OF USING DATA WAREHOUSE

There are many advantages to using a data warehouse, some of them are:

- Enhances end-user access to a wide variety of data.
- Business decision makers can obtain various kinds of trend reports e.g. the item with the most sales in a particular area / country for the last two years.

7.6 CONCERNS IN USING DATA WAREHOUSE

- Extracting, cleaning and loading data could be time consuming.
- Data warehousing project scope might increase.
- Problems with compatibility with systems already in place.
- Providing training to end-users, who end up not using the data warehouse.
- Security could develop into a serious issue, especially if the data warehouse is web accessible.

Characteristics of Data Warehousing

Subject Oriented

Operational databases, such as order processing and payroll databases, are organized around business processes or functional areas. These databases grew out of the applications they served. Thus, the data was relative to the order processing application or the payroll application. Data on a particular subject, such as products or employees, was maintained separately (and usually inconsistently) in a number of different databases. In contrast, a data warehouse is organized around subjects. This subject orientation presents the data in a much easier-to-understand format for end users and non-IT business analysts.

Integrated

Integration of data within a warehouse is accomplished by making the data consistent in format, naming, and other aspects. Operational databases, for historic reasons, often have major inconsistencies in data representations. For example, a set of operational databases may represent "male" and "female" by using codes such as "m" and "f", by "1" and "2", or by "b" and "g". Often, the inconsistencies are more complex and subtle. In a data warehouse, on the other hand, data is always maintained in a consistent fashion.

Time Variant

Data warehouses are time variant in the sense that they maintain both historical and (nearly) current data. Operational databases, in contrast, contain only the most current, up-to-date data values. Furthermore, they generally maintain this information for no more than a year (and often much less). In contrast, data warehouses contain data that is generally loaded from the operational databases daily, weekly, or monthly which is then typically maintained for a period of 3 to 10 years. This is a major difference between the two types of environments.

Historical information is of high importance to decision makers, who often want to understand trends and relationships between data. For example, the product manager for a Liquefied Natural Gas soda drink may want to see the relationship between coupon promotions and sales. This is information that is almost impossible - and certainly in most cases not cost effective - to determine with an operational database.

Non-Volatile

Non-volatility, the final primary aspect of data warehouses, means that after the data warehouse is loaded there are no changes, inserts, or deletes performed against the informational database. The data warehouse is, of course, first loaded with transformed data that originated in the operational databases.

The data warehouse is subsequently reloaded or, more likely, appended on a periodic basis (usually nightly, weekly, or monthly) with new transformed data from the operational databases. Outside of this loading process, the data warehouse generally stays static. Due to non-volatility, the data warehouse can be heavily optimized for query processing.

Typically, data in the Data Warehouse will be derived from Operational Systems Database after passing through a process of extraction (from the operational databases), transformation (putting the data into a form that is easier for humans to use), cleaning (applying quality and conformity criterion etc. to data) etc.

7.7 OLAP

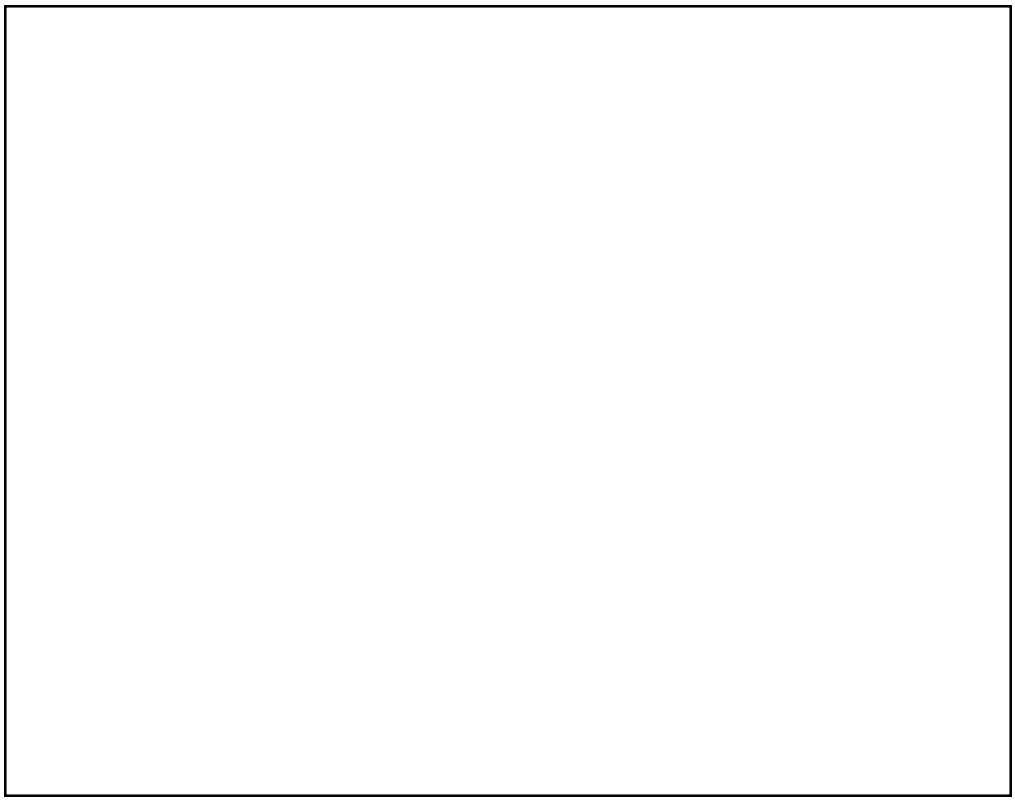


Figure 7.2

OLAP is an acronym for **On Line Analytical Processing**. It is an approach to quickly provide the answer to analytical queries that are dimensional in nature. It is part of the broader category business intelligence, which also includes Extract transform load (ETL), relational reporting and data mining. The typical applications of OLAP are in business reporting for sales, marketing, management reporting, business process management (BPM), budgeting and forecasting, financial reporting and similar areas. The term OLAP was created as a slight modification of the traditional database term OLTP (**On Line Transaction Processing**).

Databases configured for OLAP employ a multidimensional data model, allowing for complex analytical and ad-hoc queries with a rapid execution time. Nigel Pendse has suggested that an alternative and perhaps more descriptive term to describe the concept of OLAP is **Fast Analysis of Shared Multidimensional Information** (FASMI). They borrow aspects of navigational databases and hierarchical databases that are speedier than their relational kin.

The output of an OLAP query is typically displayed in a matrix (or pivot) format. The dimensions form the row and column of the matrix; the measures, the values.

7.8 FUNCTIONALITY

In the core of any OLAP system is a concept of an OLAP cube (also called *multidimensional cube*). It consists of numeric facts called *measures* which are categorized by *dimensions*. The cube metadata is typically created from a star schema or snowflake schema of tables in a relational database. Measures are derived from the records in the fact table and dimensions are derived from the dimension tables. In MOLAP products the cube is populated by copying snapshot of the data from the data source, ROLAP products work directly against the data source without copying data and HOLAP products combine the previous two approaches.

7.9 AGGREGATIONS

It has been claimed that for complex queries OLAP cubes can produce an answer in around 0.1% of the time for the same query on OLTP relational data. The single most important mechanism in OLAP which allows to achieve such performance is use of *aggregations*. Aggregations are built from the fact table by changing the granularity on specific dimensions and aggregating up data along these dimensions. The number of possible aggregations is determined by every possible combination of dimension granularities.

For example a set of customers can be grouped by city, by district or by country; so with 50 cities, 8 districts and two countries there are three hierarchical levels with 60 members. These customers can be considered in relation to products; if there are 250 products with 20 categories, three families and three departments then there are 276 product members. With just these two dimensions there are 16,560 (276 * 60) possible aggregations. As the data considered increases the number of aggregations can quickly total tens of millions or more.

The combination of all possible aggregations and the base data contain the answers to every query which can be answered from the data (as in Gray, Bosworth, Layman, and Pirahesh, 1997). Due to the potentially large number of aggregations to be calculated, often only a predetermined number are fully calculated while the remainder are solved on demand. The problem of deciding which aggregations (a.k.a. views) to calculate is known as the view selection problem. View selection can be constrained by the total size of the selected set of aggregations, the time to update them from changes in the base data, or both. The objective of view selection is typically to minimize the average time to answer OLAP queries, although some studies also minimize the update time as well. Many different approaches have been taken to view selection (which is NP-Complete), including greedy algorithms, randomized search, genetic algorithms and A* search algorithms.

7.10 TYPES

OLAP systems have been traditionally categorized using the following taxonomy:

Multidimensional

MOLAP is the 'classic' form of OLAP and is sometimes referred to as just OLAP. MOLAP uses database structures that are generally optimal for attributes such as time period, location, product or account code. The way that each dimension will be aggregated is defined in advance by one or more hierarchies.

Advantages

- Fast query performance due to optimized storage, multidimensional indexing and caching.
- Smaller on-disk size of data compared to data stored in relational database due to compression techniques.

- Automated computation of higher level aggregates of the data.
- It is very compact for low dimension data sets.
- Array model provides natural indexing.
- Effective data extract achieved through the pre-structuring of aggregated data.

Disadvantages

The processing step (data load) can be quite lengthy, especially on large data volumes. This is usually remedied by doing only incremental processing, i.e., processing only the data which has changed (usually new data) instead of reprocessing the entire data set.

MOLAP tools traditionally have difficulty querying models with dimensions with very high cardinality (i.e., millions of members).

Certain MOLAP tools (e.g., Essbase) have difficulty updating and querying models with more than ten dimensions. This limit differs depending on the complexity and cardinality of the dimensions in question. It also depends on the number of facts or measures stored. Other MOLAP tools (e.g., Microsoft Analysis Services) can handle hundreds of dimensions.

ROLAP

ROLAP works directly with relational databases. The base data and the dimension tables are stored as relational tables and new tables are created to hold the aggregated information. Depends on a specialized schema design. The discussion of the advantages and disadvantages of ROLAP below, focus on those things that are true of the most widely used ROLAP and MOLAP tools available today. In some cases there will be tools which are exceptions to any generalization made.

Advantages of ROLAP

ROLAP is considered to be more scalable in handling large data volumes, especially models with dimensions with very high cardinality (i.e. millions of members).

With a variety of data loading tools available, and the ability to fine tune the ETL code to the particular data model, load times are generally much shorter than with the automated MOLAP loads.

The data is stored in a standard relational database and can be accessed by any SQL reporting tool (the tool does not have to be an OLAP tool).

ROLAP tools are better at handling *non-aggregatable facts* or (e.g. textual descriptions). MOLAP tools tend to suffer from slow performance when querying these elements.

By decoupling the data storage from the multi-dimensional model, it is possible to successfully model data that would not otherwise fit into a strict dimensional model.

Disadvantages of ROLAP

There is a general consensus in the industry that ROLAP tools have slower performance than MOLAP tools. However, see the discussion below about ROLAP performance.

The loading of *aggregate tables* must be managed by custom ETL code. The ROLAP tools do not help with this task. This means additional development time and more code to support.

Many ROLAP dimensional model implementors skip the step of creating aggregate tables. The query performance then suffers because the larger detailed tables must be queried. This can be partially remedied by adding additional aggregate tables, however it is still not practical to create aggregate tables for all combinations of dimensions/attributes.

ROLAP relies on the general purpose database for querying and caching, and therefore several special techniques employed by MOLAP tools are not available (such as special hierarchical indexing). However, modern ROLAP tools take advantage of latest improvements in SQL language

such as CUBE and ROLLUP operators, DB2 Cube Views, as well as other SQL OLAP extensions. These SQL improvements can mitigate the benefits of the MOLAP tools.

Since ROLAP tools rely on SQL for all of the computations, they are not suitable when the model is heavy on calculations which don't translate well into SQL. Examples of such models include budgeting, allocations, financial reporting and other scenarios.

HOLAP

There is no clear agreement across the industry as to what constitutes "Hybrid OLAP", except that a database will divide data between relational and specialized storage. For example, for some vendors, a HOLAP database will use relational tables to hold the larger quantities of detailed data, and use specialized storage for at least some aspects of the smaller quantities of more-aggregate or less-detailed data. HOLAP (Hybrid Online Analytical Process) is a combination of ROLAP and MOLAP which are other possible implementation of OLAP. HOLAP allows to store part of the data in the MOLAP store and another part of the data in ROLAP store. The degree of control that cube designer has over this partitioning varies from product to product.

Student Activity 2

1. In data warehouse technology, a multiple dimensional view can be implemented by a relational database technique (ROLAP), or by a multidimensional database technique (MOLAP), or by a hybrid database technique (HOLAP). Discuss.
2. Discuss the advantages and disadvantages of :
   a. ROLAP
   b. MOLAP
   c. HOLAP

7.11 VERTICAL PARTITIONING

In this mode HOLAP stores *aggregations* in MOLAP for fast query performance, and detailed data in ROLAP to optimize time of cube *processing*.

7.12 HORIZONTAL PARTITIONING

In this mode HOLAP stores some slice of data, usually the more recent one (i.e. sliced by Time dimension) in MOLAP for fast query performance, and older data in ROLAP.

Comparison

Each type has certain benefits, although there is disagreement about the specifics of the benefits between providers.

Some MOLAP implementations are prone to database explosion. Database explosion is a phenomenon causing vast amounts of storage space to be used by MOLAP databases when certain common conditions are met: high number of dimensions, pre-calculated results and sparse multidimensional data. The typical mitigation technique for database explosion is not to materialize all the possible aggregation, but only the optimal subset of aggregations based on the desired performance vs. storage trade off.

MOLAP generally delivers better performance due to specialized indexing and storage optimizations. MOLAP also needs less storage space compared to ROLAP because the specialized storage typically includes compression techniques.

ROLAP is generally more scalable. However, large volume pre-processing is difficult to implement efficiently so it is frequently skipped. ROLAP query performance can therefore suffer.

Since ROLAP relies more on the database to perform calculations, it has more limitations in the specialized functions it can use.

HOLAP encompasses a range of solutions that attempt to mix the best of ROLAP and MOLAP. It can generally pre-process quickly, scale well, and offer good function support.

7.13 THE MULTIDIMENSIONAL DATA MODEL

This unit describes the multidimensional data model and how it is implemented in relational tables and standard form analytic workspaces. It consists of the following topics:

- The Logical Multidimensional Data Model
- The Relational Implementation of the Model
- The Analytic Workspace Implementation of the Model

7.14 THE LOGICAL MULTIDIMENSIONAL DATA MODEL

The multidimensional data model is an integral part of On-Line Analytical Processing, or OLAP. Because OLAP is on-line, it must provide answers quickly; analysts pose iterative queries during interactive sessions, not in batch jobs that run overnight. And because OLAP is also analytic, the queries are complex. The multidimensional data model is designed to solve complex queries in real time.

The multidimensional data model is important because it enforces simplicity. As Ralph Kimball states in his landmark book, *The Data Warehouse Toolkit*:

"The central attraction of the dimensional model of a business is its simplicity.... that simplicity is the fundamental key that allows users to understand databases, and allows software to navigate databases efficiently."

The multidimensional data model is composed of logical cubes, measures, dimensions, hierarchies, levels, and attributes. The simplicity of the model is inherent because it defines objects that represent real-world business entities. Analysts know which business measures they are interested in examining, which dimensions and attributes make the data meaningful, and how the dimensions of their business are organized into levels and hierarchies.

Figure7.3 shows the relationships among the logical objects.

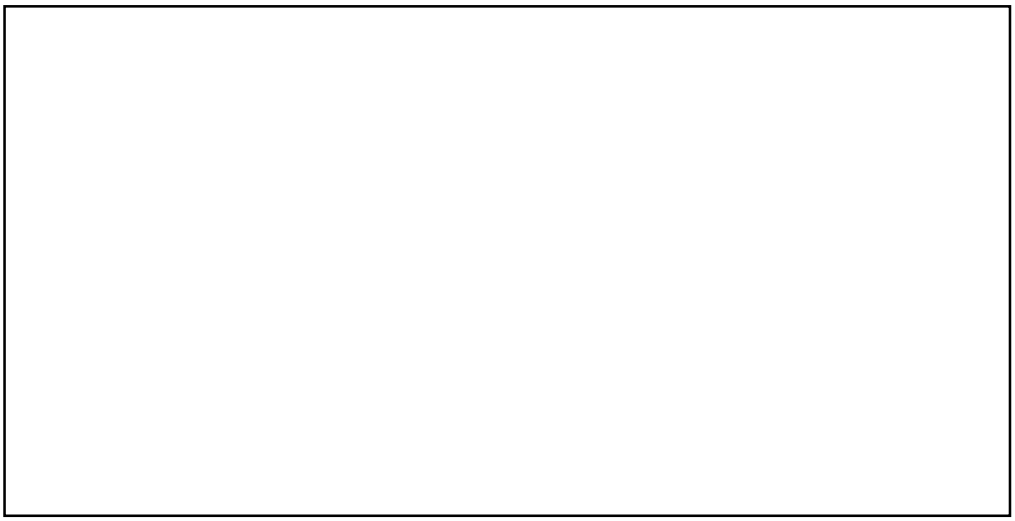


Figure7.3: Diagram of the Logical Multidimensional Model

Logical Cubes

Logical cubes provide a means of organizing measures that have the same shape, that is, they have the exact same dimensions. Measures in the same cube have the same relationships to other logical objects and can easily be analyzed and displayed together.

Logical Measures

Measures populate the cells of a logical cube with the facts collected about business operations. Measures are organized by dimensions, which typically include a Time dimension.

An analytic database contains snapshots of historical data, derived from data in a legacy system, transactional database, syndicated sources, or other data sources. Three years of historical data is generally considered to be appropriate for analytic applications.

Measures are static and consistent while analysts are using them to inform their decisions. They are updated in a batch window at regular intervals: weekly, daily, or periodically throughout the day. Many applications refresh their data by adding periods to the time dimension of a measure, and may also roll off an equal number of the oldest time periods. Each update provides a fixed historical record of a particular business activity for that interval. Other applications do a full rebuild of their data rather than performing incremental updates.

A critical decision in defining a measure is the lowest level of detail (sometimes called the grain). Users may never view this base level data, but it determines the types of analysis that can be performed. For example, market analysts (unlike order entry personnel) do not need to know that Beth Miller in Ann Arbor, Michigan, placed an order for a size 10 blue polka-dot dress on July 6, 2002, at 2:34 p.m. But they might want to find out which color of dress was most popular in the summer of 2002 in the Midwestern United States.

The base level determines whether analysts can get an answer to this question. For this particular question, Time could be rolled up into months, Customer could be rolled up into regions, and Product could be rolled up into items (such as dresses) with an attribute of color. However, this level of aggregate data could not answer the question: At what time of day are women most likely to place an order? An important decision is the extent to which the data has been pre-aggregated before being loaded into a data warehouse.

Logical Dimensions

Dimensions contain a set of unique values that identify and categorize data. They form the edges of a logical cube, and thus of the measures within the cube. Because measures are typically multidimensional, a single value in a measure must be qualified by a member of each dimension to be meaningful. For example, the Sales measure has four dimensions: Time, Customer, Product, and Channel. A particular Sales value (43,613.50) only has meaning when it is qualified by a specific time period (Feb-01), a customer (Warren Systems), a product (Portable PCs), and a channel (Catalog).

Logical Hierarchies and Levels

A **hierarchy** is a way to organize data at different levels of aggregation. In viewing data, analysts use dimension hierarchies to recognize trends at one level, drill down to lower levels to identify reasons for these trends, and roll up to higher levels to see what affect these trends have on a larger sector of the business.

Each **level** represents a position in the hierarchy. Each level above the base (or most detailed) level contains aggregate values for the levels below it. The members at different levels have a one-to-many **parent-child relation**. For example, Q1-02 and Q2-02 are the children of 2002, thus 2002 is the parent of Q1-02 and Q2-02.

Suppose a data warehouse contains snapshots of data taken three times a day, that is, every 8 hours. Analysts might normally prefer to view the data that has been aggregated into days, weeks, quarters, or years. Thus, the Time dimension needs a hierarchy with at least five levels.

Similarly, a sales manager with a particular target for the upcoming year might want to allocate that target amount among the sales representatives in his territory; the allocation requires a dimension hierarchy in which individual sales representatives are the child values of a particular territory.

Hierarchies and levels have a many-to-many relationship. A hierarchy typically contains several levels, and a single level can be included in more than one hierarchy.

Logical Attributes

An **attribute** provides additional information about the data. Some attributes are used for display. For example, you might have a product dimension that uses Stock Keeping Units (SKUs) for dimension members. The SKUs are an excellent way of uniquely identifying thousands of products, but are meaningless to most people if they are used to label the data in a report or graph. You would define attributes for the descriptive labels.

You might also have attributes like colors, flavors, or sizes. This type of attribute can be used for data selection and answering questions such as: Which colors were the most popular in women's dresses in the summer of 2002? How does this compare with the previous summer?

Time attributes can provide information about the Time dimension that may be useful in some types of analysis, such as identifying the last day or the number of days in each time period.

7.15 THE RELATIONAL IMPLEMENTATION OF THE MODEL

The relational implementation of the multidimensional data model is typically a **star schema**, as shown in Figure 7.4, or a **snowflake schema**. A star schema is a convention for organizing the data into dimension tables, fact tables, and materialized views. Ultimately, all of the data is stored in columns, and metadata is required to identify the columns that function as multidimensional objects.

In Oracle Database, you can define a logical multidimensional model for relational tables using the OLAP Catalog or AWXML.The metadata distinguishes level columns from attribute columns in the dimension tables and specifies the hierarchical relationships among the levels. It identifies the various measures that are stored in columns of the fact tables and aggregation methods for the measures. And it provides display names for all of these logical objects.

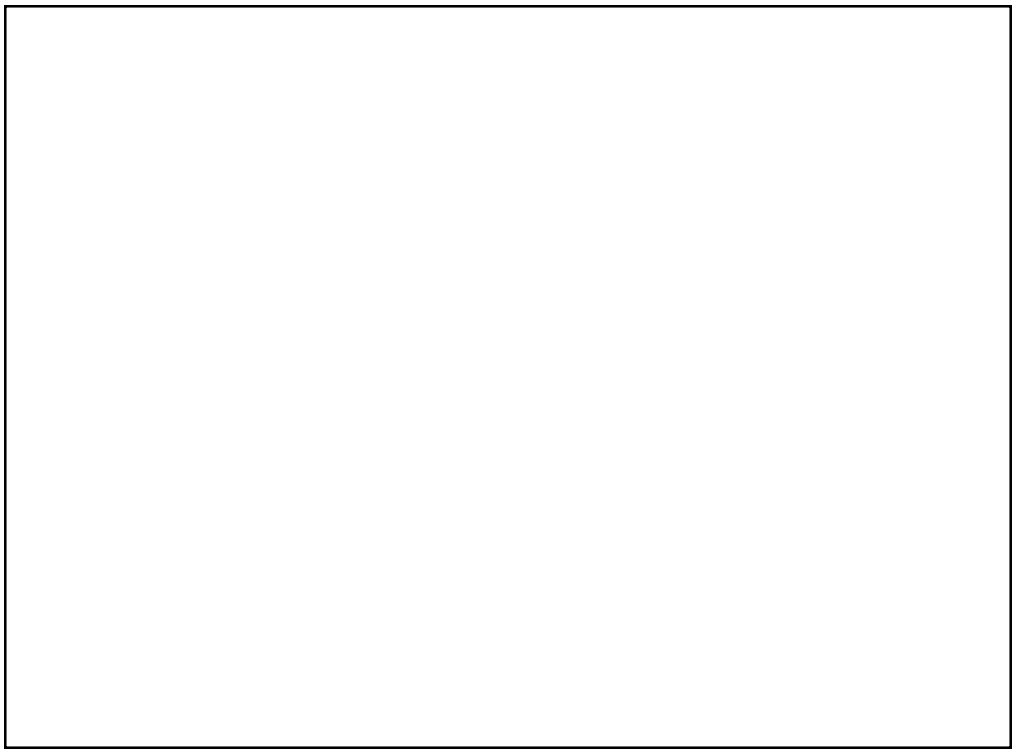


Figure 7.4: Diagram of a Star Schema

Dimension Tables

A star schema stores all of the information about a dimension in a single table. Each level of a hierarchy is represented by a column or column set in the dimension table. A dimension object can be used to define the hierarchical relationship between two columns (or column sets) that represent two levels of a hierarchy; without a dimension object, the hierarchical relationships are defined only in metadata. Attributes are stored in columns of the dimension tables.

A snowflake schema normalizes the dimension members by storing each level in a separate table.

Fact Tables

Measures are stored in fact tables. Fact tables contain a composite primary key, which is composed of several foreign keys (one for each dimension table) and a column for each measure that uses these dimensions.

Materialized Views

Aggregate data is calculated on the basis of the hierarchical relationships defined in the dimension tables. These aggregates are stored in separate tables, called summary tables or materialized views. Oracle provides extensive support for materialized views, including automatic refresh and query rewrite.

Queries can be written either against a fact table or against a materialized view. If a query is written against the fact table that requires aggregate data for its result set, the query is either redirected by query rewrite to an existing materialized view, or the data is aggregated on the fly.

Each materialized view is specific to a particular combination of levels; in Figure 7.4, only two materialized views are shown of a possible 27 (3 dimensions with 3 levels have 3**3 possible level combinations).

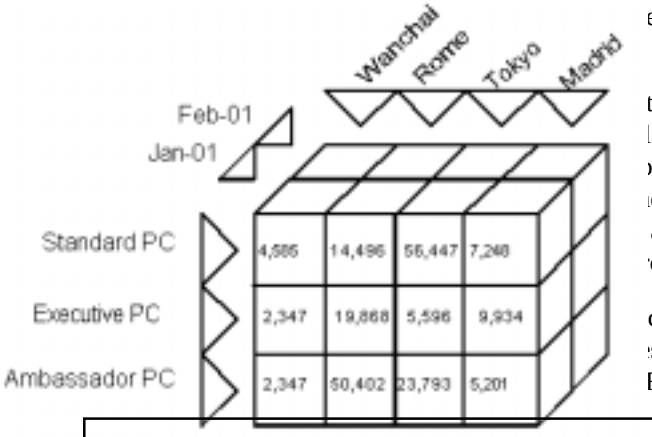
7.16 THE ANALYTIC WORKSPACE IMPLEMENTATION OF THE MODEL

Analytic workspaces have several different types of data containers, such as dimensions, variables, and relations. Each type of container can be used in a variety of ways to store different types of information. For example, a **dimension** can define an edge of a measure, or store the names of all the languages supported by the analytic workspace, or all of the acceptable values of a **relation**. Dimension objects are themselves one dimensional lists of values, while variables and relations are designed specifically to support the efficient storage, retrieval, and manipulation of multidimensional data.

Like relational tables, analytic workspaces have no specific content requirements. You can create an empty analytic workspace, populate it only with OLAP DML programs, or define a single dimension to hold a list of values. This guide, however, describes analytic workspaces that comply with **database standard form**. Database standard form (or simply, standard form) is a convention for instantiating the logical multidimensional model in a particular way so that it can be managed by the current set of Oracle OLAP utilities. It defines a set of metadata that can be queried by any application.

Multidimensional Data Storage in Analytic Workspaces

In the logical multidimensional model, a cube represents all measures with the same shape, that is, the exact same dimensions. In a cube shape, each edge represents a dimension. The dimension



e shape into cells in which data values are

ts the physical storage of multidimensional
l tables. An advantage of the cube shape is
ɔulate or view the data. This is an important
ıd display, because different analysts need
are the Sales Manager for the Pacific Rim,
oduct manager or a financial analyst.

company maintains records that quantify
s region during a specific time period. You
Figure 7.5.

Figure7.5: Comparison of Product Sales By City

Figure 7.5 compares the sales of various products in different cities for January 2001 (shown) and February 2001 (not shown). This view of the data might be used to identify products that are performing poorly in certain markets. Figure 7.5 shows sales of various products during a four-month period in Rome (shown) and Tokyo (not shown). This view of the data is the basis for trend analysis.

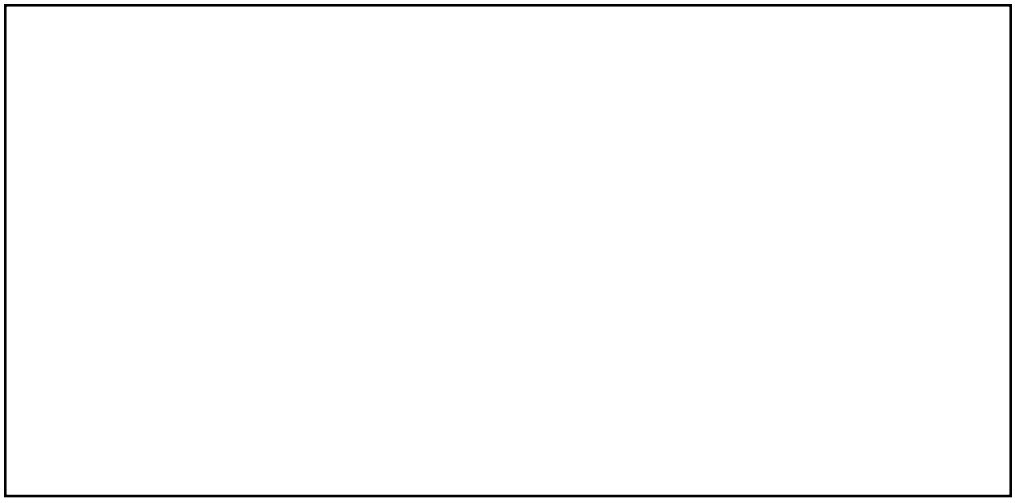


Figure 7.6: Comparison of Product Sales By Month

A cube shape is three dimensional. Of course, measures can have many more than three dimensions, but three dimensions are the maximum number that can be represented pictorially. Additional dimensions are pictured with additional cube shapes.

Database Standard Form Analytic Workspaces

Figure 7.7 shows how dimension, variable, formula, and relation objects in a standard form analytic workspace are used to implement the multidimensional model. Measures with identical dimensions compose a logical cube. All dimensions have attributes, and all hierarchical dimensions have level relations and self-relations; for clarity, these objects are shown only once in the diagram. Variables and formulas can have any number of dimensions; three are shown here.

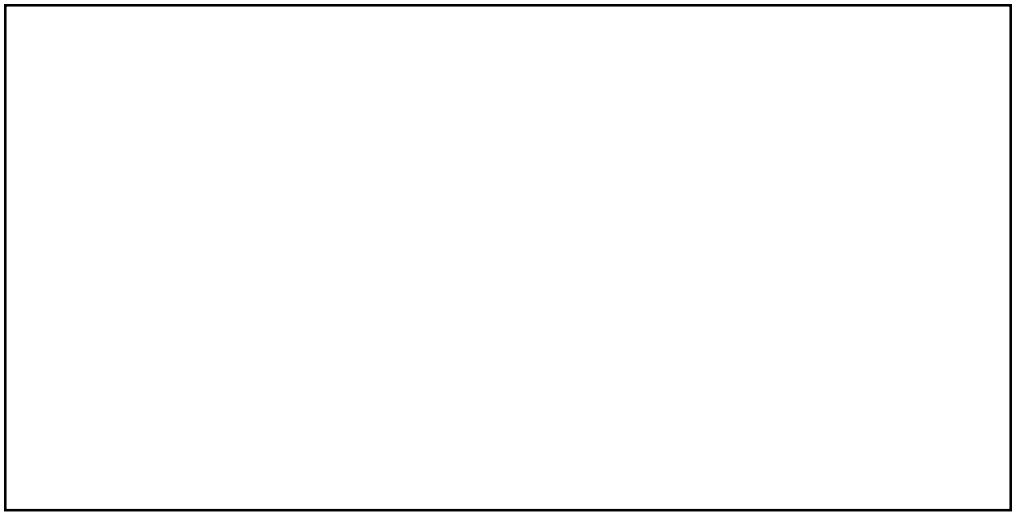


Figure7.7: Diagram of a Standard Form Analytic Workspace

Analytic Workspace Dimensions

A dimension in an analytic workspace is a highly optimized, one-dimensional index of values that serves as a key table. Variables, relations, formulas (which are stored equations) are among the objects that can have dimensions.

Dimensions have several intrinsic characteristics that are important for data analysis:

- ***Referential integrity:*** Each dimension member is unique and cannot be NA (that is, null). If a measure has three dimensions, then each data value of that measure must be qualified by a member of each dimension. Likewise, each combination of dimension members has a value, even if it is NA.

- ***Consistency:*** Dimensions are maintained as separate containers and are shared by measures. Measures with the same dimensionality can be manipulated together easily. For example, if the sales and expense measures are dimensioned by time and line, then you can create equations such as profit = sales - expense.

- ***Preserved order of members:*** Each dimension has a default **status**, which is a list of all of its members in the order they are stored. The default status list is always the same unless it is purposefully altered by adding, deleting, or moving members. Within a session, a user can change the selection and order of the status list; this is called the current status list. The current status list remains the same until the user purposefully alters it by adding, removing, or changing the order of its members.

  Because the order of dimension members is consistent and known, the selection of members can be relative. For example, this function call compares the sales values of all currently selected time periods in the current status list against sales from the prior period.

  lagdif(sales, 1, time)

- ***Highly denormalized:*** A dimension typically contains members at all levels of all hierarchies. This type of dimension is sometimes called an **embedded total** dimension.

In addition to simple dimensions, there are several special types of dimensions used in a standard form analytic workspace, such as composites and concat dimensions. These dimensions are discussed later in this guide.

Use of Dimensions in Standard Form Analytic Workspaces

In an analytic workspace, data dimensions are structured hierarchically so that data at different levels can be manipulated for aggregation, allocation, and navigation. However, all dimension members at all levels for all hierarchies are stored in a single data dimension container. For example, months, quarters, and years are all stored in a single dimension for Time. The hierarchical relationships among dimension members are defined by a parent relation, described in "Analytic Workspace Relations".

Not all data is hierarchical in nature, however, and you can create data dimensions that do not have levels. A line item dimension is one such dimension, and the relationships among its members require a model rather than a multilevel hierarchy. The extensive data modeling subsystem available in analytic workspaces enables you to create both simple and complex models, which can be solved alone or in conjunction with aggregation methods.

As a one-dimensional index, a dimension container has many uses in an analytic workspace in addition to dimensioning measures. A standard form analytic workspace uses dimensions to store various types of metadata, such as lists of hierarchies, levels, and the dimensions composing a logical cube.

Analytic Workspace Variables

A **variable** is a data value table, that is, an array with a particular data type and indexed by a specific list of dimensions. The dimensions themselves are not stored with the variable.

Each combination of dimension members defines a data cell, regardless of whether a value exists for that cell or not. Thus, the absence of data can be purposefully included or excluded from the analysis. For example, if a particular product was not available before a certain date, then the analysis may exclude null values (called NAs) in the prior periods. However, if the product was available but did not sell in some markets, then the analysis may include the NAs.

No special physical relationship exists among variables that share the same dimensions. However, a logical relationship exists because, even though they store different data that may be a different data type, they are identical containers. Variables that have identical dimensions compose a logical cube.

If you change a dimension, such as adding new time periods to the Time dimension, then all variables dimensioned by Time are automatically changed to include these new time periods, even if the other variables have no data for them. Variables that share dimensions (and thus are contained by the same logical cube) can also be manipulated together in a variety of ways, such as aggregation, allocation, modeling, and numeric calculations. This type of calculation is easy and fast in an analytic workspace, while the equivalent single-row calculation in a relational schema can be quite difficult.

Use of Variables to Store Measures

In an analytic workspace, facts are stored in variables, typically with a numeric data type. Each type of data is stored in its own variable, so that while sales data and expenses data might have the same dimensions and the same data type, they are stored in two distinct variables. The containers are identical, but the contents are unique.

An analytic workspace provides a valuable alternative to materialized views for creating, storing, and maintaining summary data. A very sophisticated aggregation system supports modeling in addition to an extensive number of aggregation methods. Moreover, finely grained aggregation rules enable you to decide precisely which data within a single measure is pre-aggregated, and which data within the same measure will be calculated at run-time.

Pre-aggregated data is stored in a compact format in the same container as the base-level data, and the performance impact of aggregating data on the fly is negligible when the aggregation rules have been defined according to known good methods. If aggregate data needed for the result set is stored in the variable, then it is simply retrieved. If the aggregate data does not exist, then it is calculated on the fly.

Use of Variables to Store Attributes

Like measures, attributes are stored in variables. However, there are significant differences between attributes and measures. While attributes are often multidimensional, only one dimension is a data dimension. A hierarchy dimension, which lists the data dimension hierarchies, and a language dimension, which provides support for multiple languages, are typical of the other dimensions.

Attributes provide supplementary information about each dimension member, regardless of its level in a dimension hierarchy. For example, a Time dimension might have three attribute variables, one for descriptive names, another for the period end dates, and a third for period time spans. These attributes provide Time member OCT-02 with a descriptive name of October 2002, an end date of 31-OCT-02, and a time span of 31. All of the other days, months, quarters, and years in the Time dimension have similar information stored in these three attribute variables.

Analytic Workspace Formulas

A formula is a stored equation. A call to any function in the OLAP DML or to any custom program can be stored in a formula. In this way, a formula in an analytic workspace is like a relational view.

In a standard form analytic workspace, one of the uses of a formula object is to provide the interface between aggregation rules and a variable that holds the data. The name of a measure is always the name of a formula, not the underlying variable. While the variable only contains stored data (the base-level data and precalculated aggregates), the formula returns a fully solved measure containing data that is both stored and calculated on the fly. This method enables all queries against a particular measure to be written against the same column of the same relational view for the analytic workspace, regardless of whether the data is calculated or simply retrieved. That column presents data acquired from the formula.

Formulas can also be used to calculated other results like ratios, differences, moving totals, and averages on the fly.

Analytic Workspace Relations

Relations are identical to variables except in their data type. A variable has a general data type, such as DECIMAL or TEXT, while a relation has a dimension as its data type. For example, a relation with a data type of PRODUCT only accepts values that are members of the PRODUCT dimension; an attempt to store any other value causes an error. The dimension members provide a finite list of acceptable values for a relation. A relation container is like a foreign key column, except that it can be multidimensional.

In a standard form analytic workspace, two relations support the hierarchical content of the data dimensions: a parent relation and a level relation.

A **parent relation** is a self-relation that identifies the parent of each member of a dimension hierarchy. This relation defines the hierarchical structure of the dimension.

A **level relation** identifies the level of each member of a dimension hierarchy. It is used to select and sort dimension members based on level.

Indexing

Most data warehouse systems support index structures and materialized views in order to increase their efficiency. Two popular methods are bitmap indexing and join indexing.

The bitmap indexing method allows quick searching in data cubes. The bitmap index is an alternative representation of the record_key list. In the bitmap index for a given attribute, there is a distinct bit vector, Bv, for each value v in the domain of the attribute. If the domain of a given attribute consists of n values, then n bits are needed for each entry in the bitmap index (i.e., there are n bit vectors). If the attribute has the value v for a given row in the data table, then the bit representing that value is set to 1 in the corresponding row of the bitmap index. All other bits for that row are set to 0.

In the HTC data warehouse, suppose the dimension item at the top level has four values - computer; cell phone, electric fan and refrigerator. Each value (e.g., cell phone) is represented by a bit vector in the bitmap index table for item. Similarly if there are two cities - Delhi and Chennai then two bit-vectors are required in the bit-indexing.

The following tables show the bit-map indexing for the two dimensions item and city.

Base table

Key	Item	City	.....	.....
0001	Computer	Delhi		
0002	Cell phone	Delhi		
0026	Computer	Chennai		
0006	Electric fan	Delhi		
0023	Refrigerator	Chennai		
0003	Cell phone	Chennai		
0014	Computer	Delhi		

Table of item bit-index

Key	Computer	CellPhone	ElectricFan	Refrigerator
0001	1	0	0	0
0002	0	1	0	0
0026	1	0	0	0
0006	0	0	1	0
0023	0	0	0	1
0003	0	1	0	0
0014	1	0	0	0

Table of city bit-index

Key	Delhi	Chennai
0001	1	0
0002	1	0
0026	0	1
0006	1	0
0023	0	1
0003	0	1
0014	1	0

Bitmap indexing is advantageous compared to hash and tree indices. It is especially useful for low-cardinality domains because comparison, join, and aggregation operations are then reduced to bit arithmetic, which substantially reduces the processing time.

Metadata Repository

Data about data is called metadata. In context of a data warehouse, metadata are the data that define warehouse objects. Metadata are created for the data names and definitions of the given warehouse. Additional metadata are created and captured for times-tamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes. A typical metadata repository should contain the following:

- A description of the structure of the data warehouse, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- Operational metadata, which include data lineage (history of migrated data and the sequence of transformations applied to it), currency of data (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- The algorithms used for summarization, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- The mapping from the operational environment to the data warehouse, which includes source databases and their contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and security (user authorization and access control).
- Data related to system performance, which include indices and profiles that improve data access and retrieval performance, in addition to rules for the timing and scheduling of refresh, update, and replication cycles.
- Business metadata, which include business terms and definitions, data owner ship information, and charging policies.

A data warehouse contains different levels of summarization, of which metadata is one type. Other types include current detailed data (which are almost always on disk), older detailed data (which are usually on tertiary storage), lightly summarized data and highly summarized data (which may or may not be physically housed).

Metadata plays very different role than other data warehouse data, and are important for many reasons. For example, metadata are used as a directory to help the decision support system analyst locate the contents of the data warehouse, as a guide to the mapping of data when the data are transformed from the operational environment to the data warehouse environment, and as a guide to the algorithms used for summarization between the current detailed data and the lightly summarized data, and between the lightly summarized data and the highly summarized data. Metadata should be stored and managed persistently (i.e., on disk).

Data Warehouse Back-End Tools and Utilities

Data warehouse systems use back-end tools and utilities to populate and refresh their data. These tools and facilities include the following functions:

- Data extraction, which typically gathers data from multiple, heterogeneous, and external sources
- Data cleaning, which detects errors in the data and rectifies them when possible
- Data transformation, which converts data from legacy or host format to ware house format
- Load, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions a Refresh, which propagates the updates from the data sources to the warehouse.

Besides cleaning, loading, refreshing, and metadata definition tools, data warehouse systems usually provide a good set of data warehouse management tools.

Data cleaning and data transformation are important steps in improving the quality of the data and, subsequently, of the data mining results. Since we are mostly interested in the aspects of data warehousing technology related to data mining, we will not get into the details.

Data Warehouse Usage

Data warehouses and data marts are used in a wide range of applications. Business executives in almost every industry use the data collected, integrated, preprocessed, and stored in data warehouses and data marts to perform data analysis and make strategic decisions. In many firms, data warehouses are used as an integral part of a plan-execute-assess “closed-loop” feedback system for enterprise management. Data warehouses are used extensively in banking and financial services, consumer goods and retail distribution sectors, and controlled manufacturing, such as demand-based production.

Typically, the longer a data warehouse has been in use, the more it will have evolved. This evolution takes place throughout a number of phases. Initially, the data warehouse is mainly used for generating reports and answering predefined queries. Progressively, it is used to analyze summarized and detailed data, where the results are presented in the form of reports and charts. Later, the data warehouse is used for strategic purposes, performing multidimensional analysis and sophisticated slice-and-dice operations. Finally, the data warehouse may be employed for knowledge discovery and strategic decision making using data mining tools. In this context, the tools for data warehousing can be categorized into access and retrieval tools, database reporting tools, data analysis tools, and data mining tools.

Business users need to have the means to know what exists in the data warehouse (through metadata), how to access the contents of the data warehouse, how to examine the contents using analysis tools, and how to present the results of such analysis.

There are three kinds of data warehouse applications: information processing, analytical processing, and data mining:

- Information processing supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts, or graphs. A current trend in data warehouse information processing is to construct low-cost Web-based accessing tools that are then integrated with Web browsers.
- Analytical processing supports basic OLAP operations, including slice-and-dice, drill-down, roll-up, and pivoting. It generally operates on historical data in both summarized and detailed forms. The major strength of on-line analytical processing over information processing is the multidimensional data analysis of data warehouse data.
- Data mining supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.

Information processing, based on queries, can find useful information. However, answers to such queries reflect the information directly stored in databases or computable by aggregate functions. They do not reflect sophisticated patterns or regularities buried in the database. Therefore, information processing is not data mining.

On-line analytical processing comes a step closer to data mining since it can derive information summarized at multiple granularities from user-specified subsets of a data warehouse. Such descriptions are equivalent to the class/concept descriptions. Since data mining systems can also mine generalized class/concept descriptions, this raises some interesting questions: “Do OLAP systems perform data mining? Are OLAP systems actually data mining systems?”

The functionalities of OLAP and data mining can be viewed as disjoint: OLAP is a data summarization/aggregation tool that helps simplify data analysis, while data mining allows the automated discovery of implicit patterns and interesting knowledge hidden in large amounts of data. OLAP tools are targeted toward simplifying and supporting interactive data analysis, whereas the goal of data mining tools is to automate as much of the process as possible, while still allowing users to guide the process. In this sense, data mining goes one step beyond traditional on-line analytical processing.

An alternative and broader view of data mining may be adopted in which data mining covers both data description and data modeling, Since OLAP systems can present general descriptions of data from data warehouses, OLAP functions are essentially for user-directed data summary and comparison (by drilling, pivoting, slicing, dicing, and other operations). These are, though limited, data mining functionalities. Yet according to this view, data mining covers a much broader spectrum than simple OLAP operations because it not only performs data summary and comparison, but also performs association, classification, prediction, clustering, time-series analysis, and other data analysis tasks.

Data mining is not confined to the analysis of data stored in data warehouses. It may analyze data existing at more detailed granularities than the summarized data provided in a data warehouse. It may also analyze transactional, spatial, textual, and multimedia data that are difficult to model with current multidimensional database technology. In this context, data mining covers a broader spectrum than OLAP with respect to data mining functionality and the complexity of the data handled.

Since data mining involves more automated and deeper analysis than OLAP, data mining is expected to have broader applications. Data mining can help business managers find and reach more suitable customers, as well as gain critical business insights that may help to drive market share and raise profits. In addition, data mining can help managers understand customer group characteristics and develop optimal pricing strategies accordingly, correct item bundling based not on intuition but on actual item groups derived from customer purchase patterns, reduce promotional spending and at the same time increase the overall net effectiveness of promotions.

7.17 SUMMARY

1. A data warehouse is the main repository of the organization's historical data.
2. The Characteristics of a data warehouse as follows:
   - Subject-oriented
   - Time-variant
   - Non-volatile
   - Integrated
3. Data warehouses in this initial stage are developed by simply copying the database of an operational system to an off-line server where the processing load of reporting does not impact on the operational system's performance.
4. OLAP tools form data into logical multi-dimensional structures and allow users to select which dimensions to view data by.

5. Integration of data within a warehouse is accomplished by making the data consistent in format, naming, and other aspects.

6. Data warehouses are time variant in the sense that they maintain both historical and (nearly) current data.

7. OLAP is an acronym for On Line Analytical Processing. It is an approach to quickly provide the answer to analytical queries that are dimensional in nature.

8. MOLAP generally delivers better performance due to specialized indexing and storage optimizations. MOLAP also needs less storage space compared to ROLAP because the specialized storage typically includes compression techniques.

9. A star schema is a convention for organizing the data into dimension tables, fact tables, and materialized views.

10. A level relation identifies the level of each member of a dimension hierarchy. It is used to select and sort dimension members based on level.

7.18 KEYWORDS

- ***A data warehouse:*** It is a computer system designed for archiving and analyzing an organisation's historical data, such as sales, salaries, or other information from day-to-day operations.

- ***Data Sources:*** Data sources refers to any electronic repository of information that contains data of interest for management use or analytics.

- ***Metadata, or "data about data:*** It is used not only to inform operators and users of the data warehouse about its status and the information held within the data warehouse, but also as a means of integration of incoming data and a tool to update and refine the underlying DW model.

7.19 REVIEW QUESTIONS

1. Describe the various characteristics of data warehouse.

2. Explain the historical evolution of data warehousing.

3. Explain the architecture of data warehouse.

4. List down the various method to store data in data warehouse.

5. Explain the structure of star and snowflake schema with the help of suitable example.

6. Explain the implementation of multidimensional data model in analytical workspace.

7. What are the differences between the three main types of data warehouse usage?

7.20 FURTHER READING

Usama Fayyad, "Mining Databases: Towards Algorithms for Knowledge Discovery", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998.

Christopher Matheus, Philip Chan, and Gregory Piatetsky-Shapiro, "Systems for Knowledge Discovery in Databases", IEEE Transactions on Knowledge and Data Engineering, 5(6):903-913, December 1993.

Rakesh Agrawal and Tomasz Imielinski, "Database Mining: A Performance Perspective", IEEE Transactions on Knowledge and Data Engineering, 5(6):914-925, December 1993.

Usama Fayyad, David Haussler, and Paul Stolorz, "Mining Scientific Data", Communications of the ACM, vol. 39, no. 11, pp. 51-57, November 1996.

David J. Hand, "Data Mining: Statistics and more?", The American Statistician, vol. 52, no. 2, pp 112-118, May 1998.

Tom M. Mitchell, "Does machine learning really work?", AI Magazine, vol. 18, no. 3, pp. 11-20, Fall 1997.

Clark Glymour, David Madigan, Daryl Pregibon, and Padhraic Smyth, "Statistical Inference and Data Mining", Communications of the ACM, vol. 39, no. 11, pp. 35-41, November 1996.

Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford, "Scalable, Distributed Data Mining using An Agent Based Architecture", Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), August 1997, Newport Beach, California, USA.

M-S. Chen, Jiawei Han, and Philip S. Yu, "Data Mining: An Overview from a Database Perspective", IEEE Transactions on Knowledge and Data Engineering, 8(6): 866-883, 1996.

Surajit Chaudhuri, "Data Mining and Database Systems: Where is the Intersection?", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998.

UNIT

8

DATA PREPROCESSING

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Describe Data preprocessing.
- Explain Data preporation
- Define Data cleaning
- Explain Data Integration and Transformation
- Describe various challenges of Data Integration
- Describe various organization challenges

U N I T S T R U C T U R E

8.1 INTRODUCTION

Data preprocessing describes any type of processing performed on raw data to prepare it for *another* processing procedure. Commonly used as a preliminary data mining practice, data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user — for example, in a neural network. There are a number of different tools and methods used for preprocessing, including: *sampling*, which selects a representative subset from a large population of data; *transformation*, which manipulates raw data to produce a single input; *denoising*, which removes noise from data; normalization, which organizes data for more efficient access; and *feature extraction*, which pulls out specified data that is significant in some particular context.

In a customer relationship management (CRM) context, data preprocessing is a component of Web mining. Web usage logs may be preprocessed to extract meaningful sets of data called *user transactions*, which consist of groups of URL references. User sessions may be tracked to identify the user, the Web sites requested and their order, and the length of time spent on each

one. Once these have been pulled out of the raw data, they yield more useful information that can be put to the user's purposes, such as consumer research, marketing, or personalization.

In data mining preprocessing refers to data preparation where you select variables samples, construct new variables and transform existing ones.

Student Activity 1

1. What do you mean by data preprocessing? Explain in context with data mining.

8.2 DATA PREPARATION

Variable selection

Ideally, you would take all the variables/features you have and use them as inputs. In practice, this doesn't work very well. One reason is that the time it takes to build a model increases with the number of variables. Another reason is that blindly including columns can lead to incorrect models. Often your knowledge of the problem domain can let you make many of these selections correctly. For example, including ID numbers as predictor variables will at best have no benefit and at worst may reduce the weight of other important variables.

Sample selection

As in the case of the variables, ideally you would want to make use of all the samples you have. In practice however samples may have to be removed to produce optimal results. You usually want to throw away data that are clearly outliers. In some cases you might want to include them in the model, but often they can be ignored based on your understanding of the problem. In other cases you may want to emphasize certain aspects of your data by duplicating samples.

Variable construction

It is often desirable to create additional inputs based on raw data. For instance forecasting demographics using a GDP per capita ratio rather than just GDP and capita can yield better results. While in theory many adaptive systems can handle this autonomously, in practice helping the system out by incorporating external knowledge can make a big difference.

Variable transformation

Often data needs to be transformed in one way or another before it can be put to use in a system. Typical transformations are scaling and normalization which puts the variables in a restricted range – a necessity for efficient and precise numerical computation.

Web usage data is collected in various ways, each mechanism collecting attributes relevant for its purpose. There is a need to pre-process the data to make it easier to mine for knowledge. Specifically, the following issues need to be addressed:

1. ***Instrumentation & Data Collection:*** Clearly improved data quality can improve the quality of any analysis on it. A problem in the Web domain is the inherent conflict between the analysis needs of the analysts (who want more detailed usage data collected), and the privacy needs of users (who want as little data collected as possible). This has lead to the development of *cookie files* on one side and *cache busting* on the other. The emerging OPS standard on collecting profile data may be a compromise on what can and will be collected. However, it is not clear how much compliance to this can be expected. Hence, there will be a continual need to develop better instrumentation and data collection techniques, based on whatever is possible and allowable at any point in time.

2. ***Data Integration:*** Portions of Web usage data exist in sources as diverse as *Web server logs, referral logs, registration files,* and *index server logs*. Intelligent integration and correlation of information from these diverse sources can reveal usage information which may not be evident from any one of them. Techniques from data integration should be examined for this purpose.

3. ***Transaction Identification:*** Web usage data collected in various logs is at a very fine

granularity. Hence, while it has the advantage of being extremely general and fairly detailed, it also has the corresponding drawback that it cannot be analyzed directly, since the analysis may start focusing on micro trends rather than on the macro trends. On the other hand, the issue of whether a trend is micro or macro depends on the purpose of a specific analysis. Hence, we believe there is a need to group individual data collection events into groups, called *Web transactions* before feeding it to the mining system. While have proposed techniques to do so, more attention needs to be given to this issue.

8.3 DATA CLEANING

Data cleaning routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data. Following are a few basic methods of performing data cleaning:

Missing Values

Imagine that you need to analyze HTC sales and customer data. You note that many tuples have no recorded value for several attributes, such as customer income. How can you go about filling in the missing values for this attribute? Let's look at the following methods:

1. ***Ignore the tuple:*** This is usually done when the class label is missing (assuming the mining task involves classification or description). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.

2. ***Fill in the missing value manually:*** In general, this approach is time-consuming and may not be feasible given a large data set with many missing values.

3. ***Use a global constant to fill in the missing value:*** Replace all missing attribute values by the same constant, such as a label like "Unknown" or -¥. If missing values are replaced by, say, "Unknown," then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common-that of "Unknown." Hence, although this method is simple, it is not recommended.

4. ***Use the attribute mean to fill in the missing value:*** Suppose that the average income of HTC customers is Rs.28,000. Use this value to replace the missing value for income.

5. ***Use the attribute mean for all samples belonging to the same class as the given tuple:*** If classifying customers according to credit_risk, replace the missing value with the average income value for customers in the same credit risk category as that of the given tuple.

6. ***Use the most probable value to fill in the missing value:*** This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for income.

Methods 3 to 6 bias the data. The filled-in value may not be correct. Method 6, however, is a popular strategy. In comparison to the other methods, it uses the most information from the present data to predict missing values. By considering the values of the other attributes in its estimation of the missing value for income, there is a greater chance that the relationships between income and the other attributes are preserved.

Noisy Data

Noise is a random error or variance in a measured variable. The data needs to be smoothened in this case. The following are data smoothing techniques:

1. ***Binning:*** Binning methods smooth a sorted data value by consulting its "neighborhood," that is, the values around it. The sorted values are distributed into a number of "buckets," or bins. Because binning methods consult the neighborhood of values, they perform local smoothing. In order to smoothen price attribute, the data for price are first sorted and then partitioned into equidepth bins of depth 3 (i.e., each bin contains three values). In smoothing by bin means, each value in a bin is replaced by the mean value of the bin. For example, if the

mean values 4, 8, and 15 in a Bin is 9. Therefore, each original value in this bin is replaced by the value 9. Similarly, smoothing by bin medians can be employed, in which each bin value is replaced by the bin median. In smoothing bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value. In general, the larger the width, the greater the effect of the smoothing. Alternatively, bins may be equiwidth where the interval range of values in each bin is constant.

Sorted data for price (in rupees): 4, 8, 15,21,21,24,25,28,34

Partition into (equidepth) bins:

Bin 1:4,8,15

Bin 2:21,21,24

Bin 3: 25, 28, 34

Smoothing by bin means:

Bin 1:9,9,9

Bin 2: 22, 22, 22

Bin 3: 29, 29, 29

Smoothing by bin boundaries:

Bin 1: 4, 4, 15

Bin 2: 21, 21, 24

Bin: 25, 25, 34

2. ***Clustering:*** Outliers may be detected by clustering, where similar values are organized into groups, or "clusters." Intuitively, values that fall outside of the set of clusters may be considered outliers.

3. ***Combined computer and human inspection:*** Outliers may be identified through a combination of computer and human inspection. In one application, for example, an information-theoretic measure was used to help identify outlier patterns in a handwritten character .database for classification. The measure's value reflected the "surprise" content of the predicted character label with respect to the known label. Outlier patterns may be informative (e.g., identifying useful data exceptions, such as different versions of the characters "0" or "7") or "garbage" (e.g., mislabelled characters). Patterns whose surprise content is above a threshold are output to a list. A human can then sort through the patterns in the list to identify the actual garbage ones. This is much faster than having to manually search through the entire database. The garbage patterns can then be excluded from use in subsequent data mining.

4. ***Regression:*** Data can be smoothed by fitting the data to a function, such as with regression. Linear regression involves finding the "best" line to fit two variables, so that one variable can be used to predict the other. Multiple linear regression is an extension of linear regression, where more than two variables are involved and the data are fit to a multidimensional surface. Using regression to find a mathematical equation to fit the data helps smooth out the noise.

Many methods for data smoothing are also methods for data reduction involving discretization. For example, the binning techniques described above reduce the number of distinct values per attribute. This acts as a form of data reduction for logic-based data mining methods, such as decision tree induction, which repeatedly make value comparisons on sorted data. Concept hierarchies are a form of data discretization that can also be used for data smoothing. A concept hierarchy for price, for example, may map real price values into inexpensive, moderately_priced, and expensive, thereby reducing the number of data values to be handled by the mining process. Some methods of classification, such as neural networks, have built-in data smoothing mechanisms.

Inconsistent Data

There may be inconsistencies in the data recorded for some transactions. Some data inconsistencies may be corrected manually using external references. For example, errors made at data entry may be corrected by performing a paper trace. This may be coupled with routines designed to help correct the inconsistent use of codes. Knowledge engineering tools may also be used to detect the violation of known data constraints. For example, known functional dependencies between attributes can be used to find values contradicting the functional constraints.

There may also be inconsistencies due to data integration, where a given attribute can have different names in different databases.

8.4 DATA INTEGRATION AND TRANSFORMATION

The ability to transform corporate data into meaningful and actionable information is the single most important source of competitive advantage in today's business world. Harnessing the data explosion to better understand the past and get direction for the future has turned out to be one of the most challenging ventures for enterprise Information Technology departments in global organizations. There are three broad categories of issues associated with data integration:

- Technology challenges
- Organizational issues
- Economic challenges

In this unit, we will explore these challenges in detail and discuss how to address them with Microsoft® SQL Server™ 2005 Integration Services (SSIS). First, let's view them in the context of a real-world scenario.

A Real-World Scenario

A major global transportation company uses its data warehouse to both analyze the performance of its operations and to predict variances in its scheduled deliveries.

Data Sources

The major sources of data in this company include order data from its DB2-based order entry system, customer data from its SQL Server-based customer relationship management (CRM) system, and vendor data from its Oracle-based ERP system. In addition to data from these major systems, data from spreadsheets tracking "extraordinary" events, which have been entered by hand by shipping supervisors, is incorporated into the data warehouse. Currently, external data such as weather information, traffic status, and vendor details (for subcontracted deliveries) are incorporated on a delayed basis from text files from various sources.

Data Consumption

Not only are the sources for these data diverse, but the consumers are also diverse both in their requirements and their geographic locations. This diversity has led to a proliferation of local systems. One of the major efforts for the Information Technology department is to establish a "single version of the truth," at least for its customer data.

Data Integration Requirements

In view of this diversity of data, business needs, and user requirements, the Information Technology department has specified the following set of data integration requirements:

- They must provide reliable and consistent historical and current data integrated from a variety of internal and external sources.
- To reduce lags in data acquisition, data from providers and vendors must be available via Web services or some other direct mechanism such as FTP.
- They need to cleanse and remove duplicate data and otherwise enforce data quality.

- Increasing global regulatory demands require that the company maintain clear audit trails. It is not enough to maintain reliable data; the data needs to be tracked and certified.

Student Activity 2

1. What do you mean by data integration? List down various data integration requirements.

8.5 CHALLENGES OF DATA INTEGRATION

At one level, the problem of data integration in our real-world scenario is extraordinarily simple. Get data from multiple sources, cleanse and transform the data, and load the data into appropriate data stores for analysis and reporting. Unfortunately, in a typical data warehouse or business intelligence project, enterprises spend 60–80% of the available resources in the data integration stage. Why is it so difficult?

Technology Challenges

Technology challenges start with source systems. We are moving from collecting data on transactions (where customers commit to getting, buying, or otherwise acquiring something) to collecting data on pre-transactions (where customer intentions are tracked via mechanisms such as Web clicks or RFID). Data is now not only acquired via traditional sources and formats, such as databases and text files, but is increasingly available in a variety of different formats (ranging from proprietary files to Microsoft Office documents to XML-based files) and from Internet-based sources such as Web services and RSS (Really Simple Syndication) streams. The most pertinent challenges are:

- Multiple sources with different formats.
- Structured, semi-structured, and unstructured data.
- Data feeds from source systems arriving at different times.
- Huge data volumes.

In an ideal world, even if we somehow manage to get all the data we need in one place, new challenges start to surface, including:

- Data quality.
- Making sense of different data formats.
- Transforming the data into a format that is meaningful to business analysts.

Suppose that we can magically get all the data we need and that we can cleanse, transform, and map the data into a useful format. There is still another shift away from traditional data movement and integration. That is the shift from fixed long batch-oriented processes to fluid and shorter on-demand processes. Batch-oriented processes are usually performed during “downtimes” when users do not place heavy demands on the system. This usually is at night during a predefined batch window of 6-8 hours, when no one is supposed to be in the office. With the increasing globalization of businesses of every size and type, this is no longer true. There is very little (if any) downtime and someone is always in the office somewhere in the world. The sun really doesn’t set on the global business.

As a result we have:

- Increasing pressure to load the data as quickly as possible.
- The need to load multiple destinations at the same time.
- Diverse destinations.

Not only do we need to do all these, but we need to do them as fast as possible. In extreme cases, such as online businesses, data needs to be integrated on a continuous basis. There are no real batch windows and latencies can not exceed minutes. In many of these scenarios, the decision making process is automated with continuously running software.

Scalability and performance become more and more important as we face business needs that can’t tolerate any downtime.

Without the right technology, systems require staging at almost every step of the warehousing and integration process. As different (especially nonstandard) data sources need to be included in the ETL (Extract, Transform, and Load) process and as more complex operations (such as data and text mining) need to be performed on the data, the need to stage the data increases. As illustrated in 8.1, with increased staging the time taken to "close the loop," (i.e., to analyze, and take action on new data) increases as well. These traditional ELT architectures (as opposed to value-added ETL processes that occur prior to loading) impose severe restrictions on the ability of systems to respond to emerging business needs.

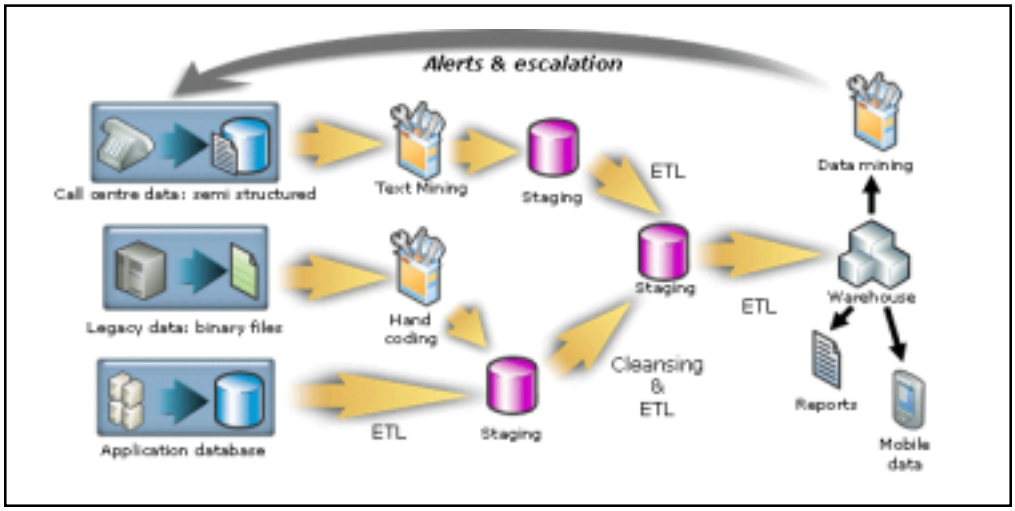


Figure 8.1

Finally, the question of how data integration ties into the overall integration architecture of the organization is becoming more important when both the real-time transactional technology of application integration and the batch-oriented high-volume world of data integration technology are needed to solve the business problems of the enterprise.

8.6 ORGANIZATIONAL CHALLENGES

There are two broad issues with data integration in a large organization; these are the "power" problem, and the "comfort zone" problem.

Power Challenge: Data is power and it is usually very hard to make people think of data in terms of a real valuable shared asset of the company. For enterprise data integration to be successful, all the owners of multiple data sources have to whole-heartedly buy into the purpose and direction of the project. Lack of cooperation from the relevant parties is one the major reasons for the failure of data integration projects. Executive sponsorship, consensus building, and a strong data integration team with multiple stakeholders are a few of the critical success factors that can help resolve the issues.

Comfort Zone Challenge: Problems of data integration, when analyzed in the context of an isolated need, can be solved in multiple ways. About 60% of data integration is solved by hand-coding. The technology used to solve similar problems can range from replication, ETL, SQL, to EAI. People gravitate towards the technology they are familiar with. Although these approaches have overlapping capabilities and can perhaps do the job in isolated cases, these technologies are optimized to solve different sets of problems. When trying to solve the problem of enterprise data integration, the lack of a sound architecture with appropriate technology choices can turn out to be a recipe for failure.

Economic Challenges

The organizational and technology related issues previously outlined conspire together to make data integration the most expensive part of any data warehouse/business intelligence project. The major factors that add to the cost of data integration are:

- Getting the data out in the format that is necessary for data integration ends up being a slow and torturous process fraught with organizational power games.

- Cleansing the data and mapping the data from multiple sources into one coherent and meaningful format is extraordinarily difficult.
- More often than not, standard data integration tools don't offer enough functionality or extensibility to satisfy the data transformation requirements for the project. This can result in the expenditure of large sums of money in consulting costs to develop special ETL code to get the job done.
- Different parts of the organization focus on the data integration problem in silos.

When there is a need to put them all together, additional costs are incurred to integrate these efforts into enterprise-wide data integration architecture.

As the data warehousing and business intelligence needs of the organization evolve, faulty data integration architecture becomes more and more difficult to maintain and the total cost of ownership skyrockets.

8.7 DATA REDUCTION

Now consider that you have selected data from the HTC data warehouse for analysis. The data set will likely be huge. Complex data analysis and mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible.

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results.

Following are some strategies for data reduction:

1. Data cube aggregation, where aggregation operations are applied to the data in the construction of a data cube.
2. Dimension reduction, where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.
3. Data compression, where encoding mechanisms are used to reduce the data set size.
4. Numerosity reduction, where the data are replaced or estimated by alternative, smaller data representations such as parametric models (which need store only the model parameters instead of the actual data), or nonparametric methods such as clustering, sampling, and the use of histograms.
5. Discretization and concept hierarchy generation, where raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction and are a powerful tool for data mining.

Data Cube Aggregation

Imagine that you have collected the data for your analysis. These data consist of the HTC sales per quarter, for the years 1998 to 2000. You are, however, interested in the annual sales (total per year), rather than the total per quarter. Thus the data can be aggregated so that the resulting data summarize the total sales per year instead of per quarter. The resulting data set is smaller in volume, without loss of information necessary for the analysis task.

Dimensionality Reduction

Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task, or redundant. For example, if the task is to classify customers as to whether or not they are likely to purchase a popular new computer at HTC when notified of a sale, attributes such as the customer's telephone number are likely to be irrelevant, unlike attributes such as age or computer_savvyness. Although it may be possible for a domain expert to pick out some of the useful attributes, this can be a difficult and time-consuming task, especially when the behavior of the data is not well known. Leaving out relevant attributes or keeping irrelevant attributes may be

detrimental, causing confusion for the mining algorithm employed. This can result in discovered patterns of poor quality. In addition, the added volume of irrelevant or redundant attributes can slow down the mining process.

Dimensionality reduction reduces the data set size by removing such attributes (or dimensions) from it. Typically, methods of attribute subset selection are applied. The goal of attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. Mining on a reduced set of attributes has an additional benefit. It reduces the number of attributes appearing in the discovered patterns, helping to make the patterns easier to understand.

For n attributes, there are 2^n possible subsets. An exhaustive search for the optimal subset of attributes can be prohibitively expensive, especially as n and the number of data classes increase. Therefore, heuristic methods that explore a reduced search space are commonly used for attribute subset selection. These methods are typically greedy in that, while searching through attribute space, they always make what looks to be the best choice at the time. Their strategy is to make a locally optimal choice in the hope that this will lead to a globally optimal solution. Such greedy methods are effective in practice and may come close to estimating an optimal solution.

The "best" (and "worst") attributes are typically determined using tests of statistical significance, which assume that the attributes are independent of one another. Many other attribute evaluation measures can be used, such as the information gain measure used in building decision trees for classification.

Basic heuristic methods of attribute subset selection include the following techniques:

1. ***Stepwise forward selection:*** The procedure starts with an empty set of attributes. The best of the original attributes is determined and added to the set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.

2. ***Stepwise backward elimination:*** The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.

3. ***Combination of forward selection and backward elimination:*** The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.

The stopping criteria for methods 1 to 3 may vary. The procedure may employ a threshold on the measure used to determine when to stop the attribute selection process.

If the mining task is classification, and the mining algorithm itself is used to determine the attribute subset, then this is called a wrapper approach; otherwise, if is a filter approach. In general, the wrapper approach leads to greater accuracy since it optimizes the evaluation measure of the algorithm while removing attributes. However, it requires much more computation than a filter approach.

Data Compression

In data compression, data encoding or transformations are applied so as to obtain a reduced or compressed representation of the original data. If the original data can be reconstructed from the compressed data without any loss of information, the data compression technique used is called lossless. If, instead, we can reconstruct only an approximation of the original data, then the data compression technique is called lossy. There are several well-tuned algorithms for string compression. Although they are typically lossless, they allow only limited manipulation of the data. In this section, we instead focus on two popular and effective methods of lossy data compression: wavelet transforms and principal components analysis.

Wavelet Transforms

The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector D, transforms it to a numerically different vector, D', of wavelet coefficients. The two vectors are of the same length.

The usefulness of this transform lies in the fact that the wavelet transformed data can be truncated. A compressed approximation of the data can be retained by storing only a small fraction of the strongest of the wavelet coefficients. For example, all wavelet coefficients larger than some user-specified threshold can be retained. The remaining coefficients is set to 0. The resulting data representation is therefore very sparse, so that operations that can take advantage of data sparsity are computationally very fast if performed in wavelet space. The technique also works to remove noise without smoothing out the main features of the data, making it effective for data cleaning as well. Given a set of coefficients, an approximation of the original data can be constructed by applying the inverse of the DWT used.

The DWT is closely related to the discrete Fourier transform (DFT), a signal processing technique involving trigonometric sines and cosines. In general, however, the DWT achieves better lossy compression. That is, if the same number of coefficients is retained for a DWT and a DFT of a given data vector, the DWT version will provide a more accurate approximation of the original data. Hence, for an equivalent approximation, the DWT requires less space than the DFT. Unlike the DFT, wavelets are quite localized in space, contributing to the conservation of local detail.

There is only one DFT, yet there are several families of DWTs. Popular wavelet transforms include the Haar_2, Daubechies_4, and Daubechies_6 transforms. The general procedure for applying a discrete wavelet transform Uses a hierarchical pyramid algorithm that halves the data at each iteration, resulting in fast computational speed. The method is as follows:

1. The length, L, of the input data vector must be an integer power of 2. This condition can be met by padding the data vector with zeros, as necessary.

2. Each transform involves applying two functions. The first applies some data smoothing, such as a sum or weighted average. The second performs a weighted difference, which acts to bring out the detailed features of the data.

3. The two functions are applied to pairs of the input data, resulting in two sets of data of length L/2. In general, these represent a smoothed or low frequency version of the input data, and the high frequency content of it, respectively.

4. The two functions are recursively applied to the sets of data obtained in the previous loop, until the resulting data sets obtained are of length 2.

5. A selection of values from the data sets obtained in the above iterations are designated the wavelet coefficients of the transformed data.

Wavelet transforms can be applied to multidimensional data, such as a data cube. This is done by first applying the transform to the first dimension, then to the second, and so on. The computational complexity involved is linear with respect to the number of cells in the cube. Wavelet transforms give good results on sparse or skewed data and on data with ordered attributes. Lossy compression by wavelets is reportedly better than JPEG compression, the current commercial standard. Wavelet transforms have many real-world applications, including the compression of fingerprint images, computer vision, analysis of time-series data, and data cleaning.

Principal Components Analysis

Components analysis is a method of data compression. Suppose that the data to be compressed consist of N tuples or data vectors, from k dimensions. Principal components analysis, or PCA (also called the KarhunenLoeve, or K-L method), searches for c k-dimensional orthogonal vectors that can best be used to represent the data, where c£k. The original data are thus projected onto a much smaller space, resulting in data compression. PCA can be used as a form of dimensionality reduction. However, unlike attribute subset selection, which reduces the attribute set size by retaining a subset of the initial set of attributes, PCA combines the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set.

The basic procedure is as follows:

1. The input data are normalized, so that each attribute falls within the same range. This step helps ensure that attributes with large domains will not dominate attributes with smaller domains.

2. PCA computes c orthonormal vectors that provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others. These vectors are referred to as the principal components. The input data are a linear combination of the principal components.

3. The principal components are sorted in order of decreasing significance or strength. The principal components essentially serve as a new set of axes for the data, providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on.

4. Since the components are sorted according to decreasing order of significance, the size of the data can be reduced by eliminating the weaker components, that is, those with low variance. Using the strongest principal components, it should be possible to reconstruct a good approximation of the original data.

PCA is computationally inexpensive, can be applied to ordered and unordered attributes, and can handle sparse data and skewed data. Multidimensional data of more than two dimensions can be handled by reducing the problem to two dimensions. For example, a 3-D data cube for sales with the dimensions itemtype, branch, and year must first be reduced to a 2-D cube, such as with the dimensions item_type and branch x year. In comparison with wavelet transforms for data compression, PCA tends to be better at handling sparse data, while wavelet transforms are more suitable for data of high dimensionality.

Numerosity Reduction

Data volume can also be reduced choosing alternative, smaller forms of data representation. Techniques of numerosity reduction can be applied for this purpose. These techniques may be parametric or nonparametric. For parametric methods, a model is used to estimate the data, so that typically only the data parameters need be stored, instead of the actual data. (Outliers may also be stored.) Log-linear models, which estimate discrete multidimensional probability distributions, are an example. Nonparametric methods for storing reduced representations of the data include histograms, clustering, and sampling.

Let's have a look at each of the numerosity reduction techniques mentioned above.

Regression and Log-Linear Models

Regression and log-linear models can be used to approximate the given data. In linear regression, the data are modeled to fit a straight line. For example, a random variable, Y (called a response variable), can be modeled as a linear function of another random variable, X (called a predictor variable), with the equation

$$Y = a + \beta X$$

where the variance of Y is assumed to be constant. The coefficients a and b (called regression coefficients) specify the Y - intercept and slope of the line, respectively. These coefficients can be solved for by the method of least squares, which minimizes the error between the actual line separating the data and the estimate of the line. Multiple regression is an extension of linear regression allowing a response variable Y to be modeled as a linear function of a multidimensional feature vector.

Log-linear models approximate discrete multidimensional probability distributions. The method can be used to estimate the probability of each cell in a base cuboid for a set of discretized attributes, based on the smaller cuboids making up the data cube lattice. This allows higher-order data cubes to be constructed from lower-order ones. Log-linear models are therefore also useful for data compression (since the smaller-order cuboids together typically occupy less space than the base cuboid) and data smoothing (since cell estimates in the smaller-order cuboids are less subject to sampling variations than cell estimates in the base cuboid).

Histograms

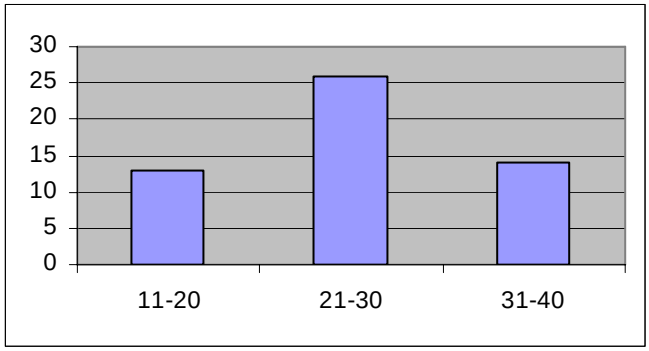
Histograms use binning to approximate data distributions and are a popular form of data reduction. A histogram for an attribute A partitions the data distribution of A into disjoint subsets, or

buckets. The buckets are displayed on a horizontal axis, while the height (and area) of a bucket typically reflects the average frequency of the values represented by the bucket. If each bucket represents only a single attribute-value/frequency pair, the buckets are called singleton buckets. Often, buckets instead represent continuous ranges for the given attribute.

The following data are a list of prices of commonly sold items at HTC (rounded to the nearest rupee). The numbers have been sorted: 11, 11, 15, 15, 15, 15, 15, 18, 18, 20, 20, 20, 20, 22, 24, 24, 24, 25, 25, 25, 25, 25, 25, 28, 28, 28, 28, 28, 28, 28, 28, 28, 30, 30, 30, 30, 30, 30, 30, 31, 31, 31, 31, 35, 35, 35, 35, 35, 38, 38, 40, 40, 40.

The histogram for this data using singleton buckets is:

To further reduce the data, it is common to have each bucket denote a continuous range of values for the given attribute. In the histogram given below, each bucket represents a different Rs.10 range for price.



There are several partitioning rules, including the following:

- ***Equiwidth:*** In an equiwidth histogram, the width of each bucket range is uniform.
- ***Equidepth (or equiheight):*** In an equidepth histogram, the buckets are created so that, roughly, the frequency of each bucket is constant (that is, each bucket contains roughly the same number of contiguous data samples).
- ***V-Optimal:*** If we consider all of the possible histograms for a given number of buckets, the V-Optimal histogram is the one with the least variance. Histogram variance is a weighted sum of the original values that each bucket represents, where bucket weight is equal to the number of values in the bucket.
- ***MaxDiff:*** In a MaxDiff histogram, we consider the difference between each pair of adjacent values. A bucket boundary is established between each pair for pairs having the b - 1 largest differences, where b is user-specified.

V-Optimal and MaxDiff histograms tend to be the most accurate and practical. Histograms are highly effective at approximating both sparse and dense data, as well as highly skewed and uniform data. The histograms described above for single attributes can be extended for multiple attributes. Multidimensional histograms can capture dependencies between attributes. Such histograms have been found effective in approximating data with up to five attributes. More

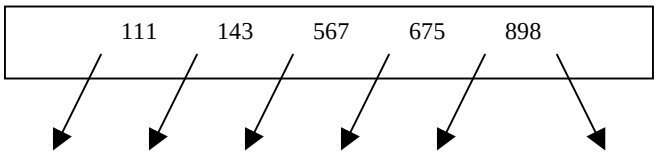
studies are needed regarding the effectiveness of multidimensional histograms for very high dimensions. Singleton buckets are useful for storing outliers with high frequency.

Clustering

Clustering techniques consider data tuples as objects. They partition the objects into groups or clusters, so that objects within a duster are similar to one another and dissimilar to objects in other dusters. Similarity is commonly defined in terms of how close the objects are in space, based on a distance function. The quality of a cluster may be represented by its diameter, the maximum distance between any two objects in the cluster. Centroid distance is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid (denoting the "average object," or average point in space for the cluster).

In database systems, multidimensional index trees are primarily used for providing fast data access. They can also be used for hierarchical data reduction, providing a multi resolution clustering of the data. This can be used to provide approximate answers to queries. An index tree recursively partitions the multidimensional space for a given set of data objects, with the root node representing the entire space. Such trees are typically balanced, consisting of internal and leaf nodes. Each parent node contains keys and pointers to child nodes that, collectively, represent the space represented by the parent node. Each leaf node contains pointers to the data tuples they represent (or to the actual tuples).

An index tree can therefore store aggregate and detail data at varying levels of resolution or abstraction. It provides a hierarchy of clusterings of the data set, where each cluster has a label that holds for the data contained in the cluster. If we consider each child of a parent node as a bucket, then an index tree can be considered as a hierarchical histogram. For example, consider the root of a B+- tree as shown below, with pointers to the data keys 111, 143, 567, 675, and 898.



Suppose that the tree contains 1000 tuples with keys ranging from 1 to 999. The data in the tree can be approximated by an equidepth histogram of six buckets for the key ranges 1 to 110, 111 to 142, 143 to 566, 567 to 674, 675 to 897, and 898 to 999. Each bucket contains roughly 1000/6 items. Similarly, each bucket is subdivided into smaller buckets, allowing for aggregate data at a finer-detailed level. The use of multidimensional index trees as a form of data reduction relies on an ordering of the attribute values in each dimension. Multidimensional index trees include R-trees, quad-trees; and their variations. They are well suited for handling both sparse and skewed data.

Sampling

Sampling can be used as a data reduction technique since it allows a large data set to be represented by a much smaller random sample (or subset) of the data. Suppose that a large data set, D, contains N tuples. Let's have a look at some possible samples for D.

Simple random sample without replacement (SRSWOR) of size n: This is created by drawing n of the N tuples from D (n < N), where the probability of drawing any tuple in D is 1/N, that is, all tuples are equally likely.

- ***Simple random sample with replacement (SRSWR) of size n:*** This is similar to SRSWOR, except that each time a tuple is drawn from D, it is recorded and then replaced. That is; after a tuple is drawn, it is placed back in D so that it maybe drawn again.
- ***Cluster sample:*** If the tuples in D are grouped into M mutually disjoint clusters, then an SRS of m clusters can be obtained, where m<M. For example, tuples in a database are usually retrieved a page at a time, so that each page can be considered a cluster. A reduced data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the tuples.

- ***Stratified sample:*** If D is divided into mutually disjoint parts called strata, a stratified sample of D is generated by obtaining an SRS at each stratum. This helps to ensure a representative sample, especially when the data are skewed. For example, a stratified sample may be obtained from customer data, where a stratum is created for each customer age group. In this way, the age group having the smallest number of customers will be sure to be represented.

An advantage of sampling for data reduction is that the cost of obtaining a sample is proportional to the size of the sample, n, as opposed to N, the data set size. Hence, sampling complexity is potentially sublinear to the size of the data. Other data reduction techniques can require at least one complete pass through D. For a fixed sample size, sampling complexity increases only linearly as the number of data dimensions, d, increases, while techniques using histograms, for example, increase exponentially in d.

When applied to data reduction, sampling is most commonly used to estimate the answer to an aggregate query. It is possible (using the central limit theorem) to determine a sufficient sample size for estimating a given function within a specified degree of error. This sample size, n, may be extremely small in comparison to N. Sampling is a natural choice for the progressive refinement of a reduced data set. Such a set can be further refined by simply increasing the sample size.

8.8 DISCRETIZATION AND CONCEPT HIERARCHY

A continuous attribute may have infinite number of values. For example, Temperature attribute of a clinical thermometer, may take any value between 90F to 110F. Discretization techniques can be used to reduce the number of values for a continuous attribute, by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values. Reducing the number of values for an attribute is especially beneficial if decision-tree-based methods of classification mining are to be applied to the preprocessed data. Typically these methods are recursive. Large amount of time is spent on sorting the data. Hence, the smaller the number of distinct values to sort, the faster these methods should be.

A concept hierarchy for a given numeric attribute defines a discretization of the attribute. Concept hierarchies can be used to reduce the data by collecting and replacing low-level concepts (such as numeric values for the attribute age) by higher-level concepts (such as young, middle-aged, or old). Although detail is lost by such data generalization, the generalized data may be more meaningful and easier to interpret, and will require less space than the original data. Mining on a reduced data set will require fewer input/output operations and be more efficient than mining on a larger, ungeneralized data set. More than one concept hierarchy can be defined for the same attribute in order to accommodate the needs of the various users.

Manual definition of concept hierarchies can be a tedious and time-consuming task for the user or domain expert. Fortunately, many hierarchies are implicit within the database schema and can be defined at the schema definition level. Concept hierarchies often can be automatically generated or dynamically refined based on statistical analysis of the data distribution.

Discretization and Concept Hierarchy Generation for Numeric Data

It is difficult and laborious to specify concept hierarchies for numeric attributes due to the wide diversity of possible data ranges and the frequent updates of data values. Such manual specification can also be quite arbitrary.

Concept hierarchies for numeric attributes can be constructed automatically based on data distribution analysis. We examine five methods for numeric concept hierarchy generation: binning, histogram analysis, cluster analysis, entropy-based discretization, and data segmentation by natural partitioning.

Binning

These methods can also be used for discretization. For example, attribute values can be discretized by distributing the values into bins, and replacing each bin value by the bin mean or median, as in smoothing by bin means or smoothing by bin medians, respectively. These techniques can be applied recursively to the resulting partitions in order to generate concept hierarchies.

Histogram Analysis

Histograms can also be used for discretization. Partitioning rules can be used to define the ranges of values. With an equidepth histogram, the values are partitioned so that, ideally, each partition contains the same number of data samples. The histogram analysis algorithm can be applied recursively to each partition in order to automatically generate a multilevel concept hierarchy, with the procedure terminating once a pre-specified number of concept levels has been reached. A minimum interval size can also be used per level to control the recursive procedure. This specifies the minimum width of a partition, or the minimum number of values for each partition at each level.

Cluster Analysis

A clustering algorithm can be applied to partition data into clusters or groups. Each cluster forms a node of a concept hierarchy, where all nodes are at the same conceptual level. Each cluster may be further decomposed into several subclusters, forming a lower level of the hierarchy. Clusters may also be grouped together in order to form a higher conceptual level of the hierarchy.

Entropy-Based Discretization

An information-based measure called entropy can be used to recursively partition the values of a numeric attribute A, resulting in a hierarchical discretization. Such a discretization forms a numerical concept hierarchy for the attribute. Given a set of data tuples, S, the basic method for entropy-based discretization of A is as follow:

1. Each value of A can be considered a potential interval boundary or threshold T. For example, a value v of A can partition the samples in S into two subsets satisfying the conditions $A < v$ and $A \geq v$, respectively, thereby creating a binary discretization.

2. Given S, the threshold value selected is the one that maximizes the information gain resulting from the subsequent partitioning. The information gain is

$$I(S,T) = \frac{|S_1|}{|S|}\text{Ent}(S_1) + \frac{|S_2|}{|S|}\text{Ent}(S_2),$$

where S_1 and S_2 correspond to the samples in S satisfying the conditions $A < T$ and $A \geq T$, respectively. The entropy function Ent for a given set is calculated based on the class distribution of the samples in the set. For example, given m classes, the entropy of S_1 is

$$\text{Ent}(S_1) = -\sum_{i=1}^{m} P_i \log_2(P_i),$$

where Pi is the probability of class i in S_1, determined by dividing the number of samples of class i in S_1 by the total number of samples in S_1. The value of Ent(S_2) can be computed similarly.

3. The process of determining a threshold value is recursively applied to each partition obtained, until some stopping criterion is met, such as

Ent(S) - I(S, T) > 8.

Entropy-based discretization can reduce data size. Unlike the other methods mentioned here so far, entropy-based discretization uses class information. This makes it more likely that the interval boundaries are defined to occur in places that may help improve classification accuracy. The information gain and entropy measures described here are also used for decision tree induction.

Segmentation by Natural Partitioning

Although binning, histogram analysis, clustering, and entropy-based discretization are useful in the generation of numerical hierarchies, many users would like to see numerical ranges partitioned into relatively uniform, easy-to-read intervals that appear intuitive or natural. For example, annual salaries broken into ranges like (Rs.50,000, Rs.60,000] are often more desirable than ranges like (Rs.51,263.98, Rs.60,872.34], obtained by some sophisticated clustering analysis.

The 3-4-5 rule can be used to segment numeric data into relatively uniform, natural intervals. In general, the rule partitions a given range of data into 3, 4, or 5 relatively equiwidth intervals, recursively and level by level, based on the value range at the most significant digit. The rule is as follows:

If an interval covers 3, 6, 7, or 9 distinct values at the most significant digit, then partition the range into 3 intervals (3 equiwidth intervals for 3, 6, 9, and 3 intervals in the grouping of 2-3-2 for 7).

If it covers 2, 4, or 8 distinct values at the most significant digit, then partition the range into 4 equiwidth intervals.

If it covers 1,5, or l0 distinct values at the most significant digit, then partition the range into 5 equiwidth intervals.

The rule can be recursively applied to each interval, creating a concept hierarchy for the given numeric attribute. Since there could be some dramatically large positive or negative values in a data set, the top-level segmentation, based merely on the minimum and maximum values, may derive distorted results. For example, the assets of a few people could be several orders of magnitude higher than those of others in a data set. Segmentation based on the maximal asset values may lead to a highly biased hierarchy. Thus the top-level segmentation can be performed based on the range of data values representing the majority (e.g., 5th percentile to 95th percentile) of the given data. The extremely high or low values beyond the top level segmentation will form distinct interval(s) that can be handled separately, but in a similar manner.

8.9 SUMMARY

- Data preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure.
- There are a number of different tools and methods used for preprocessing, including: sampling, transformation, denoising, and feature extraction.
- In a customer relationship management (CRM) context, data preprocessing is a component of Web mining.
- Web usage data is collected in various ways, each mechanism collecting attributes relevant for its purpose. There is a need to pre-process the data to make it easier to mine for knowledge.
- Instrumentation & Data Collection, Data Integration, Transaction Identification are the various pre processes.
- The ability to transform corporate data into meaningful and actionable information is the single most important source of competitive advantage in today's business world.
- Data is now not only acquired via traditional sources and formats, such as databases and text files, but is increasingly available in a variety of different formats.
- There are two broad issues with data integration in a large organization; these are the "power" problem, and the "comfort zone" problem.
- The major factors that add to the cost of data integration are:
    - Getting the data out in the format that is necessary for data integration ends up being a slow and torturous process fraught with organizational power games.

- Cleansing the data and mapping the data from multiple sources into one coherent and meaningful format is extraordinarily difficult.
- More often than not, standard data integration tools don't offer enough functionality or extensibility to satisfy the data transformation requirements for the project. This can result in the expenditure of large sums of money in consulting costs to develop special ETL code to get the job done.

8.10 KEYWORDS

- ***Data preprocessing:*** It describes any type of processing performed on raw data to prepare it for another processing procedure.
- ***Clustering:*** Clustering techniques consider data tuples as objects.
- ***Sampling:*** Sampling can be used as a data reduction technique since it allows a large data set to be represented by a much smaller random sample (or subset) of the data.

8.11 REVIEW QUESTIONS

1. What are the different issues to be address in order to preprocess the data to mine the knowledge?
2. Suppose that the data for analysis include the attribute age. The age value for the data tuples are (in increasing order):

   13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

   a. Use smoothing by bin means to smooth the above data, using a bin depth of 3.

   b. How might you determine outliers in the data?

   c. What other methods are there for data smoothing?

3. Using the data for age given in question 3, answer the following:

   a. Use min-max normalization to transform the value 35 for age onto the range [0.0, 1.0].

   b. Use z-score normalization to transform the value 35 for age, where the standard deviation of age is 12.94 years.

   c. Comment on which method you would prefer to use for the given data, giving reasons as to why.

4. Write short note on:

   a. Wavelet Transformation

   b. Principal Components Analysis

   c. Numerosity Reduction

8.12 FURTHER READINGS

Adriaans, P. and Zantige, D. Data Mining, Harlow, UK: Addison-Wesley, 1996

Berry, M.J.A. and Linoff, G., Data Mining Techniques for Marketing, Sales, and Customer Support, New York, NY: John Wiley and Sons, 1997

Bishop, C. Neural Networks for Pattern Recognition, Oxford, UK: Clarendon Press, 1995

Breiman, L., Fredman, J., Olshen, R.A., and Stone, C.J., Classification and Regression Trees, Monterey, CA: Wadsworth & Brooks, 1984

Chester, M., Neural Networks: A Tutorial, Englwood cliffs, NJ, Prentice Hall, 1993

Cressie, N., Statistics for Spatial Data, Revised Edition, Wiley, New York, 1993

Fayyad, U.M.; Piatestsky-Shapiro G.; Smyth D.; and Uthurusamy R., Advances in Knowledge Discovery and Data Mining, Cambridge, MA: AAAI Press/MIT Press, 1996

Fukunaga, K., Introduction to Statistical Pattern Recognition, Academic Press, 1990

Groth, R., Data Mining, A Hands-On Approach for business Professionals, Prentice Hall, 1998 [with demo software]

Hand, D.J., Mannila, H., Smyth, P., Principles of Data Mining, MIT Press, [late] 2000

Jain, A. and Dubes, R., Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988

Jordan, M.I., Learning in Graphical Models, MIT Press, 1999

Kargupta, Hillol, and Chan, Philip, Advances in Distributed and Parallel Knowledge Discovery, MIT/AAAI Press, 2000

Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs," New York: Springer-Verlag, 1994

Mitchell, T.M., Machine Learning, New York, NY: McGrawl Hill, 1997

Ripley, B.D., Pattern Recognition and Neural Networks, Cambridge, UK: Cambridge University Press, 1996

Shekhar, S. and Chawla, S., Spatial Databases: Issues, Implementations, and Trends, Prentice Hall, 2000

Tucker, A.B., Ed, The Computer Science and Engineering Handbook, CRC Press, 1997

Tukey, J., Exploratory Data Analysis, Reading, MA, Addison-Wesley, 1977

Weiss, S.M. and Indurkhya, N., Predictive Data Mining: A Practical Guide, San Francisco, CA: Morgan Kaufmann Publishers, 1998

Welstead, S.T., Neural Network and Fuzzy Logic Applications in C/C++, New York, John Wiley & Sons, 1994

Witten, I.H. and Frank, E., Data Mining: Practical Machine Learning Tools and Techniques with JAVA Implementations, San Francisco, CA: Morgan Kaufmann, 1999

UNIT

9

ELEMENT OF DATA MINING SYSTEM

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Know the need for DMQL
- Describe various Proposals for Association Rule Mining
- Discuss Primitives
- Understand Measures of Interestingness

U N I T S T R U C T U R E

9.1 INTRODUCTION

Many data mining algorithms enable to extract different types of patterns from data (e.g., local patterns like item sets and association rules, models like classifiers). To support the whole knowledge discovery process, we need for integrated systems which can deal either with patterns and data. The inductive database approach has emerged as an unifying framework for such systems. Following this database perspective, knowledge discovery processes become querying processes for which query languages have to be designed. In the prolific field of association rule mining, different proposals of query languages have been made to support the more or less declarative specification of both data and pattern manipulations.. It enables to identify nowadays shortcomings and to point out some promising directions of research in this area.

9.2 THE NEED FOR DATA MINING QUERY LANGUAGES

Since the first definition of the Knowledge Discovery in Databases (KDD) domain in (Piatetsky-Shapiro and Frawley, 1991), many techniques have been proposed to support these “From Data to Knowledge” complex interactive and iterative processes. In practice, knowledge elicitation is based on some extracted and materialized (collections of) patterns which can be global (e.g., decision trees) or local (e.g., item sets, association rules). Real life KDD processes imply complex pre-processing manipulations (e.g., to clean the data), several extraction steps with different parameters and types of patterns (e.g., feature construction by means of constrained item sets followed by a classifying phase, association rule mining for different thresholds values and different objective measures of interestingness), and post-processing manipulations (e.g., elimination of redundancy in extracted patterns, crossing-over operations between patterns and

data like the search of transactions which are exceptions to frequent and valid association rules or the selection of misclassified examples with a decision tree). Looking for a tighter integration between data and patterns which hold in the data, Imielinski and Mannila have proposed in (Imielinski and Mannila, 1996) the concept of inductive database (IDB). In an IDB, ordinary queries can be used to access and manipulate data, while *inductive queries* can be used to generate (mine), manipulate, and apply patterns. KDD becomes an extended querying process where the analyst can control the whole process since he/she specifies the data and/or patterns of interests.

Therefore, the quest for query languages for IDBs is an interesting goal. It is actually a long-term goal since we still do not know which are the relevant primitives for data mining. In some sense, we still lack from a well-accepted set of primitives. It might recall the context at the end of the 60's before the Codd's relational algebra proposal. In some limited contexts, researchers have, however, designed data mining query languages. Data mining query languages can be used for specifying inductive queries on some pattern domains. They can be more or less coupled to standard query languages for data manipulation or pattern post processing manipulations.

More precisely, a data mining query language, should provide primitives to

(1) Select the data to be mined and pre-process these data,

(2) Specify the kind of patterns to be mined,

(3) Specify the needed background knowledge (as item hierarchies when mining generalized association rules),

(4) Define the constraints on the desired patterns, and

(5) Post process extracted patterns.

Furthermore, it is important that data mining query languages satisfy the closure property, i.e., the fact that the result of a query can be queried. Following a classical approach in database theory, it is also needed that the language is based on a well-defined (operational or even better declarative) semantics. It is the only way to make query languages that are not only "syntactical sugar" on top of some algorithms but true query languages for which query optimization strategies can be designed. Again, if we consider the analogy with SQL, relational algebra has paved the way towards query processing optimizers that are widely used today. Ideally, we would like to study containment or equivalence between mining queries as well.

Last but not the least, the evaluation of data mining queries is in general very expensive. It needs for efficient constraint-based data mining algorithms, the so-called solvers (De Raedt, 2003; Boulicaut and Jeudy, 2005). In other terms, data mining query languages are often based on primitives for which some more or less ad-hoc solvers are available. It is again typical of a situation where a consensus on the needed primitives is yet missing. So far, no language proposal is generic enough to provide support for broad kind applications during the whole KDD process. However, in the active field of association rule mining, some interesting query languages have been proposed.

Supporting Association Rule Mining Processes

In this context, data is considered as a multi set of transactions, i.e., sets of items. Frequent associations rules are built on frequent item sets (item sets which are subsets of a certain percent - age of the transactions). Many objective interestingness measures can inform about the quality of the extracted rules, the confidence measure being one of the most used. Importantly, many objective measures appear to be complementary: they enable to rank the rules according to different points of view. Therefore, it seems important to provide support for various measures, including the definition of new ones, e.g., application specific ones. When a KDD process is based on item sets or association rules, many operations have to be performed by means of queries. First, the language should allow manipulating and extracting source data. Typically, the raw data is not always available as transactional data. One of the typical problems concerns the transformation of numerical attributes into items (or Boolean properties). More generally, deriving

the transactional context to be mined from raw data can be a quite tedious task (e.g., deriving a transactional data set about WWW resources loading per session from raw WWW logs in a WWW Usage Mining application).

Some of these preprocessing are supported by SQL but a programming extension like PL/SQL is obviously needed. Then, the language should allow the user to specify a broad kind of constraints on the desired patterns (e.g., thresholds for the objective measures of interestingness, syntactical constraints on items which must appear or not in rule components). So far, the primitive constraints and the way to combine them is tightly linked with the kinds of constraints the underlying evaluation engine or solvers can process efficiently (typically anti-monotonic or succinct constraints). One can expect that minimal frequency and minimal confidence constraints are available. However, many other primitive constraints can be useful, including the ones based on aggregates (Ng *et al*., 1998) or closures (Jeudy and Boulicaut, 2002; Boulicaut, 2004).

Once rules have been extracted and materialized (e.g., in relational tables), it is important that the query language provides techniques to manipulate them. We can wish, for instance, to find a cover of a set of extracted rules (i.e., non redundant association rules based on closed sets (Bastide *et al*., 2000)), which requires to have subset operators, primitives to access bodies and heads of rules, and primitives to manipulate closed sets or other condensed representations of frequent sets (Boulicaut, 2004) and (Calders and Goethals, 2002).

Another important issue is the need for crossing-over primitives. It means that, for instance, we need simple way to select transactions that satisfy or do not satisfy a given rule. The so-called closure property is important. It enables to combine queries, to support the reuse of KDD scenarios, and it gives rise to opportunities for compiling schemes over sequences of queries (Boulicaut *et al*., 1999).

Finally, we could also ask for a support to pattern uses. In other terms, once relevant patterns have been stored, they are generally used by some software component. To the best of our knowledge, very few tools have been designed for this purpose (see (Imielinski *et al*., 1999) for an exception).

We can distinguish two major approaches in the design of data mining query languages. The first one assumes that all the required objects (data and pattern storage systems and solvers) are already embedded into a common system. The motivation for the query language is to provide more understandable primitives: the risk is that the query language provides mainly "syntactic sugar" on top of solvers. In that framework, if data are stored using a classical relational DBMS, it means that source tables are views or relations and that extracted patterns are stored using the relational technology as well. MSQL, DMQL and MINE RULE can be considered as representative of this approach. A second approach assumes that we have no predefined integrated systems and that storage systems are loosely coupled with solvers which can be available from different providers. In that case, the language is not only an interface for the analyst but also a facilitator between the DBMS and the solvers. It is the approach followed by OLE DB for DM (Microsoft). It is an API between different components that also provides a language for creating and filling extraction contexts, and then access them for manipulations and tests. It is primarily designed to work on top of SQL Server and can be plugged with different solvers provided that they comply the API standard.

Primitives

As stated earlier, a primitive is a low level parameter-set or command-set that represents a unit of mining task and which the mining system understands. Each user has different data mining task in mind. A data mining task can be specified in the form of a data mining query, which is input to the data mining system. The following primitives can be used in a data mining query:

- ***Task-relevant data:*** It is that portion of the whole database which needs to be processed. Suppose that a manager of HTC is in charge of sales in Chennai and Mumbai. He would like to study the buying trends of customers in these two cities respectively. Rather than mining the entire database, he can specify that only the data relating to customer purchases in Chennai and Mumbai need be retrieved, along with the related customer profile information. He can also specify attributes of interest to be considered in the mining process. These are referred to as relevant attributes. For example, if he is interested only in

studying possible relationships between, say, the items purchased and customer annual income and age, then the attributes name of the relation item, and salary and age of the relation customer, can be specified as the relevant attributes for mining.

- ***The kinds of knowledge to be mined:*** This primitive specifies the data mining functions to be performed. The function could be characterization, discrimination, association, classification, clustering, or evolution analysis.
- ***Background knowledge:*** Domain knowledge or background knowledge about may also be provided by the user. This knowledge is useful for guiding the knowledge discovery process and for evaluating the patterns found. The knowledge is generally represented in the form of concept hierarchies. They allow data to be mined at multiple levels of abstraction.
- ***Interestingness measures:*** These primitives are used to separate uninteresting patterns from knowledge. They may be used to guide the mining process or, after discovery, to evaluate the discovered patterns. Different kinds of knowledge may have different interestingness measures. For example, interestingness measures for association rules include support (the percentage of task–relevant data tuples for which the rule pattern appears} and confidence (an estimate of the strength of the implication of the rule). Rules whose support and confidence values are below user-specified thresholds are considered uninteresting.
- ***Presentation and visualization of discovered patterns:*** These primitives refer to the form in which discovered patterns are to be displayed or presented to the user. Users can choose from different forms for knowledge presentation, such as rules, tables, charts, graphs, decision trees, and cubes.

Student Activity 1

1. Write in details about the five primitives for specifying a data mining task.

Task-Relevant Data

Generally, a user is interested in only a subset of the database instead of the whole database. It is impractical to indiscriminately mine the entire database for reasons understood.

In case of a relational database, the set of task-relevant data can be collected using a relational query involving operations like selection, projection, join, and aggregation. This forms a "subtask" of the data mining task. The data collection process results in a new data relation, called the initial data relation. The initial data relation can be ordered or grouped according to the conditions specified in the query. The data may be cleaned or transformed prior to applying data mining analysis. The set of task-relevant data for data mining is called a minable view since virtual relations are called views in the field of databases.

The task-relevant data primitive can be specified for the data mining task of studying associations between items frequently purchased at HTC by customers in Chennai, with the following information:

- The name of the database or data warehouse to be used.
- The names of the tables or data cubes containing the relevant data.
- Conditions for selecting the relevant data.
- The relevant attributes or dimensions.

User may also specify that the data retrieved be grouped by certain attributes, such as "group by salary". Given this information, an SQL query can be used to retrieve the task-relevant data.

In a data warehouse, data are typically stored in a multidimensional database, known as a data cube, which can be implemented using a multidimensional array structure, a relational structure, or a combination of both. The set of task-relevant data can be specified by condition-based data filtering, slicing or dicing of the data cube. Slicing is extracting data for a given set of attributes while dicing is extracting data from intersection of several slices. Moreover, in a data mining query, the conditions provided for data selection can be at a level that is conceptually higher than

the data in the database or data warehouse. For example, a user may specify a selection on cities at HTC using the concept capital = "capital city of India" even though individual cities in the database may not be stored according to capital, but rather, at a lower conceptual, such as "Chennai"; "Patna"; or "Mumbai" etc. A concept hierarchy on city that specifies that "capital city of India" is at a higher concept level, composed of the lowerlevel concepts {"Chennai"; "Patna"}, can be used in the collection of the task-relevant data.

However, specification of the relevant attributes or dimensions is a difficult task for users. A user may have only a rough idea of what the interesting attributes for exploration might be. Furthermore, when specifying the data to be mined, the user may overlook additional relevant data having strong semantic links to them. For example, the sales of certain items may be closely linked to particular events such as Diwali or Holi, or to particular groups of customers, yet these factors may not be included in the general data analysis request. For such cases, mechanisms can be used that help give a more precise specification of the task-relevant data. These include functions to evaluate and rank attributes according to their relevance with respect to the operation specified. In addition, techniques that search for attributes with strong semantic ties can be used to enhance the initial data set specified by the user.

The Kind of Knowledge to be Mined

To determine the data mining function to be performed, it is important to specify the kind of knowledge to be mined. The kinds of knowledge include concept description (characterization and discrimination), association, classification, prediction, clustering, and evolution analysis.

Besides specifying the kind of knowledge to be mined for a given data mining task, the user can be more specific and provide pattern templates that all discovered patterns must match. These templates, or meta-patterns (also called metarules or metaqueries), can be used to guide the discovery process.

Consider a case. Suppose that a user studying the buying habits of HTC customers may choose to mine association rules of the form

$$P(X:\text{customer}, W) \wedge Q(X, Y) \Rightarrow \text{buys}(X, Z)$$

where X is a key of the customer relation; P and Q are predicate variables that can be instantiated to the relevant attributes or dimensions specified as part of the task-relevant data, and W, Y, and Z are object variables that can take on the values of their respective predicates for customers X.

The search for association rules is confined to those matching the given metarule, such as

$$\text{age}(X, \text{"30. . . 39"}) \wedge \text{salary}(X, \text{"5000. . . 7000"}) \Rightarrow \text{buys}(X, \text{"cell phone"})$$ [2.2%,60%] and

$$\text{occupation}(X, \text{"teacher"}) \wedge \text{age}(X, \text{"20 . . .29"}) \Rightarrow \text{buys}(X, \text{"computer"})$$ [1.4%,70%].

The former rule states that customers in their thirties, with an annual income of between 5000 and 7000 are likely (with 60% confidence) to purchase a cell phone, and such cases represent about 2.2% of the total number of transactions. The latter rule states that customers who are students and in their twenties are likely (with 70% confidence) to purchase a computer, and such cases represent about 1.4% of the total number of transactions.

Background Knowledge: Concept Hierarchies

As stated earlier, background knowledge is information about the domain to be mined that can be useful in the discovery process. A concept hierarchy defines a sequence of mappings from a set of low level concepts to higher-level, more general concepts. A concept hierarchy for the dimension location is shown in Figure 9.1, mapping low-level concepts (i.e., cities) to more general concepts (i.e., countries).

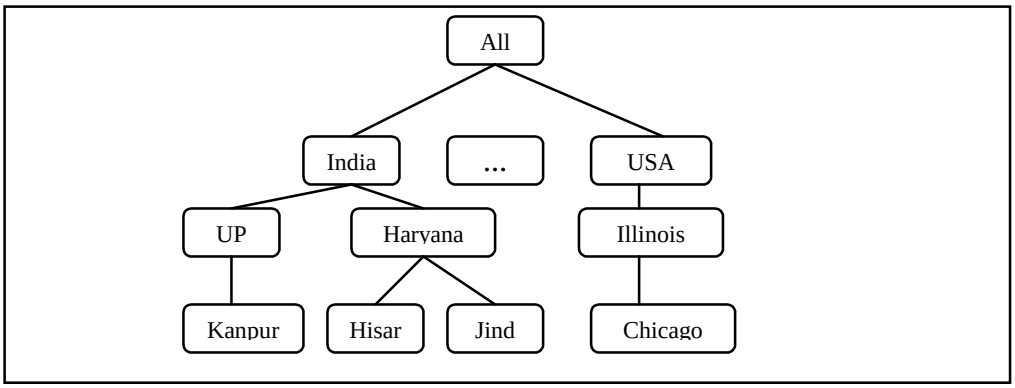


Figure 9.1

Notice that this concept hierarchy is represented as a set of nodes organized in a tree, where each node, in itself, represents a concept. A special node, all, is reserved for root of the tree. It denotes the most generalized value of the given dimension. If not explicitly shown, it is implied. This concept hierarchy consists of four levels. By convention, levels within a concept hierarchy are numbered from top to bottom, starting with level 0 for the all node. In our example, level 1 represents the concept country, while levels 2 and 3 represent the concepts province or state and city, respectively. The leaves of the hierarchy correspond to the dimension's raw data values (primitive level data). These are the most specific values, or concepts, of the given attribute or dimension. Although a concept hierarchy often defines a taxonomy represented in the shape of a tree, it may also be in the form of a general lattice or partial order.

Concept hierarchies can be provided by system users, domain experts, or knowledge engineers. The mappings are typically data or application-specific. Concept hierarchies can often be automatically discovered or dynamically refined based on statistical analysis of the data distribution.

There are four major types of concept hierarchies.

- ***Schema hierarchies:*** A schema hierarchy (or a schema-defined hierarchy) is a total or partial order among attributes in the database schema. Schema hierarchies may formally express existing semantic relationships between attributes. Typically, a schema hierarchy specifies a data warehouse dimension.

  Given the schema of a relation for address containing the attributes street, city, province/state, and country, we can define a location schema hierarchy by the following total order:

  street < city < province/state < country

  This means that street is at a conceptually lower level than city,. which is lower than province/state, which is conceptually lower than country. A schema hierarchy provides metadata information. Its specification in terms of a total or partial order among attributes is more concise than an equivalent definition that lists all instances of streets, provinces or states, and countries.

- ***Set-grouping hierarchies:*** A set-grouping hierarchy organizes values for a given attribute or dimension into groups of constants or range values. A total or partial order can be defined among groups. Set-grouping hierarchies can be used to refine or enrich schema-defined hierarchies, when the two types of hierarchies are combined. They are typically used for defining small sets of object relationships.

  A set-grouping hierarchy for the attribute age can be specified in terms of ranges, as in the following:

  $$\{young, middle\text{-}aged, old\} \subset all(age)$$

  $$\{20...39\} \subset young$$

  $$\{40...59\} \subset middle\_aged$$

  $$\{60...89\} \subset old$$

- ***Operation-derived hierarchies:*** An operation-derived hierarchy is based on operations specified by users, experts, or the data mining system. Operations can include the decoding of information encoded strings, information extraction from complex data objects, and data clustering.

  For instance, an e-mail address or a URL of the WWW may contain hierarchy information relating departments, universities (or companies), and countries. Decoding operations can be defined to extract such information in order to form concept hierarchies.

  The e-mail address excelbooks@vsnl.in gives the partial order login-name < ISP < country", forming a concept hierarchy for e-mail addresses. Similarly, the URL address http:// www.excelbooks/catalog/main/index.html can be decoded so as to provide a partial order that forms the base of a concept hierarchy for URLs.

- ***Rule-based hierarchies:*** A rule-based hierarchy occurs when either a whole concept hierarchy or a portion of it is defined by a set of rules and is evaluated dynamically based on the current database data and the rule definition.

  The following rules may be used to categorize HTC items as low_profit_margin items, medium_profit_margin items, and high-profit_margin items, where the profit margin of an item X is defined as the difference between the retail price and actual cost of X. Items having a profit margin of less than Rs.50 may be defined as low_profit_margin items, items earning a profit between Rs.50 and Rs.250 may be defined as medium_profit_rnargin items, and items earning a profit of more than Rs.250 may be defined as high-profit_rnargin items.

  $\text{low\_profit\_margin}(X) \Leftarrow \text{price}(X, P1) \wedge \text{cost}(X, P2) \wedge ((P1\text{-}P2) < 50)$

  $\text{medium\_profit\_margin}(X) \Leftarrow \text{price}(X, P1) \wedge \text{cost}(X, P2) \wedge ((P1\text{-}P2)$

  $\geq 50) \Leftarrow ((P1 - P2) \leq 50)$

  $\text{high\_profit\_rnargin}(X) \Leftarrow \text{price}(X, P1) \wedge \text{cost}(X, P2) \wedge ((P1\text{-}P2) > 250)$

Measures of Interestingness

Although specification of the task-relevant data and of the kind of knowledge to be mined may substantially reduce the number of patterns generated, a data mining process may still generate a large number of patterns. Typically, only a small fraction of these patterns will actually be of interest to the given user. Thus, users need to further confine the number of uninteresting patterns returned by the process. This can be achieved by specifying interestingness measures that estimate the simplicity, certainty, utility, and novelty of patterns.

Objective measures are based on the structure of patterns and the statistics underlying them. In general, each measure is associated with a threshold that can be controlled by the user. Rules that do not meet the threshold are considered uninteresting, and hence are not presented to the user as knowledge.

- ***Simplicity:*** A factor contributing to the interestingness of a pattern is the pattern's overall simplicity for human comprehension. Objective measures of pattern simplicity can be viewed as functions of the pattern structure, defined in terms of the pattern size in bits, or the number of attributes or operators appearing in the pattern. For example, the more complex the structure of a rule is, the more difficult it is to interpret, and, hence, the less interesting it is likely to be.

  Rule length is a simplicity measure. For rules expressed in conjunctive normal form (i.e., as a set of conjunctive predicates), rule length is typically defined as the number of conjuncts in the rule. Association, discrimination, or classification rules whose lengths exceed a user-defined threshold can be considered uninteresting. For patterns expressed as decision trees, simplicity may be a function of the number of tree leaves or tree nodes.

- ***Certainty:*** Every discovered pattern should have a measure of certainty associated with it that assesses the validity or "trustworthiness" of the pattern. A certainty measure for association rules of the form "$A \Rightarrow B$", where A and B are sets of items, is confidence. Given a set of task-relevant data tuples (or transactions in a transaction database) the confidence of "$A \Rightarrow B$" is defined as

confidence(A⇒B)= (number of tuples_containing_A_and_B) / (number oftuples_containing_A)

Suppose the set of task-relevant data consists of transactions of items purchased from the computer department of HTC. A confidence of 85% for the association rule buys(X, "computer") ⇒ buys(X, "electric fan") means that 85% of all customers who purchased a computer also bought electric fan. A confidence value of 100%, or 1, indicates that the rule is always correct on the data analyzed. Such rules are called exact.

- ***Utility:*** The potential usefulness of a pattern is a factor defining its interestingness. It can be estimated by a utility function, such as support. The support of an association pattern refers to the percentage of task-relevant data tuples (or transactions) for which the pattern is true. For association rules of the form "A⇒B" where A and B are sets of items, it is defined as

  support(A => B)= (number of tuples containing A and B)/(number of all tuples)

  Association rules that satisfy both a user-specified minimum confidence threshold and user-specified minimum support threshold are referred to as strong association rules and are considered interesting. Rules with low support likely represent noise, or rare or exceptional cases.

- ***Novelty:*** Novel patterns are those that contribute new information or increased performance to the given pattern set. For example, a data exception may be considered novel in that it differs from that expected based on a statistical model or user beliefs. Another strategy for detecting novelty is to remove redundant patterns. If a discovered rule can be implied by another rule that is already in the knowledge base or in the derived rule set, then either rule should be re-examined in order to remove the potential redundancy.

Presentation and Visualization of Discovered Patterns

For data mining to be effective, data mining systems should be able to display the discovered patterns in multiple forms, such as rules, tables, crosstabs (cross-tabulations), pie or bar charts, decision trees, cubes, or other visual representations. Allowing the visualization of discovered patterns in various forms can help users with different backgrounds to identify patterns of interest and to interact or guide the system in further discovery. A user should be able to specify the forms of presentation to be used for displaying the discovered patterns.

9.3 DATA MINING QUERY LANGUAGES

Data mining query languages can be designed to support feature that makes the data mining interactive. The importance of the design of a good data mining query language can also be seen from observing the history of relational database systems. Relational database systems have dominated the database market for decades. The standardization of relational query languages, which occurred at the early stages of relational database development, is widely credited for the success of the relational database field. Although each commercial relational database system has its own graphical user interface, the underlying core of each interface is a standardized relational query language. The standardization of relational query languages provided a foundation on which relational systems were developed and evolved. It facilitated information exchange and technology transfer, and promoted commercialization and wide acceptance of relational database technology.

Designing a comprehensive data mining language is challenging because data mining covers a wide spectrum of tasks, from data characterization to mining association rules, data classification, and evolution analysis. Each task has different requirements. The design of an effective data mining query language requires a deep understanding of the power, limitation, and underlying mechanisms of the various kinds of data mining tasks.

Based on the primitives, we design a query language for data mining called DMQL (Data Mining Query Language). DMQL allows the ad hoc mining of several kinds of knowledge from relational databases and data warehouses at multiple levels of abstraction. The language adopts an SQL-like syntax, so that it can easily be integrated with the relational query language SQL.

Syntax for Task-Relevant Data Specification

The first step in defining a data mining task is the specification of the task-relevant data, that is, the data on which mining is to be performed. This involves specifying the database and tables or data warehouse containing the relevant data, conditions for selecting the relevant data, the relevant attributes or dimensions for exploration, and instructions regarding the ordering or grouping of the data retrieved. DMQL provides clauses for the specification of such information, as follows:

- ***Use database ⟨database_name⟩ or use data warehouse ⟨data_warehousename⟩:*** The use clause directs the mining task to the database or data warehouse specified.
- ***From ⟨relation(s)/cube(s) ⟩ [where ácondition⟩]:*** The from and where clauses respectively specify the database tables or data cubes involved, and the conditions defining the data to be retrieved.
- ***In relevance to ⟨attribute_or_dimemsion_list⟩:*** This clause lists the attributes or dimensions for exploration.
- ***Order by ⟨ordeclist⟩:*** The order by clause specifies the sorting order of the task-relevant data.
- ***Group by ⟨groupingjist⟩:*** The group by clause specifies criteria for grouping the data.
- ***Having ⟨condition⟩:*** The having clause specifies the condition by which groups of data are considered relevant.

Let us consider an example on DMQL. This example shows how to use DMQL to specify the task-relevant data for the mining of associations between items frequently purchased at HTC by customers located in Chennai, with respect to customer salary and age. In addition, the user specifies that the data are to be grouped by date. The data are retrieved from a relational database.

```
use database HTC
in relevance to I.name, I.price, C.salary, C.age
from customer C, item I, pur P, items_sold S
where I.item_ID=S.item_ID and S.trans_ID=P.trans_ID and P.custID=
C.cust_ID and C.city = “Chennai”
group by P.date
```

Syntax for Specifying the Kind of Knowledge to be Mined

The áMine_Knowledge_Specificationñ statement is used to specify the kind of knowledge to be mined. In other words, it indicates the data mining functionality to be performed. Its syntax is defined below for characterization, discrimination, association, and classification.

Characterization:

```
⟨Mine_Knowledge_Specification⟩ ::=
        mine characteristics [as ⟨pattern_name⟩]
        analyze ⟨measure(s)⟩
```

This specifies that characteristic descriptions are to be mined. The analyze clause, when used for characterization, specifies aggregate measures, such as count, sum, or count% (percentage count, i.e., the percentage of tuples in the relevant data set with the specified characteristics). These measures are to be computed for each data characteristic found.

The following specifies that the kind of knowledge to be mined is a characteristic description describing customer purchasing habits. For each characteristic, the percentage of task-relevant tuples satisfying that characteristic is to be displayed.

mine characteristics as customerPurchasing

analyze count%

Discrimination:

⟨Mine_Knowledge_Specification⟩ ::=

mine comparison [as ⟨pattern_name⟩]

for ⟨target_class⟩ where ⟨target_condition⟩

{versus ⟨contrast_class_i⟩ where ⟨contrast_condition_i⟩}

analyze ⟨measure(s)⟩

The above statement specifies that discriminant descriptions are to be mined. These descriptions compare a given target class of objects with one or more other contrasting classes. Hence, this kind of knowledge is referred to as a comparison. As for characterization, the analyze clause specifies aggregate measures, such as count, sum, or count%, to be computed and displayed for each description.

The user may define categories of customers, and then mine descriptions of each category. For instance, a user may define bigSpenders as customers who purchase Items that cost Rs.100 or more on average, and budgetSpenders as customers who purchase items at less than Rs.100 on average. The mining of discriminant descriptions for customers from each of these categories can be specified in DMQL as:

mine comparison as purchaseGroups

for bigSpenders where avg(I.price) ≥ 100

versus budgetSpenders where avg(I.price)<100

analyze count

Association:

⟨MincKnbwledge_Specification⟩ ::=

mine associations [as ⟨pattern_name⟩]

[matching ⟨metapattern⟩

This specifies the mining of patterns of association. When specifying association mining, the user has the option of providing templates (also known as metapatterns or metarules) with the matching clause. The metapatterns can be used to focus the discovery towards the patterns that match the given metapatterns, thereby enforcing additional syntactic constraints for the mining task. In addition to providing syntactic constraints, the metapatterns represent data hunches or hypotheses that the user finds interesting for investigation. Mining with the use of metapatterns, or metarule-guided mining, allows additional flexibility for ad hoc rule mining. While metapatterns may be used in the mining of other forms of knowledge, they are most useful for association mining due to the vast number of potentially generated associations.

Syntax for Concept Hierarchy Specification

Concept hierarchies allow the mining of knowledge at multiple levels of abstraction. In order to accommodate the different viewpoints of users with regard to the data, there maybe more than one concept hierarchy per attribute or dimension. For instance, some users may prefer to organize branch locations by provinces and states, while others may prefer to organize them according to languages used. In such cases, a user can indicate which concept hierarchy is to be used with the statement use hierarchy ⟨hierarchy_name⟩ for ⟨attribute_or_dimension⟩ Otherwise, a default hierarchy per attribute or dimension is used.

Definition of a schema hierarchy: Earlier, we defined a schema hierarchy for a relation address as the total order street < city < province/state < country. This can be defined in the data mining

define hierarchy location_hierarchy on address as

[street, city, province/state, country]

The ordering of the listed attributes is important. In fact, a total order is defined that specifies that street is conceptually one level lower than city, which is in turn conceptually one level lower than province/state, and so on.

Definition of a set-grouping hierarchy: The set-grouping hierarchy for age can be defined in terms of ranges as follows, where, by convention, the most general concept, all, is placed at the root of the hierarchy (i.e., at level 0).

define hierarchy age_hierarchy for age on customer as

level1: {young, middle_aged, old} < level0: all

level2: {20 . . . 39} < level1: young

level2: {40 . . . 59} < level1: middle_aged

level2: {60 . . . 89} < level1: old

Syntax for Interestingness Measure Specification

The user can help control the number of uninteresting patterns returned by the data mining system by specifying measures of pattern interestingness and their corresponding thresholds. Interestingness measures include the confidence, support, noise, and novelty. Interestingness measures and thresholds can be specified by the user with the statement

with [⟨interescmeasure_name⟩] threshold = ⟨threshold_value⟩

In mining association rules, a user can confine the rules to be found by specifying a minimum support and minimum confidence threshold of 5% and 70%, respectively, with the statements

with support threshold = 5%

with confidence threshold = 70%

Syntax for Pattern Presentation and Visualization Specification

Data mining query language needs syntax that allows users to specify the display of discovered patterns in one or more forms, including rules, tables, crosstabs, pie or bar charts, decision trees, cubes, curves, or surfaces. We define the DMQL display statement for this purpose:

display as ⟨result_form⟩

where the (result_form) could be any of the knowledge presentation or visualization forms listed above.

Interactive mining should allow the discovered patterns to be viewed at different concept levels or from different angles. This can be accomplished with roll-up and drill-down operations. Patterns can be rolled up, or viewed at a more general level, by climbing up the concept hierarchy of an attribute or dimension (replacing lower-level concept values by higher-level values). Generalization can also be performed by dropping attributes or dimensions. For example, suppose that a pattern contains the attribute city. Given the location hierarchy city < province/state < country < continent, then dropping the attribute city from the patterns will generalize the data to the next highest level attribute, province/state. Patterns can be drilled down on, or viewed at a less general level, by stepping down the concept hierarchy of an attribute or dimension. Patterns can also be made less general by adding attributes or dimensions to their description. The attribute added must be one of the attributes listed in the in relevance to clause for task-relevant specification. The user can alternately view the patterns at different levels of abstractions with the use of the following DMQL syntax:

⟨Multilevel_Manipulation⟩ ::= roll up on ⟨attribute_or_dimension⟩

|drill down on ⟨attribute_or_dimension⟩

|add ⟨attribute_or_dimension⟩

|drop ⟨attribute_or_dimension⟩

Suppose that descriptions are mined based on the dimensions location, age, and salary. Users can "roll up on location" or "drop age" to generalize the discovered patterns.

Standardization of Data Mining Primitives

Besides DMQL there have been research efforts to design other data mining languages and industry efforts to standardize data mining primitives and languages. Here we introduce a few examples.

MSQL is a data mining query language, proposed by Imielinski and Virmani [IV99]. The language uses SQL-like syntax and SQL primitives including sorting and group-by. Since an enormous number of rules can be generated in data mining, MSQL provides primitives like, GetRules and SelectRules for both rule generation and rule selection. It treats data and rules uniformly and, therefore, optimization can be explored by either performing selective, query-based rule generation from data, or manipulating or querying the generated set of rules.

Other research efforts on data mining language design include the MINE RULE operator, proposed by Meo, Psaila, and Ceri [MPC96]. It follows SQL-like syntax and serves as rule generation queries for mining association rules. Another proposal, by Tsur, Ullman, Abitboul, Clifton, et al. [TUA +98], uses Datalog syntax to express query flocks, which facilitates both mining and testing rules.

Microsoft proposed a data mining language called OLE DB for Data Mining (DM). This is a notable effort towards the standardization of data mining language primitives. Having a standard data mining language will help to strengthen the data mining industry by facilitating the development of data mining platforms and of data mining systems, and the sharing of data mining results.

OLE DB for DM, together with OLE DB and OLE DB for OLAP, constitute three important steps by Microsoft towards the standardization of database, database warehousing, and data mining languages. The specifications of OLE DB for DM cover the primitives for creation and use of several important data mining modules, including association, predictive modeling (classification and prediction), and clustering.

CRISP-DM (CRoss-Industry Standard Process for Data Mining) is another standardization effort related to data mining.

Student Activity 3

1. Write DMQL for:

   a. Task-Relevant Data Specification

   b. The kind of knowledge to be mined

   c. Concept Hierarchy Specification

   d. Interestingness Measure Specifications

   e. Pattern Presentation and Visualization Specification

9.4 SYSTEM ARCHITECTURES

With popular and diverse applications of data mining, it is expected that a good variety of data mining systems will be designed and developed in future years. Although rich and powerful data mining functions form the core of a data mining system, like most software systems, the architecture and design of a data mining system is critically important. A good system architecture will facilitate the system to make best use of the software environment, accomplish data mining tasks in an efficient and timely manner, interoperate and exchange information with other information systems,

be adaptable to users' diverse requirements, and evolve with time.

With decades of research and development in the database and information industry, database systems and data warehouse systems have become the mainstream information systems. Tremendous amounts of data and information have been stored and/or integrated in such systems. Moreover, comprehensive information processing and data analysis infrastructures have been or will be continuously and systematically constructed surrounding database systems and data warehouses. These include accessing, integration, consolidation, and transformation of multiple, heterogeneous databases, OOBC/OLE DB connections, Web-accessing and service facilities, and reporting and OLAF analysis tools.

Under this situation, a critical question in the design of a data mining system becomes whether we should couple or integrate a data mining (DM) system with a database (DB) system and/or a data warehouse (DW) system, and if we should, how to do it correctly. To answer these questions, we need to examine the possible ways of coupling or integrating a DM system and a DB/DW system. Based on different architecture designs, a DM system can be integrated with a DB/DW system using the following coupling schemes: no coupling, loose coupling, semi-tight coupling, and tight coupling. Let us examine these one by one.

- ***No coupling:*** No coupling means that a DM system will not utilize any function of a DB or DW system. It may fetch data from a particular source (such as a file system), process data using some data mining algorithms, and then store the mining results in another file.

  Such a system, though simple, suffers from several drawbacks. First, a DB system provides a great deal of flexibility and efficiency at storing, organizing, accessing, and processing data. Without using a DB/DW system, a DM system may spend a substantial amount of time finding, collecting, cleaning, and transforming data. In DB and/or DW systems, data tend to be well organized, indexed, cleaned, integrated, or consolidated, so that finding the task-relevant, high-quality data becomes an easy task. Second, there are many tested, scalable algorithms and data structures implemented in DB and DW systems. It is feasible to realize efficient, scalable implementations using such systems. Moreover, most data have been or will be stored in DB/DW systems. Without any coupling of such systems, a DM system will need to use other tools to extract data, making it difficult to integrate such a system into an information processing environment. Thus, no coupling represents a poor design.

- ***Loose coupling:*** Loose coupling means that a DM system will use some facilities of a DB or DW system, fetching data from a data repository managed by these systems, performing data mining, and then storing the mining results either in a file or in a designated place in a database or data warehouse.

  Loose coupling is better than no coupling since it can fetch any portion of data stored in databases or data warehouses by using query processing, indexing, and other system facilities. It incurs some advantages of the flexibility, efficiency, and other features provided by such systems. However, many loosely coupled mining systems are main memory-based. Since mining itself does not explore data structures and query optimization methods provided by DB or DW systems, it is difficult for loose coupling to achieve high scalability and good performance with large data sets.

- ***Semi-tight coupling:*** Semi-tight coupling means that besides linking a DM system to a DB/DW system, efficient implementations of a few essential data mining primitives (identified by the analysis of frequently encountered data mining functions) can be provided in the DB/DW system. These primitives can include sorting, indexing, aggregation, histogram analysis, multi way join, and pre-computation of some essential statistical measures, such as sum, count, max, min, standard deviation, and so on. Moreover, some frequently used intermediate mining results can be pre-computed and stored in the DB/DW system. Since these intermediate mining results are either pre-computed or can be computed efficiently, this design will enhance the performance of a DM system.

- ***Tight coupling:*** Tight coupling means that a DM system is smoothly integrated into the DB/DW system. The data mining subsystem is treated as one functional component of an information system. Data mining queries and functions are optimized based on mining

query analysis, data structures, indexing schemes, and query processing methods of a DB or DW system. With further technology advances, DM, DB, and DW systems will evolve and integrate together as one information system with multiple functionalities. This will provide a uniform information processing environment.

9.5 SUMMARY

1. Many data mining algorithms enable to extract different types of patterns from data (e.g., local patterns like item sets and association rules, models like classifiers).
2. In the prolific field of association rule mining, different proposals of query languages have been made to support the more or less declarative specification of both data and pattern manipulations.
3. Evaluation of data mining queries is in general very expensive.
4. Once rules have been extracted and materialized (e.g., in relational tables), it is important that the query language provides techniques to manipulate them.
5. A schema hierarchy is a total or partial order among attributes in the database schema.
6. For data mining to be effective, data mining system should be able to display the discovered patterns in multiple forms, such as rules, tables, cross tabs, pie or bar chards, decision tree or other visual representations.
7. Concept Hierarchy allow the mining of knowledge at multiple levels of abstraction.
8. Microsoft proposed a data mining language called OLE DB for data mining.
9. No Coupling means that a DM system will not utilize any function of a DB or DW system.
10. Loose coupling means that a DM system will use some facilities of a DB or DW system.

9.6 KEYWORDS

Primitives: A primitive is a low level parameter-set or command-set that represents a unit of mining task and which the mining system understands.

Minable view: The set of task-relevant data for data mining is called a minable view.

Tight coupling: It means that a DM system is smoothly integrated into the DB/DW system.

9.7 REVIEW QUESTIONS

1. Explain the importance of concept hierarchies in data mining with the help of suitable example.
2. List down various standardization of data mining primitives.
3. Describe the differences between the following architecture for the integration of a data mining system with a database or data warehouse system: no coupling, loose coupling, semi tight coupling and tight coupling.

9.8 FURTHER READINGS

Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurasamy, "Advances in Knowledge Discovery and Data Mining", AAAI Press/ The MIT Press, 1996.

J. Ross Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1993.

Michael Berry and Gordon Linoff, "Data Mining Techniques (For Marketing, Sales, and Customer Support), John Wiley & Sons, 1997.

Sholom M. Weiss and Nitin Indurkhya, "Predictive Data Mining: A Practical Guide", Morgan Kaufmann Publishers, 1998.

Alex Freitas and Simon Lavington, "Mining Very Large Databases with Parallel Processing", Kluwer Academic Publishers, 1998.

Adriaans, P. and Zantige, D. Data Mining, Harlow, UK: Addison-Wesley, 1996

Berry, M.J.A. and Linoff, G., Data Mining Techniques for Marketing, Sales, and Customer Support, New York, NY: John Wiley and Sons, 1997

Bishop, C. Neural Networks for Pattern Recognition, Oxford, UK: Clarendon Press, 1995

Breiman, L., Fredman, J., Olshen, R.A., and Stone, C.J., Classification and Regression Trees, Monterey, CA: Wadsworth & Brooks, 1984

Chester, M., Neural Networks: A Tutorial, Englwood cliffs, NJ, Prentice Hall, 1993

Cressie, N., Statistics for Spatial Data, Revised Edition, Wiley, New York, 1993

Fayyad, U.M.; Piatestsky-Shapiro G.; Smyth D.; and Uthurusamy R., Advances in Knowledge Discovery and Data Mining, Cambridge, MA: AAAI Press/MIT Press, 1996

Fukunaga, K., Introduction to Statistical Pattern Recognition, Academic Press, 1990

UNIT

10

CONCEPT DESCRIPTION

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Define Descriptive Data Mining.
- Know about Data Generalization.
- Describe Analytical Characterization.
- Know about methods of attribute relevance Analysis.
- Make a study on Mining clause comparision.
- Describe different Statistical Measures.

U N I T S T R U C T U R E

10.1 INTRODUCTION

The simplest kind of descriptive data mining is concept description. A concept usually refers to a collection of data such as frequent_buyers, graduate_student, and so on. As a data mining task, concept description is not a simple enumeration of the data. Instead, concept description generates descriptions for characterization and comparison of the data. It is some time called class description, when the concept to be described refers to a class of objects. Characterization provides a concise and succinct summarization of the given collection of data, while concept or class comparison provides comparing two or more collection of data. Since concept description involves both characterization and comparison.

10.2 CHARACTERIZATION AND GENERALIZATION

Data and objects in databases often contain detailed information at primitive concept levels. For example, the item relation in a sales database may contain attributes describing low-level item information such as item_lD, name, brand, category, supplier, place_made, and price. It is useful to be able to summarize a large set of data and present it at a high conceptual level. For example, summarizing a large set of items relating to Diwali season sales provides a general description of such data, which can be very helpful for sales and marketing managers. This requires an important functionality in data mining: data generalization.

Data generalization is a process that abstracts a large set of task-relevant data in a database from a relatively low conceptual level to higher conceptual levels. Methods for the efficient and flexible generalization of large data sets can be categorized according to two approaches: (1) the data cube (or OLAP) approach and (2) the attribute-oriented induction approach. The data cube approach has already been described. In this section, we describe the attribute-oriented induction approach.

Attribute-Oriented Induction

The attribute-oriented induction (AOI) approach to data generalization and summarization-based characterization was first proposed in 1989, a few years prior to the introduction of the data cube approach. The data cube approach can be considered as a data warehouse-based, precomputation-oriented, materialized-view approach. It performs off-line aggregation before an OLAP or data mining query is submitted for processing. On the other hand, the attribute-oriented induction approach, at least in its initial proposal, is a relational database query-oriented, generalization-based, on-line data analysis technique. However, there is no inherent barrier distinguishing the two approaches based on on-line aggregation versus off-line precomputation. Some aggregations in the data cube can be computed on-line, while off-line precomputation of multidimensional space can speed up attribute-oriented induction as well.

The general idea of attribute-oriented induction is to first collect the task-relevant data using a relational database query and then perform generalization based on the examination of the number of distinct values of each attribute in the relevant set of data. The generalization is performed by either attribute removal or attribute generalization. Aggregation is performed by merging identical generalized tuples and accumulating their respective counts. This reduces the size of the generalized data set. The resulting generalized relation can be mapped into different forms for presentation to the user, such as charts or rules.

Suppose that a user would like to describe the general characteristics of graduate students, in the Hisar-University database, given the attributes name, gender, subject, birth_place, birth_date, residence, phoneNo (telephone number), and marks. A data mining query for this characterization can be expressed in the data mining query language DMQL as follows:

```
use Hisar_University_DB

mine characteristics as "Science_Students"

in relevance to name, gender, subject, birth_place, birth_date, residence,
phoneNo, marks from student

where status is "graduate"
```

What does the 'where status in "graduate"' clause mean? This where clause implies that a concept hierarchy exists for the attribute status. Such a concept hierarchy organizes primitive-level data values for status, such as "M.Sc."; "M.A."; "M.B.A."; "Ph.D." into higher conceptual levels, such as "graduate" and "undergraduate". This use of concept hierarchies does not appear in traditional relational query languages, yet is a common feature in data mining query languages.

Transforming a data mining query to a relational query: The data mining query presented above is transformed into the following relational query for the collection of the task-relevant set of data:

```
use Hisar_University_DB

select name, gender, subject, birth_place, birth_date, residence, phoneNo,
marks from student

where status in {"M.Sc:; "M.A."; "Ph.D."}
```

The transformed query is executed against the relational database, Hisar_University_DB; and returns the data in a table. This table is called the (task-relevant) initial working relation. It is the data on which induction will be performed.

Now that the data are ready for attribute-oriented induction, how is attribute-oriented induction

performed? The essential operation of attribute-oriented induction is data generalization, which can be performed in either of two ways on the initial working relation: attribute removal and attribute generalization.

Attribute removal is based on the following rule: If there is a large set of distinct values for an attribute of the initial working relation, but either (1) there is no generalization operator on the attribute (e.g., there is no concept hierarchy defined for the attribute), or (2) its higher-level concepts are expressed in terms of other attributes, then the attribute should be removed from the working relation.

Attribute generalization is based on the following rule: If there is a large set of distinct values for an attribute in the initial working relation, and there exists a set of generalization operators on the attribute, then a generalization operator should be selected and applied to the attribute. This rule is based on the following reasoning. Use of a generalization operator to generalize an attribute value within a tuple, or rule, in the working relation will make the rule cover more of the original data tuples, thus generalizing the concept it represents. This corresponds to the generalization rule known as climbing generalization trees in learning from examples, or concept tree ascension.

There are many possible ways to control a generalization process. We will describe two common approaches.

The first technique, called attribute generalization threshold control, either sets one generalization threshold for all of the attributes, or sets one threshold for each attribute. If the number of distinct values in an attribute is greater than the attribute threshold, further attribute removal or attribute generalization should be performed. Data mining systems typically have a default attribute threshold value (typically ranging from 2 to 8) and should allow experts and users to modify the threshold values as well. If a user feels that the generalization reaches too high a level for a particular attribute, the threshold can be increased. This corresponds to drilling down along the attribute. Also, to further generalize a relation, the user can reduce the threshold of a particular attribute, which corresponds to rolling up along the attribute.

The second technique, called generalized relation threshold control, sets a threshold for the generalized relation. If the number of (distinct) tuples in the generalized relation is greater than the threshold, further generalization should be performed. Otherwise, no further generalization should be performed. Such a threshold may also be preset in the data mining system (usually within a range of 10 to 30), or set by an expert or user, and should be adjustable. For example, if a user feels that the generalized relation is too small, she can increase the threshold, which implies drilling down. Otherwise, to further generalize a relation, she can reduce the threshold, which implies rolling up.

These two techniques can be applied in sequence: first apply the attribute threshold control technique to generalize each attribute, and then apply relation threshold control to further reduce the size of the generalized relation. No matter which generalization control technique is applied, the user should be allowed to adjust the generalization thresholds in order to obtain interesting concept descriptions.

In many database-oriented induction processes, users are interested in obtaining quantitative or statistical information about the data at different levels of abstraction. Thus, it is important to accumulate count and other aggregate values in the induction process. Conceptually, this is performed as follows. A special measure, or numerical attribute, that is associated with each database tuple is the aggregate function, count. Its value for each tuple in the initial working relation is initialized to 1. Through attribute removal and attribute generalization, tuples within the initial working relation may be generalized, resulting in groups of identical tuples. In this case, all of the identical tuples forming a group should be merged into one tuple. The count of this new, generalized tuple is set to the total number of tuples from the initial working relation that are represented by (i.e., were merged into) the new generalized tuple.

The data cube implementation of attribute-oriented induction can be performed in two ways.

Construct a data cube on-the-fly for the given data mining query: This method constructs a data cube dynamically based on the task-relevant set of data. This is desirable if either the task - relevant data set is too specific to match any predefined data cube, or it is not very large. Since such a data cube is computed only after the query is submitted, the major motivation for

constructing such a data cube is to facilitate efficient drill-down analysis. With such a data cube, drilling down below the level of the prime relation will simply require retrieving data from the cube, or performing minor generalization from some intermediate level data stored in the cube instead of generalization from the primitive-level data. This will speed up the drill-down process. However, since the attribute oriented data generalization involves the computation of a query-related data cube, it may involve more processing than simple computation of the prime relation and thus increase the response time. A balance between the two may be struck by computing a cube-structured "subprime" relation in which each dimension of the generalized relation is a few levels deeper than the level of the prime relation. This will facilitate drilling down to these levels with a reasonable storage and processing cost, although further drilling down beyond these levels will still require generalization from the primitive-level data. Notice that such further drilling down is more likely to be localized, rather than spread out over the full spectrum of the cube.

Use a predefined data cube: An alternative method is to construct a data cube before a data mining query is posed to the system, and use this predefined cube for subsequent data mining. This is desirable if the granularity of the task-relevant data can match that of the predefined data cube and the set of task-relevant data is quite large. Since such a data cube is precomputed, it facilitates attribute relevance analysis, attribute-oriented induction, slicing and dicing, roll-up, and drill-down. The cost one must pay is the cost of cube computation and the nontrivial storage overhead. A balance between the computation/storage overheads and the accessing speed may be attained by precomputing a selected set of all of the possible materializable cuboids.

10.3 ANALYTICAL CHARACTERIZATION

Measures of attribute relevance analysis can be used to help identify irrelevant or weakly relevant attributes that can be excluded from the concept description process. The incorporation of this preprocessing step into class characterization or comparison is referred to as analytical characterization or analytical comparison, respectively.

The first limitation of class characterization for multidimensional data analysis in data warehouses and OLAP tools is the handling of complex objects. The second limitation is the lack of an automated generalization process.

Methods should be introduced to perform attribute (or dimension) relevance analysis in order to filter out statistically irrelevant or weakly relevant attributes, and retain or even rank the most relevant attributes for the descriptive mining task at hand. Class characterization that includes the analysis of attribute/dimension relevance is called analytical characterization. Class comparison that includes such analysis is called analytical comparison.

Intuitively, an attribute or dimension is considered highly relevant with respect to a given class if it is likely that the values of the attribute or dimension may be used to distinguish the class from others. For example, it is unlikely that the color of an automobile can be used to distinguish expensive from cheap cars, but the model, make, style, and number of cylinders are likely to be more relevant attributes. Moreover, even within the same dimension, different levels of concepts may have dramatically different powers for distinguishing a class from others. For example, in the birth_date dimension, birth_day and birth_month are unlikely to be relevant to the salary of employees. However, the birth_decade (i.e., age interval) may be highly relevant to the salary of employees. This implies that the analysis of dimension relevance should be performed at multilevels of abstraction, and only the most relevant levels of a dimension should be included in the analysis.

Methods of Attribute Relevance Analysis

There have been many studies in machine learning; statistics, fuzzy and rough set theories, and so on, on attribute relevance analysis. The general idea behind attribute relevance analysis is to compute some measure that is used to quantify the relevance of an attribute with respect to a given class or concept. Such measures include information gain, the Gini index, uncertainty, and correlation coefficients.

An introduce to information gain analysis technique is given here. Let S be a set of training samples, where the class label of each sample is known. Each sample is in fact a tuple. One attribute is used to determine the class of the training samples. For instance, the attribute status

can be used to define the class label of each sample as either "graduate" or "undergraduate". Suppose that there are m classes. Let S contain S_i samples of class C_i, for i = 1, . . . , m. An arbitrary sample belongs to class C_i with probability s_i/s, where s is the total number of samples in set S. The expected information needed to classify a given sample is

$$I(s_1, s_2, ..., s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

An attribute A with values $\{a_1, a_2, ...a_v\}$ can be used to partition S into the subsets $\{S_1, S_2, \ldots, S_v\}$, where S_j contains those samples in S that have value a_j of A. Let S_j contain s_{ij} samples of class C_i. The expected information based on this partitioning by A is known as the entropy of A. It is the weighted average:

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + ...+ s_{mj}}{s} I\left(s_{1j}, ..., s_{mj}\right).$$

The information gain obtained by this partitioning on A is defined by

$$Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A).$$

In this approach to relevance analysis, we can compute the information gain for each of the attributes defining the samples in S. The attribute with the highest information gain is considered the most discriminating attribute of the given set. By computing the information gain for each attribute, we therefore obtain a ranking of the attributes. This ranking can be used for relevance analysis to select the attributes to be used in concept description.

Attribute relevance analysis for concept description is performed as follows:

- ***Data collection:*** Collect data for both the target class and the contrasting class by query processing. For class comparison, both the target class and the contrasting class are provided by the user in the data mining query. For class characterization, the target class is the class to be characterized, whereas the contrasting class is the set of comparable data that-are not in the target class.

- ***Preliminary relevance analysis using conservative AOI:*** This step identifies a set of dimensions and attributes on which the selected relevance measure is to be applied. Since different levels of a dimension may have dramatically different relevance with respect to a given class, each attribute defining the conceptual levels of the dimension should be included in the relevance analysis in principle. Attribute-oriented induction (AOI) can be used to perform some preliminary relevance analysis on the data by removing or generalizing attributes having a very large number of distinct values (such as name and phone number). Such attributes are unlikely to be found useful for concept description. To be conservative, the AOI performed here should employ attribute generalization thresholds that are set reasonably large so as to allow more (but not all) attributes to be considered in further relevance analysis by the selected measure. The relation obtained by such an application of AOI is called the candidate relation of the mining task.

- ***Remove irrelevant and weakly relevant attributes using the selected relevance analysis measure:*** Evaluate each attribute in the candidate relation using the selected relevance analysis measure. The relevance measure used in this step may be built into the data mining system or provided by the user. For example, the information gain measure described above may be used. The attributes are then sorted (i.e., ranked) according to their computed relevance to the data mining task.

- ***Generate the concept description using AOI:*** Perform AOI using a less conservative set of attribute generalization thresholds. If the descriptive mining task is class characterization, only the initial target class working relation is included here. If the descriptive mining task is class comparison, both the initial target class working relation and the initial contrasting class working relation are included.

If the mined concept descriptions involve many attributes, analytical characterization should be performed. This procedure first removes irrelevant or weakly relevant attributes prior to performing generalization.

Suppose that we want to mine the general characteristics describing graduate students at Hisar University using analytical characterization. Given are the, attributes name, gender, course, birth_place, birth_date, phoneNo, and marks.

In Step 1, the target class data are collected, consisting of the set of graduate students. Data for a contrasting class are also required in order to perform relevance analysis. This is taken to be the set of undergraduate students.

In Step 2, preliminary relevance analysis is performed via attribute removal and attribute generalization by applying attribute-oriented induction with conservative attribute generalization thresholds. The attributes name and phoneNo are removed because their number of distinct values exceeds their respective attribute analytical thresholds. Also concept hierarchies are used to generalize birth_place to birth_country, and birth_date to age_range. The attributes course and marks are also generalized to higher abstraction levels using the concept hierarchies. Hence, the attributes remaining for the candidate relation are gender, course, birth_country, age_range, and marks as shown below.

Candidate relation obtained for analytical characterization: target class(graduate students)

Gender	**Subject**	**CountryBorn**	**AgeGroup**	**Marks**	**Count**
M	Science	India	21....25	60	16
F	Science	Nepal	26...30	90	22
M	Maths	Nepal	26...30	90	18
F	Science	Nepal	26...30	90	25
M	Science	India	21..25	90	21
F	Maths	India	21...25	90	18

Candidate relation obtained for analytical characterization: contrasting class(under graduate student)

Gender	**Subject**	**CountryBorn**	**AgeGroup**	**Marks**	**Count**
M	Science	Nepal	<=20	70	18
F	MBA	India	<=20	50	20
M	MBA	India	<=20	50	22
F	Science	India	21...25	50	24
M	Maths	Nepal	21..25	70	22
F	Maths	India	<=20	90	24

In Step 3, the attributes in the candidate relation are evaluated using the selected relevance analysis measure, such as information gain. Let C_1 correspond to the class graduate and C_2 correspond to the class undergraduate. There are 120 samples of class graduate and 130 samples of class undergraduate. To compute the information gain of each attribute, we first compute the expected information needed to classify a given sample:

$$I(s_1, s_2) = I(120, 130) = -\frac{120}{250}\log_2\frac{120}{250} - \frac{130}{250}\log_2\frac{130}{250} = 0.9988$$

Next, we need to compute the entropy of each attribute. Let's try the attribute subject. We need to look at the distribution of graduate and undergraduate students for each value of subject. We compute the expected information for each of these distributions.

For subject= "Science";

$S_{11} = 84 \quad S_{21} = 42 \quad I(S_{11}, S_{21}) = 0.9183$

For subject = "Maths";

$S_{12} = 36 \quad S_{22} = 46 \quad I(S_{12}, S_{22}) = 0.9892$

For subject = "MBA";

$S_{13} = 0 \qquad S_{23} = 42 \quad I(S_{13}, S_{23}) = 0$

The expected information needed to classify a given sample if the samples are partitioned according to subject is

$$E(\text{subject}) = \frac{126}{250} I(s_{11}, s_{21}) + \frac{82}{250} I(s_{12,} s_{22}) + \frac{42}{250} I(s_{13}, s_{23}) = 0.7873$$

Hence, the gain in information from such a partitioning would be

$$\text{Gain(subject)} = I(S_1, S_2) - E(\text{subject}) = 0.2115.$$

Similarly, we can compute the information gain for each of the remaining attributes. The information gain for each attribute, sorted in increasing order, is 0.0003 for gender, 0.0407 for countryBorn, 0.2115 for subject and 0.4490 for marks. Suppose that we use an attribute relevance threshold of 0.1 to identify weakly relevant attributes. The information gain of the attributes gender and countryBorn are below the threshold, and therefore considered weakly relevant. Thus, they are removed. The contrasting class is also removed, resulting in the initial target class working relation.

In Step 4, attribute-oriented induction is applied to the initial target class working relation, using following algorithm:

Algorithm: Attribute_oriented_induction. Mining generalized characteristics in a relational database given a user's data mining request.

Input: (i) DB, a relational database; (ii) DMQuery, a data mining query; (iii) a_list, a list of attributes (containing attributes a_j); (iv) Gen(a_j), a set of concept hierarchies or generalization operators on attributes a_i; (v) a_en_thresh(a_j), attribute generalization thresholds for each ai.

Output: P, a Prime_eneralizedJelation.

Method: The method is outlined as follows:

1. W$\leftarrow$ get_task_relevant_data(DMQuery, DB)
2. prepare_for_generalization(W)

   //This is implemented as follows:

   (a) Scan W and collect the distinct values for each attribute a_j. (Note: If W is very large, this may be done by examining a sample of W.)

   (b) For each attribute ai, determine whether a_j should be removed, and if not, compute its minimum desired level L_i based on its given or default attribute threshold, and determine the mapping-pairs (v, v'), where v is a distinct value of a_j in W, and v_i is its corresponding generalized value at level L_i.

3. P $\leftarrow$ generalization(W);

   The Prime_generalized_relation, P, is derived by replacing each value v in W while accumulating count and computing any other aggregate values. This step can be implemented efficiently using either of the two variations shown on next page.

(a) For each generalized tuple, insert the tuple into a sorted prime relation P by a binary search; if the tuple is already in P, simply increase its count and other aggregate values accordingly; otherwise, insert it into P.

(b) Since in most cases the number of distinct values at the prime relation level is small, the prime relation can be coded as an m-dimensional array where m is the number of attributes in P, and each dimension contains the corresponding generalized attribute values. Each array element holds the corresponding count and other aggregation values, if any. The insertion of a generalized tuple is performed by measure aggregation in the corresponding array element.

10.4 MINING CLASS COMPARISON

In many applications, users may not be interested in having a single class (or concept) described or characterized, but rather would prefer to mine a description that compares or distinguishes one class (or concept) from other comparable classes (or concepts). Class discrimination or comparison (hereafter referred to as class comparison) mines descriptions that distinguish a target class from its contrasting classes. Notice that the target and contrasting classes must be comparable in the sense that they share similar dimensions and attributes. For example, the three classes person, address, and item are not comparable. However, the sales in the last three years are comparable classes, and so are computer science students versus physics students.

In general, the procedure for class comparison is as follows:

1. ***Data collection:*** The set of relevant data in the database is collected by query processing and is partitioned respectively into a target class and one or a set of contrasting class(es).

2. ***Dimension relevance analysis:*** If there are many dimensions and analytical comparison is desired, then dimension relevance analysis should be performed on these classes and only the highly relevant dimensions are included in the further analysis.

3. ***Synchronous generalization:*** Generalization is performed on the target class to the level controlled by a user or expert specified dimension threshold, which results in a prime target class relation/cuboid. The concepts in the contrasting class(es) are generalized to the same level as those in the prime target class relation/cuboid, forming the prime contrasting class(es) relation/cuboid.

4. ***Presentation of the derived comparison:*** The resulting class comparison description can be visualized in the form of tables, graphs, and rules. This presentation usually includes a "contrasting" measure (such as count%) that reflects the comparison between the target and contrasting classes. The user can adjust the comparison description by applying drill-down, roll-up, and other OLAP operations to the target and contrasting classes, as desired.

The above discussion outlines a general algorithm for mining analytical comparisons in databases. In comparison with analytical characterization, the above algorithm involves synchronous generalization of the target class with the contrasting classes so that classes are simultaneously compared at the same levels of abstraction.

Suppose that you would like to compare the general properties between the graduate students and the undergraduate students at Hisar-University, given the attributes name, gender, subject, birth_place, birth_date, residence, phoneNo and marks.

This data mining task can be expressed in DMQL as follows:

```
use Hisar_University_DB

mine comparison as "grad_vs_undergrad_students"

in relevance to name, gender, subject, birth_place, birth_date, residence,
phoneNo, marks

for "graduate_students"

where status in "graduate"
```

versus "undergraduate_students"

where status in "undergraduate"

analyze count%

from student

First, the query is transformed into two relational queries that collect two sets of task - relevant data: one for the initial target class working relation, and the other for the initial contrasting class working relation, as shown in tables below.

Initial target class working relation (graduates)

Name	Gender	Subject	Birth_place	Birth_date	Residence	PhoneNo	Marks
Subodh	M	Maths	Hisar	12.11.1983	Hisar	32231	56
Sarika	F	MBA	Karnal	04.06.1978	Hisar	34244	87
Pramod	M	MCA	Delhi	14.09.1981	Jind	54345	67

Initial contrasting class working relation (under graduates)

Name	Gender	Subject	Birth_place	Birth_date	Residence	PhoneNo	Marks
Raman	M	Biology	Hisar	02.01.1985	Hisar	44231	76
Savita	F	Physics	Karnal	04.06.1983	Hisar	65244	67
....	....	....	....	....	....	....	....

This can also be viewed as the construction of a data cube, where the status {graduate, undergraduate} serves as one dimension, and the other attributes form the remaining dimensions.

Secondly, dimension relevance analysis is performed on the two classes of data. After this analysis, irrelevant or weakly relevant dimensions, such as name, gender, birth_place, residence, and phoneNo, are removed from the resulting classes. Only the highly relevant attributes are included in the subsequent analysis.

Thirdly, synchronous generalization is performed: Generalization is performed on the target class to the levels controlled by user- or expert-specified dimension thresholds, forming the prime target class relation/cuboid. The contrasting class is generalized to the same levels as those in the prime target class relation/cuboid, forming the prime contrasting class(es) relation/cuboid, as presented in tables below. In comparison with undergraduate students, graduate students tend to be older and have a higher marks, in general.

Prime generalized relation for the target class (graduates)

subject	age_range	marks	count%
Science	21..25	60	5.53%
Science	26..30	60	5.02%
Science	over 30	70	5.86%
....			....
Maths		90	4.68%

subject	age_range	marks	count%
Science	16..20	50	5.53%
Science	16..20	60	4.53%
Science	26..30	60	2.32%
....	....	....	....
Maths	over 30	90	0.68%

Finally, the resulting class comparison is presented in the form of tables, graphs, and/or rules. This visualization includes a contrasting measure (such as count%) that compares between the target class and the contrasting class. For example, 5.02% of the graduate students majoring in Science are between 26 and 30 years of age and have 60 marks, while only 2.32% of undergraduates have these same characteristics. Drilling and other OLAP operations may be performed on the target and contrasting classes as deemed necessary by the user in order to adjust the abstraction levels of the final description.

10.5 DESCRIPTIVE STATISTICAL MEASURES

For many data mining tasks users would like to learn data characteristics regarding both central tendency and data dispersion. Measures of central tendency include mean, median, mode, and midrange, while measures of data dispersion include quartiles, outliers, and variance. These descriptive statistics are of great help in understanding the distribution of the data. Such measures have been studied extensively in the statistical literature. From the data mining point of view, we need to examine how they can be computed efficiently in large multidimensional databases.

Measuring Central Tendency

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}.$$

Arithmetic mean, average or simply mean is the most common and most effective numerical measure of the "center" of a set of data. The mean of a set of n observations $X_1, X_2, \ldots, X_n$ is defined as:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i . \quad \text{median} = L_1 + \left(\frac{n/2 - (\sum f)l}{f_{median}} \right) c,$$

This corresponds to the built-in aggregate function, average (avg() in SQL), provided in relational database systems. In most data cubes, sum and count are saved in precomputation. Thus, the derivation of average is straightforward, using the formula

average = sum/count.

Sometimes, each value X_i in a set may be associated with a weight W_i, for i = 1, . . . , n. The weights reflect the significance, importance, or occurrence frequency attached to their respective values. In this case, we can compute

This is called the weighted arithmetic mean or the weighted average.

Although the mean is the single most useful quantity that we use to describe a set of data, it is not the only, or even always the best, way of measuring the center of a set of data. For skewed data, a better measure of the center of data is the median, M. Suppose that the values forming a given

set of data are in numerical order. The median is the middle value of the ordered set if the number of values n is an odd number; otherwise (i.e., if n is even), it is the average of the middle two values.

Although it is not easy to compute the exact median value in a large database, an approximate median can be efficiently computed. For example, for grouped data, the median, obtained by interpolation, is given by

$$\text{median} = L_1 + \left(\frac{n/2 - (\sum f)l}{f_{\text{median}}} \right) c,$$

where L_1 is the lower class boundary of (i.e., lowest value for) the class containing the median, n is the number of values in the data.

Another measure of central tendency is the mode. The mode for a set of data is the value that occurs most frequently in the set. It is possible for the greatest frequency to correspond to several different values, which results in more than one mode. Data sets with one, two, or three modes are respectively called unimodal, bimodal, and trimodal. In general, a data set with two or more modes is multimodal. At the other extreme, if each data value occurs only once, then there is no mode.

For unimodal frequency curves that are moderately skewed (asymmetrical), we have the following empirical relation:

mean - mode = 3 x (mean - median).

This implies that the mode for unimodal frequency curves that are moderately skewed can easily be computed if the mean and median values are known.

The midrange, that is, the average of the largest and smallest values in a data set, can be used to measure the central tendency of the set of data. It is trivial to compute the midrange using the SQL aggregate functions, max () and min ().

Measuring the Dispersion of Data

The degree to which numeric data tend to spread is called the dispersion, or variance of the data. The most common measures of data dispersion are the five-number summary (based on quartiles), the interquartile range, and the standard deviation. The plotting of boxplots (which show outlier values) also serves as a useful graphical method.

Quartiles, Outliers, and Boxplots

The kth percentile of a set of data in numerical order is the value x having the property that k percent of the data entries lie at or below x. Values at or below the median M (discussed in the previous subsection) correspond to the 50th percentile.

The most commonly used percentiles other than the median are quartiles. The first quartile, denoted by Q1, is the 25th percentile; the third quartile, denoted by Q3, is the 75th percentile. The quartiles, including the median, give some indication of the center, spread, and shape of a distribution. The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. This distance is called the interquartile range (IQR) and is defined as

IQR = Q3 - Q1.

We should be aware that no single numerical measure of spread, such as IQR, is very useful for describing skewed distributions. The spreads of two sides of a skewed distribution are unequal. Therefore, it is more informative to also provide the two quartiles Q1 and Q3, along with the median, M. One common rule of thumb for identifying suspected outliers is to single out values falling at least 1.5 x IQR above the third quartile or below the first quartile.

Because Q1, M and Q3 contain no information about the endpoints (e.g., tails) of the data, a fuller summary of the shape of a distribution can be obtained by providing the highest and lowest data

values as well. This is known as the five-number summary. The five-number summary of a distribution consists of the median M, the quartiles Q1 and Q3, and the smallest and largest individual observations, written in the order Minimum, Q1, M, Q3, Maximum.

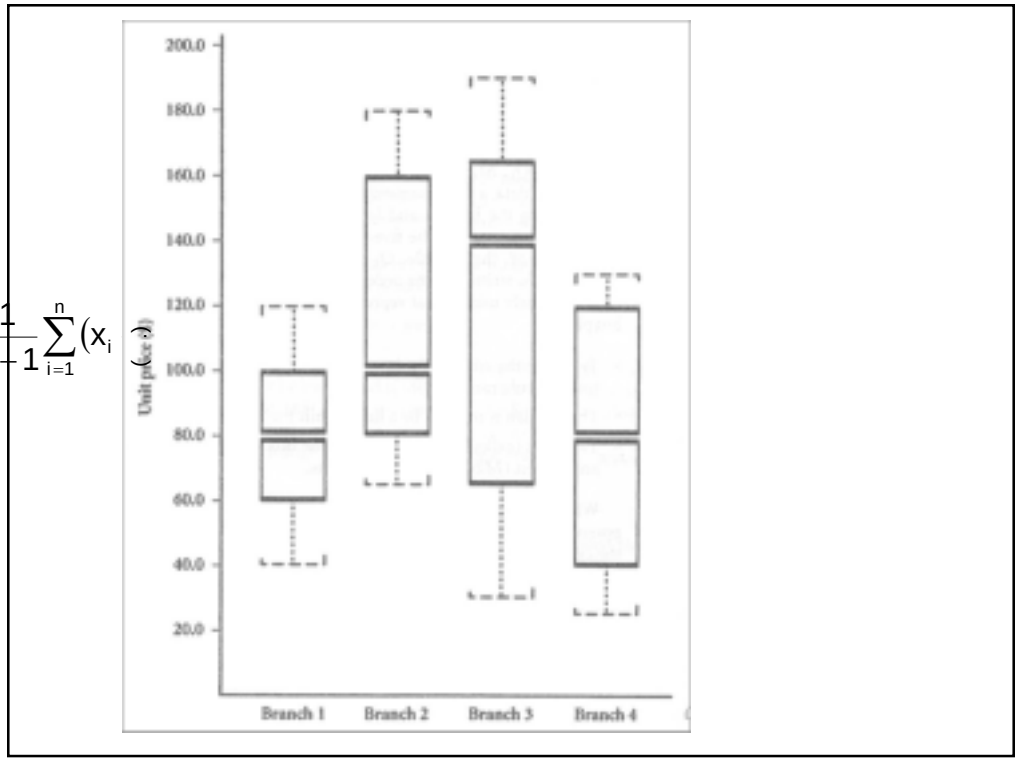
A popularly used visual representation of a distribution is the boxplot. In a boxplot:

Typically, the ends of the box are at the quartiles, so that the box length is the interquartile range, IQR.

The median is marked by a line within the box.

Two lines (called whiskers) outside the box extend to the smallest (Minimum) and largest {Maximum} observations.

When dealing with a moderate number of observations, it is worthwhile to plot potential outliers individually. To do this in a boxplot, the whiskers are extended to the extreme high and low observations only if these values are less than 1.5 x IQR beyond the quartiles. Otherwise, the whiskers terminate at the most extreme observations occurring within 1.5 x IQR of the quartiles. The remaining cases are plotted individually. Boxplots can be used in the comparisons of several sets of compatible data. Figure given below 10.1shows boxplots for unit price data for items sold at four branches of HTC during a given time period. For branch 1, we see that the median price of items sold is Rs.50, Q1 is Rs.60, Q3 is Rs.100.



$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i$$

Figure 10.1

Variance and Standard Deviation

The variance of n observations $X_1, X_2, \ldots, X_n$ is

The standard deviation S is the square root of the variance.

Student Activity 1

1. For class characterization, what are the major differences between a data cube based implementation and a relational implementation such as attribute-oriented induction? Discuss which method is most efficient and under what condition this is so.

10.6 SUMMARY

1. The simplest kind of descriptive data mining is concept description.
2. A concept usually refers to a collection of data such as frequent_buyers, graduate_student, and so on.
3. Data Generalization is a process that abstract a large set of task relevant data in a database from a relatively low conceptual level to higher conceptual levels.
4. There are many possible ways to control a generalization process.
5. The data cube implementation of attribute oriented induction can be performed in two ways.
6. Data mining can be classified into descriptive data mining and predictive data mining.
7. A popularly used visual representation of a distribution is the boxplots.
8. Concept description has a close ties with data generalization.
9. A database usually does not store the negative data explictitly.
10. Concept description can be performed incrementally, in parallel, or in a distributed manner.

10.7 KEYWORDS

Data generalization: It is a process that abstracts a large set of task-relevant data in a database from a relatively low conceptual level to higher conceptual levels.

Interquartile range: The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. This distance is called the interquartile range (IQR)

10.8 REVIEW QUESTIONS

1. Describe Attribute-Oriented Induction with the help of suitable example.
2. List down the steps for attribute relevance analysis for concept description.
3. Outline a data cube-based incremental algorithm for mining analytical class comparisons.
4. Suppose that the data for analysis includes the attributes age. The age values for the data tuples are (in increasing order):

   13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

   a. What is the mean of the data?
   b. What is median?
   c. What is the mode of the data?
   d. What is the mid range of the data?
   e. Find the first quartile(Q1) and the third quartile (Q3) of the data

5. Write short note on:

   a. Five number Summary
   b. Boxplots

10.9 FURTHER READINGS

Bezdek, J. C., & Pal, S. K. (1992). *Fuzzy models for pattern recognition: Methods that search for structures in data*. New York: IEEE Press

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (Eds.). (1996). *Advances in knowledge discovery and data mining.* AAAI/MIT Press.

Han, J., & Kamber, M. (2000). *Data mining: Concepts and techniques*: Morgan Kaufmann.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: Data mining, inference, and prediction:* New York: Springer.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. New Jersey: Prentice Hall.

Jensen, F. V. (1996). *An introduction to bayesian networks*. London: University College London Press.

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis*. New York: John Wiley.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine learning, neural and statistical classification*: Ellis Horwood.

Fukunaga, K., *Introduction to Statistical Pattern Recognition,* Academic Press, 1990

Groth, R., *Data Mining, A Hands-On Approach for business Professionals*, Prentice Hall, 1998 [with demo software]

Hand, D.J., Mannila, H., Smyth, P., *Principles of Data Mining*, MIT Press, [late] 2000

Jain, A. and Dubes, R., *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988

Jordan, M.I., *Learning in Graphical Models*, MIT Press, 1999

Kargupta, Hillol, and Chan, Philip, *Advances in Distributed and Parallel Knowledge Discovery*, MIT/AAAI Press, 2000

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*," New York: Springer-Verlag, 1994

Mitchell, T.M., *Machine Learning*, New York, NY: McGrawl Hill, 1997

Ripley, B.D., *Pattern Recognition and Neural Networks*, Cambridge, UK: Cambridge University Press, 1996

UNIT

11

MINING ASSOCIATION RULE

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Describe Brute Force.
- Discuss some properties of frequent item sets.
- Know about single and Multidimensional associations.
- Know more about constant based associations.
- Know about Metarules.
- Explain Market basket Analysis.

U N I T S T R U C T U R E

11.1 INTRODUCTION

One of the reasons behind maintaining any database is to enable the user to find interesting patterns and trends in the data. For example, in a supermarket, the user can figure out which items are being sold most frequently. But this is not the only type of 'trend' which one can possibly think of. The goal of database mining is to automate this process of finding interesting patterns and trends. Once this information is available, we can perhaps get rid of the original database. The output of the data-mining process should be a "summary" of the database. This goal is difficult to achieve due to the vagueness associated with the term 'interesting'. The solution is to define various types of trends and to look for only those trends in the database. One such type constitutes the association rule.

In the rest of the discussion, we shall assume the supermarket example, where each record or tuple consists of the items of a single purchase. However the concepts are applicable in a large number of situations.

In the present context, an association rule tells us about the association between two or more items. For example: In 80% of the cases when people buy bread, they also buy milk. This tells us of the association between bread and milk. We represent it as -

bread => milk | 80%

This should be read as - "Bread means or implies milk, 80% of the time." Here 80% is the "confidence factor" of the rule.

Association rules can be between more than 2 items. For example -

bread, milk => jam | 60%

bread => milk, jam | 40%

Given any rule, we can easily find its confidence. For example, for the rule

bread, milk => jam

we count the number say n_1, of records that contain bread and milk. Of these, how many contain jam as well? Let this be n_2. Then required confidence is n_2/n_1.

This means that the user has to guess which rule is interesting and ask for its confidence. But our goal was to "automatically" find all interesting rules. This is going to be difficult because the database is bound to be very large. We might have to go through the entire database many times to find all interesting rules.

11.2 BRUTE FORCE

The common-sense approach to solving this problem is as follows -

Let $I = \{ i_1, i_2, ..., i_n \}$ be a set of items, also called as an itemset. The number of times, this itemset appears in the database is called its "support". Note that we can speak about support of an itemset and confidence of a rule. The other combinations - support of a rule and confidence of an itemset are not defined.

Now, if we know the support of 'I' and all its subsets, we can calculate the confidence of all rules which involve these items. For example, the confidence of the rule

$i_1, i_2, i_3 => i_4, i_5$

$$\text{is} \frac{\text{support of } \{i_1, i_2, i_3, i_4, i_5\}}{\text{support of } \{i_1, i_2, i_3\}}$$

So, the easiest approach would be to let 'I' contain all items in the supermarket. Then setup a counter for every subset of 'I' to count all its occurrences in the database. At the end of one pass of the database, we would have all those counts and we can find the confidence of all rules. Then select the most "interesting" rules based on their confidence factors. How easy.

The problem with this approach is that, normally 'I' will contain atleast about 100 items. This means that it can have 2^{100} subsets. We will need to maintain that many counters. If each counter is a single byte, then about 10^{20} GB will be required. Clearly this can't be done.

11.3 MINIMUM SUPPORT

To make the problem tractable, we introduce the concept of minimum support. The user has to specify this parameter - let us call it minsupport. Then any rule

$i_1, i_2, ..., i_n => j_1, j_2, ..., j_n$

needs to be considered, only if the set of all items in this rule which is $\{ i_1, i_2, ..., i_n, j_1, j_2, ..., j_n \}$ has support greater than minsupport.

The idea is that in the rule

bread, milk => jam

if the number of people buying bread, milk and jam together is very small, then this rule is hardly worth consideration (even if it has high confidence).

Our problem now becomes - Find all rules that have a given minimum confidence and involves itemsets whose support is more than minsupport. Clearly, once we know the supports of all these itemsets, we can easily determine the rules and their confidences. Hence we need to concentrate on the problem of finding all itemsets which have minimum support. We call such itemsets as **frequent itemsets**.

11.4 SOME PROPERTIES OF FREQUENT ITEMSETS

The methods used to find frequent itemsets are based on the following properties -

1. **Every subset of a frequent itemset is also frequent.** Algorithms make use of this property in the following way - we need not find the count of an itemset, if all its subsets are not frequent. So, we can first find the counts of some short itemsets in one pass of the database. Then consider longer and longer itemsets in subsequent passes. When we consider a long itemset, we can make sure that all its subsets are frequent. This can be done because we already have the counts of all those subsets in previous passes.

2. Let us divide the tuples of the database into partitions, not necessarily of equal size. Then **an itemset can be frequent only if it is frequent in atleast one partition**. This property enables us to apply divide and conquer type algorithms. We can divide the database into partitions and find the frequent itemsets in each partition. An itemset can be frequent only if it is frequent in atleast one of these partitions. To see that this is true, consider k partitions of sizes n_1, n_2,..., n_k.

3. Let minimum support be s.

4. Consider an itemset which does not have minimum support in any partition. Then its count in each partition must be less than sn_1, sn_2,..., sn_k respectively.

5. Therefore its total count must be less than the sum of all these counts, which is $s(n_1 + n_2 +...+ n_k)$.

6. This is equal to s*(size of database).

7. Hence the itemset is not frequent in the entire database.

Market Basket Analysis

Suppose, as manager of a branch of HTC, you would like to learn more about the buying habits of your customers. You may also want to know which groups or sets of items are customers likely to purchase on a given trip to the store. To answer your question, market basket analysis may be performed on the retail data of customer transactions at your store. The results may be used to plan marketing or advertising strategies, as well as catalog design. For instance, market basket analysis may help managers design different store layouts. In one strategy, items that are frequently purchased together can be placed in close proximity in order to further encourage the sale of such items together. If customers who purchase computers also tend to buy cellphone at the same time, then placing the computer display close to the cellphone display may help to increase the sales of both of these items. In an alternative strategy, placing computer and cellphone at opposite ends of the store may entice customers who purchase such items to pick up other items along the way. For instance, after deciding on an expensive computer, a customer may observe security systems for sale while heading towards the cellphone display. Market basket analysis can also help retailers to plan which items to put on sale at reduced prices. If customers tend to purchase computers and printers together, then having a sale on printers may encourage the sale of printers as well as computers.

If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of that item. Each basket can then be represented

by a Boolean vector of values assigned to these variables. The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently associated or purchased together. These patterns can be represented in the form of association rules. For example, the information that customers who purchase computers also tend to buy financial management software at the same time is represented in Association Rule below:

computer $\Rightarrow$ cell_phone

[support = 2%, confidence = 60%]

This reflects the usefulness and certainty of discovered rules. A support of 2% for Association Rule above means that 2% of all the transactions under analysis show that computer and cellphone are purchased together. A confidence of 60% means that 60% of the customers who purchased a computer also bought the software. Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold. Such thresholds can be set by users or domain experts.

Let $\Im = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let D, the task-relevant data, be a set of database transactions where each transaction T is a set of items such that $T \subseteq \Im$. Each transaction is associated with an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset \Im$, $B \subset \Im$, and $A \cap B = \phi$. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., both A and B). This is taken to be the probability, $P(A \cup B)$. The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, P(B|A). That is,

$support(A \Rightarrow B) = P(A \cup B)$

$confidence(A \Rightarrow B) = P(B\backslash A)$.

Rules that satisfy both a minimum support threshold (min_sup) and a minimum confidence threshold (min_conf) are called strong. By convention, we write support and confidence values so as to occur between 0% and 100%, rather than 0 to 1.0.

A set of items is referred to as an itemset. An itemset that contains k items is a k-itemset. The set {computer, cell _phone} is a 2-itemset. The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is also known, simply, as the frequency, support count, or count of the itemset. An item set satisfies minimum support if the occurrence frequency of the itemset is greater than or equal to the product of min_sup and the total number of transactions in D. The number of transactions required for the itemset to satisfy minimum support is therefore referred to as the minimum support count. If an itemset satisfies minimum support, then it is a frequent itemset. The set of frequent k-itemsets is commonly denoted by L_k.

Association rule mining is a two-step process:

1. ***Find all frequent itemsets:*** By definition, each of these itemsets will occur at least as frequently as a pre-determined minimum support count.

2. ***Generate strong association rules from the frequent itemsets:*** By definition, these rules must satisfy minimum support and minimum confidence.

Additional interestingness measures can be applied, if desired. The second step is the easier of the two. The overall performance of mining association rules is determined by the first step.

Market basket analysis is just one form of association rule mining. In fact, there are many kinds of association rules. Association rules can be classified in various ways, based on the following criteria:

- Based on the types of values handled in the rule: If a rule concerns associations between the presence or absence of items, it is a Boolean association rule.

  If a rule describes associations between quantitative items or attributes, then it is a quantitative association rule. In these rules, quantitative values for items or attributes are

partitioned into intervals. The following rule is an example of a quantitative association rule, where X is a variable representing a customer:

age (X, "30 . . .39") $\wedge$ income(X, "42000 . . . 48000") $\Rightarrow$ buys(X, cell_phone)

The quantitative attributes, age and income, have been discretized.

- Based on the dimensions of data involved in the rule: If the items or attributes in an association rule reference only one dimension, then it is a single dimensional association rule. The rule could be rewritten as:

  buys(X, "computer")$\Rightarrow$buys(X, cell_phone)

  If a rule references two or more dimensions, such as the dimensions buys, time_of_transaction, and customer_category, then it is a multidimensional association rule.

- Based on the levels of abstractions involved in the rule set: Some methods for association rule mining can find rules at differing levels of abstraction. For example, suppose that a set of association rules mined includes the following rules:

  age (X, "30 . . . 39") $\Rightarrow$ buys(X, "laptop computer")

  age(X, "30 . . . 39") $\Rightarrow$ buys(X, "computer")

  The items bought are referenced at different levels of abstraction. (e.g., "computer" is a higher-level abstraction of "laptop computer".) We refer to the rule set mined as consisting of multilevel association rules. If, instead, the rules within a given set do not reference items or attributes at different levels of abstraction, then the set contains single-level association rules.

- Based on various extensions to association mining: Association mining can be extended to correlation analysis, where the absence or presence of correlated items can be identified. It can also be extended to mining maxpatterns (i.e., maximal frequent patterns) and frequent closed itemsets. A maxpattern is a frequent pattern, p, such that any proper super patterns of p is not frequent. A frequent closed itemset is a frequent closed item set where an itemset c is closed if there exists no proper superset of c, c', such that every transaction containing c also contains c'. Maxpatterns and frequent closed itemsets can be used to substantially reduce the number of frequent item sets generated in mining.

11.5 SINGLE DIMENSIONAL ASSOCIATIONS

Apriori is an influential algorithm for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties, as we shall see below. Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k + 1) itemsets. First, the set of frequent l-itemsets is found. This set is denoted L_1. L_1 is used to find L_2, the set of frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-itemsets can be found. The finding of each L_k requires one full scan of the database.

To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property, presented below, is used to reduce the search space.

In order to use the Apriori property, all nonempty subsets of a frequent itemset must also be frequent. This property is based on the following observation. By definition, if an itemset I does not satisfy the minimum support threshold, minsup, then I is not frequent, that is, P(I) < min_sup. If an item A is added to the itemset I, then the resulting itemset (i.e., I $\cup$ A) cannot occur more frequently than I. Therefore, I $\cup$ A is not frequent either, that is, P(I $\cup$ A) < min_sup.

This property belongs to a special category of properties called anti-monotone in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. It is called anti-monotone because the property is monotonic in the context of failing a test.

To understand how apriori is used, let us look at how L_{k-1} is used to find L_k. A two-step process is followed, consisting of join and prune actions.

1. The join step: To find L_k, a set of candidate k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k. Let l_1 and l_2 be itemsets in L_{k-1}. The notation $l_i[j]$ refers to the jth item in l_1 (e.g., l_1 [k - 2] refers to the second to the last item in l_1). By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order.

   The join, $L_{k-1} \lhd \rhd L_{k-1}$, is performed, where members of L_{k-1} are joinable if their first (k - 2) items are in common. That is, members l_1 and l_2 of L_{k-1} are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] = l_2[k-1])$. The condition $l_1[k-1] = l_2[k-1])$ simply ensures that no duplicates are generated. The resulting itemset formed by joining l_1 and l_2 is $l_1[1]\, l_1[2] \ldots .. l_1[k-1] l_2[k-1]$.

2. The prune step: C_k is a superset of L_k, that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in C_k. A scan of the database to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k). C_k, however, can be huge, and so this could involve heavy computation. To reduce the size of C_k, the Apriori property is used as follows. Any (k - l)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k – 1)-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k. This subset testing can be done quickly by maintaining a hash tree of all frequent itemsets.

To illustrate this algorithm, let us take an example based on the HTC transaction database, D given below.

TID	List of item_IDs
100	1, 2, 5
200	2, 4
300	2, 3
400	1, 2, 4
500	1, 3
600	2, 3
700	1, 3
800	1, 2, 3, 5
900	1, 2, 3

$\binom{|L_1|}{2}$

There are nine transactions in this database, that is, |D| = 9. Let us apply Apriori algorithm for finding frequent itemsets in D.

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C_1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.

2. Suppose that the minimum transaction support count required is 2 (i.e., min_sup = 2/9 = 22%). The set of frequent l-itemsets, L_1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support.

3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses $L_1 \bowtie L_1$ to generate a candidate set of 2-itemsets, C_2. C_2 consists of 2-itemsets.

4. Next, the transactions in D are scanned and the support count of each candidate itemset in C_2 is accumulated.

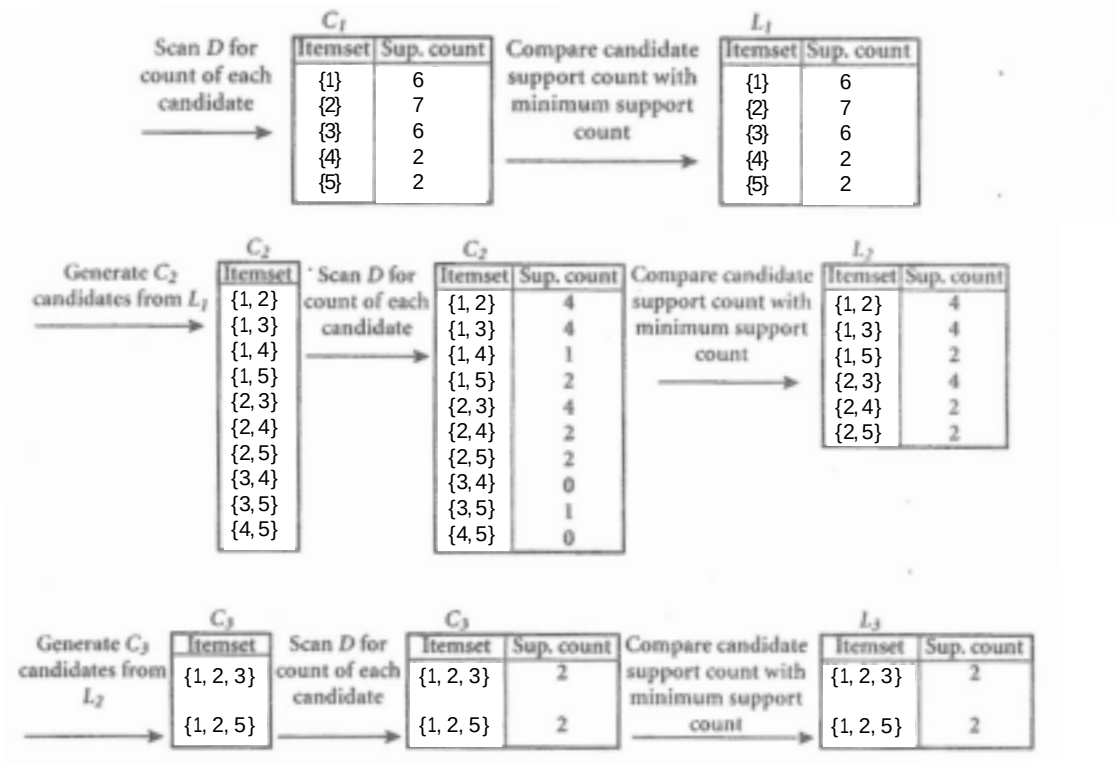
5. The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.

6. The set of candidate 3-itemsets, C_3, is generated. First, let $C_3 = L_2 \bowtie L_2 = \{\{1,2,3\}, \{1,2,5\}, \{1,3,5\}, \{2,3,4\}, \{2,3,5\}, \{2,4,5\}\}$. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We therefore remove them from C_3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L_3. Note that when given a candidate k-itemset, we only need to check if its (k - I)-subsets are frequent since the Apriori algorithm uses a level-wise search strategy.

7. The transactions in D are scanned in order to determine L_3, consisting of those candidate 3-itemsets in C_3 having minimum support.

8. The algorithm uses $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, C_4. Although the join results in $\{\{1,2,3,5\}\}$, this itemset is pruned since its subset $\{\{1,2,3,5\}\}$ is not frequent. Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets.

The result is presented in the following tables.



The Apriori algorithm and its related procedures:

Algorithm: Find frequent item sets using an iterative level-wise approach based on candidate generation.

Input: Database, D, of transactions; minimum support threshold, min_sup.

Output: L, frequent itemsets in D.

Method:

(1) L_1 = find_frequent_1-itemsets(D);

(2) for($k = 2$; $L_{k-1} \neq \phi$; k++) {

(3) C_k = aprior_gen(L_{k-1}, min_sup);

(4) for each transaction $t \in D$ {// scan D for counts

(5) C_t=subset(C_k, f); // get the subsets of t that are candidates

(6) for each candidate $c \in C_t$

(7) c.count++;

(8) }

(9) $L_k = \{c \in C_k | c.count \geq min\_sup\}$

(10) }

(11) return $L = \cup_k L_k$;

procedure aprior_gen(L_{k-1}: frequent (k - 1)-itemsets; min_sup: minimum support threshold)

(1) for each itemset $l_1 \in$ Lk-1

(2) for each itemset $l_2 \in L_{k-1}$

(3) if$(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge ... \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then

(4) $c = l_1 \bowtie l_2$; // join step: generate candidates

(5) if has_infrequent_subset(c, L_{k-1} then

(6) delete c; // prune step: remove unfruitful candidate

(7) else add c to C_k;

(8) }

(9) return C_k;

procedure has_infrequent_subset(c: candidate k-itemset; L_{k-1}: frequent (k - l)-itemsets);

// use prior knowledge

(1) for each (k - 1)-subset s of c

(2) if $5 \notin L_{k-1}$ then

(3) return TRUE;

(4) return FALSE;

Once the frequent itemsets from transactions in a database D have been found, it is straightforward to generate strong association rules from them (where strong association rules satisfy both minimum support and minimum confidence). This can be done using the following equation for confidence, where the conditional probability is expressed in terms of item set support count:

$$\text{confidence}\ (A \Rightarrow B) = P(B|\ A) = \frac{\text{sup port}\ \_\text{count}\ (A \cup B)}{\text{sup port}\ \_\text{count}\ (A)},$$

where support_count($A \cup B$) is the number of transactions containing the itemsets $A \cup B$, and support_count(A) is the number of transactions containing the itemset A. Based on this equation, association rules can be generated as follows:

For each frequent item set l, generate all nonempty subsets of l.

For every nonempty subset s of l, output the rule "s => (l – s)" if min_conf, where $\frac{\text{support\_count}(l)}{\text{support\_count}(s)} \geq$ min_conf is the minimum confidence threshold.

Since the rules are generated from frequent itemnsets, each one automatically satisfies minimum support, Frequent item sets can be stored ahead of time in hash tables along with their counts so

that they can be accessed quickly.

An example based on the transactional data for HTC is given below. Suppose the data contain the frequent itemset l = {1, 2, 5}. What are the association-rules that can be generated from l? The nonempty subsets of l are {1,2}, {1,5}, {2,5}, {1}, {2}, and {5}. The resulting association rules areas shown below, each listed with its confidence:

$1 \wedge 2 \Rightarrow 5$, confidence = 2/4 = 50%

$1 \wedge 5 \Rightarrow 2$, confidence = 2/2 = 100%

$2 \wedge 5 \Rightarrow 1$, confidence = 2/2 = 100%

$1 \Rightarrow 2 \wedge 5$, confidence = 2/6 = 33%

$2 \Rightarrow 1 \wedge 5$, confidence = 2/7 = 29%

$5 \Rightarrow 1 \wedge 2$, confidence = 2/2 = 33%

If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, since these are the only ones generated that are strong.

11.6 MULTIDIMENSIONAL ASSOCIATIONS

Association rules that involve two or more dimensions or predicates can be referred to as multidimensional association rules. Multidimensional association rules with no repeated predicates are called interdimension association rules. We may also be interested in mining multidimensional association rules with repeated predicates, which contain multiple occurrences of some predicates. These rules are called hybrid-dimension association rules. An example of such a rule is the following, where the predicate buys is repeated:

$$age(X, \text{“20 . . .29”}) \wedge buys(X, \text{“laptop”}) \Rightarrow buys(X, \text{“b/w printer”})$$

Note that database attributes can be categorical or quantitative. Categorical attributes have a finite number of possible values, with no ordering among the values (e.g., occupation, brand, color). Categorical attributes are also called nominal attributes, since their values are “names of things:’ Quantitative attributes are numeric and have an implicit ordering among values (e.g., age, income, price). Techniques for mining multidimensional association rules can be categorized according to three basic approaches regarding the treatment of quantitative attributes.

In the first approach, quantitative attributes are discretized using predefined concept hierarchies. This discretization occurs prior to mining. For instance, a concept hierarchy for income may be used to replace the original numeric values of this attribute by ranges, such as. “0. . . 20000”, “21000 . . . 30000”, “31000 . . . 40000”, and so on. Here, discretization is static and predetermined. The discretized numeric attributes, with their range values, can then be treated as categorical attributes (where each range is considered a category). We refer to this as mining multidimensional association rules using static discretization of quantitative attributes.

In the second approach, quantitative attributes are discretized into “bins” based on the distribution of the data. These bins may be further combined during the mining process. The discretization process is dynamic and established so as to satisfy some mining criteria, such as maximizing the confidence of the rules mined. Because this strategy treats the numeric attribute values as quantities rather than as predefined ranges or categories, association rules mined from this approach are also referred to as quantitative association rules.

In the third approach, quantitative attributes are discretized so as to capture the semantic meaning of such interval data. This dynamic discretization procedure considers the distance between data points. Hence, such quantitative association rules are also referred to as distance-based association rules.

Mining Multidimensional Association Rules Using Static Discretization of Quantitative Attributes

Quantitative attributes, in this case, are discretized prior to mining using predefined concept

hierarchies, where numeric values are replaced by ranges. Categorical attributes may also be generalized to higher conceptual levels if desired. If the resulting task-relevant data are stored in a relational table, then the Apriori algorithm requires just a slight modification so as to find all frequent predicate sets rather than frequent itemsets (i.e., by searching through all of the relevant attributes, instead of searching only one attribute, like buys). Finding all frequent k-predicate sets will require k or k + 1 scans of the table. Other strategies, such as hashing, partitioning, and sampling may be employed to improve the performance.

Alternatively, the transformed task-relevant data may be stored in a data cube. Data cubes are well suited for the mining of multidimensional association rules, since they are multidimensional by definition.

Due to the ever-increasing use of data warehousing and OLAP technology, it is possible that a data cube containing the dimensions of interest to the user may already exist, fully materialized. A strategy similar to that employed in Apriori can be used, based on prior knowledge that every subset of a frequent predicate set must also be frequent. This property can be used to reduce the number of candidate predicate sets generated. In cases where no relevant data cube exists for the mining task, one must be created.

Mining Quantitative Association Rules

Quantitative association rules are multidimensional association rules in which the numeric/ attributes are dynamically discretized during the mining process so as to satisfy some mining criteria, such as maximizing the confidence or compactness of the rules mined. In this section, we will focus specifically on how to mine quantitative association rules having two quantitative attributes on the left-hand side of the rule, and one categorical attribute on the right-hand side of the rule, for example,

$$A_{quanl} \wedge A_{quan2} \Rightarrow A_{cat}$$

where A_{quan1} and A_{quan2} are tests on quantitative attribute ranges (where the ranges are dynamically determined), and A_{cat} tests a categorical attribute from the task-relevant data. Such rules have been referred to as two-dimensional quantitative association rules, since they contain two quantitative dimensions. For instance, suppose you are curious about the association relationship between pairs of quantitative attributes, like customer age and income, and the type of 'television that customers like to buy. An example of such a 2- D quantitative association rule is

$$age(X, \text{“30}\ldots\text{39”}) \wedge income(X, \text{“42000}\ldots\text{48000”}) \Rightarrow buys(X, \text{“cellphone”})$$

To see how to find such rule, let's look at an approach used in a system called ARCS (Association Rule Clustering System), which borrows ideas from image processing. Essentially, this approach maps pairs of quantitative attributes onto a 2-D grid for tuples satisfying a given categorical attribute condition. The grid is then searched for clusters of points, from which the association rules are generated. The following steps are involved in ARCS:

- ***Binning:*** Quantitative attributes can have a very wide range of values defining their domain. Just think about how big a 2-D grid would be if we plotted age and income as axes, where each possible value of age was assigned a unique position on one axis, and similarly, each possible value of income was assigned a unique position on the other axis! To keep grids down to a manageable size, we instead partition the ranges of quantitative attributes into intervals. These intervals are dynamic in that they may later be further combined during the mining process. The partitioning process is referred to as binning, that is, where the intervals are considered "bins." Three common binning strategies are

  Equiwidth binning, where the interval size of each bin is the same,

  Equidepth binning, where each bin has approximately the same number of tuples assigned to it, and

  Homogeneity-based binning, where bin size is determined so that the tuples in each bin are uniformly distributed.

  ARCS uses equiwidth binning, where the bin size for each quantitative attribute is input by the user. A 2-D array for each possible bin combination involving both quantitative attributes

is created. Each array cell holds the corresponding count distribution for each possible class of the categorical attribute of the rule right-hand side. By creating this data structure, the task-relevant data need only be scanned once. The same 2-D array can be used to generate rules for any value of the categorical attribute, based on the same two quantitative attributes.

- ***Finding frequent predicate sets:*** Once the 2-D array containing the count distribution for each category is set up, this can be scanned in order to find the frequent predicate sets (those satisfying minimum support) that also satisfy minimum confidence. Strong association rules can then be generated from these predicate sets, using a rule generation algorithm.
- ***Clustering the association rules:*** The strong association rules obtained in the previous step are then mapped to a 2-D grid.

11.7 ASSOCIATION MINING AND CORRELATION ANALYSIS

Most association rule mining algorithms employ a support-confidence framework. In spite of using minimum support and confidence thresholds to help weed out or exclude the exploration of uninteresting rules, many rules that are not interesting to the user may still be produced.

In data mining all of the strong association rules discovered (i.e., those rules satisfying the minimum support and minimum confidence thresholds) are not necessarily interesting to the user. Whether a rule is interesting or not can be judged either subjectively or objectively. Ultimately, only the user can judge if a given rule is interesting or not, and this judgment, being subjective, may differ from one user to another. However, objective interestingness measures, based on the statistics "behind" the data, can be used as one step towards the goal of weeding out uninteresting rules from presentation to the user.

To see how can we tell which strong association rules are really interesting, let's examine the following example.

Suppose we are interested in analyzing transactions at HTC with respect to the purchase of computer games and videos. Let game refer to the transactions containing computer games, and video refer to those containing videos. Of the 10,000 transactions analyzed, the data show that 6000 of the customer transactions included computer games, while 7500 included videos, and 4000 included both computer games and videos. Suppose that a data mining program for discovering association rules is run on the data, using a minimum support of, say, 30% and a minimum confidence of 60%. The following association rule is discovered:

$$\text{buys(X, "computer games")} \Rightarrow \text{buys(X, "videos")} \text{ [support = 40\%, confidence = 66\%]}$$

This rule is a strong association rule and would therefore be reported, since its support value of 40% and confidence value of 66% satisfy the minimum support and minimum confidence thresholds, respectively. However, it is misleading since the probability of purchasing videos is 75%, which is even larger than 66%. In fact, computer games and videos are negatively associated because the purchase of one of these items actually decreases the likelihood of purchasing the other. Without fully understanding this phenomenon, one could make unwise business decisions based on the rule derived.

The above example also illustrates that the confidence of a rule $A \Rightarrow B$ can be deceiving in that it is only an estimate of the conditional probability of itemset B given itemset A. It does not measure the real strength (or lack of strength) of the implication between A and B. Hence, alternatives to the support-confidence framework can be useful in mining interesting data relationships.

Association rules mined using a support-confidence framework are useful for many applications. However, the support-confidence framework can be misleading in that it may identify a rule $A \Rightarrow B$ as interesting when, in fact, the occurrence of A does not imply the occurrence of B. In this section, we consider an alternative framework for finding interesting relationships between data itemsets based on correlation.

The occurrence of itemset A is independent of the occurrence of itemset B if $P(A \cup B) = P(A)P(B)$; otherwise itemsets A and B are dependent and correlated as events. This definition can easily be

extended to more than two itemsets. The correlation between the occurrence of A and B can be measured by computing

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}.$$

If the resulting value is less than 1, then the occurrence of A is negatively correlated with (or discourages) the occurrence of B. If the resulting value is greater than 1, then A and B are positively correlated, meaning the occurrence of one implies the occurrence of the other. If the resulting value is equal to 1, then A and B are independent and there is no correlation between them.

The equation is equivalent to P(BIA)/P(B), which is also called the lift of the association rule A⇒B. For example, if A corresponds to the sale of computer games and B corresponds to the sale of videos, then given the current market conditions, the sale of games is said to increase or "lift" the likelihood of the sale of videos by a factor of the value returned by the equation.

To help filter out misleading "strong" associations of the form A⇒B, we need to study how the two itemsets, A and B, are correlated. Let game refer to the transactions of this example that do not contain computer games, and video refer to those that do not contain videos. The transactions can be summarized in a contingency table. A contingency table is shown below.

	game	game	Σ_{row}
video	4,000	3,500	7,500
video	2,000	500	2,500
Σ_{col}	6,000	4,000	10,000

From the table, we can see that the probability of purchasing a computer game is P({game}) = 0.60, the probability of purchasing a video is P({video}) = 0.75, and the probability of purchasing both is P({game, video}) = 0.40. P({game, video})/(P({game}) x P({video})) = 0.40/(0.60 x 0.75) = 0.89. Since this value is less than 1, there is a negative correlation between the occurrences of {game} and {video}. The numerator is the likelihood of a customer purchasing both, while the denominator is what the likelihood would have been if the two purchases were completely independent. Such a negative correlation cannot be identified by a support-confidence framework.

This motivates the mining of rules that identify correlations, or correlation rules. A correlation rule is of the form $\{i_1, i_2,....i_m\}$ where the occurrences of the items are correlated. Given a correlation value, the X^2 statistic can be used to determine if the correlation is statistically significant. The X^2 statistic can also determine negative implication.

An advantage of correlation is that it is upward closed. This means that if a set S of items is correlated (i.e., the items in S are correlated), then every superset of S is also correlated. In other words, adding items to a set of correlated items does not remove the existing correlation. The X^2 statistic is also upward closed within each significance level.

When searching for sets of correlations to form correlation rules, the upward closure property of correlation and X^2 can be used. Starting with the empty set, we may explore the item set space (or itemset lattice), adding one item at a time, looking for "minimal correlated itemsets-itemsets that are correlated although no subset of them is correlated. These itemsets form a border within the lattice. Because of closure, no item set below this border will be correlated. Since all supersets of a minimal correlated itemset are correlated, we can stop searching upward. An algorithm that performs a series of such "walks" through itemset space is called a random walk algorithm. Such an algorithm can be combined with tests of support in order to perform additional pruning. Random walk algorithms can easily be implemented using data cubes. It is an open problem to adapt the procedure described here to very large databases. Another limitation is that the X^2 statistic is less accurate when the contingency table data are sparse, and can be misleading for contingency tables larger than 2 x 2.

11.8 CONSTRAINT BASED ASSOCIATIONS

For a given set of task-relevant data, the data mining process may uncover thousands of rules, many of which are uninteresting to the user. In constraint-based mining, mining is performed under the guidance of various kinds of constraints provided by the user. These constraints include the following:

- ***Knowledge type constraints:*** These specify the type of knowledge to be mined, such as association.
- ***Data constraints:*** These specify the set of task-relevant data.
- ***Dimension/level constraints:*** These specify the dimension of the data, or levels of the concept hierarchies, to be used.
- ***Interestingness constraints:*** These specify thresholds on statistical measures of rule interestingness, such as support and confidence.
- ***Rule constraints:*** These specify the form of rules to be mined. Such constraints may be expressed as metarules (rule templates), as the maximum or minimum number of predicates that can occur in the rule antecedent or consequent, or as relationships among attributes, attribute values, and/or aggregates.

The above constraints can be specified using a high-level declarative data mining query language.

The first four of the above types of constraints have already been addressed in earlier parts of this book and chapter. In this section, we discuss the use of rule constraints to focus the mining task. This form of constraint-based mining allows users to specify the rules to be mined according to their intention, thereby making the data mining process more effective. In addition, a sophisticated mining query optimizer can be used to exploit the constraints specified by the user, thereby making the mining process more efficient. Constraint-based mining encourages interactive exploratory mining and analysis.

11.9 METARULE-GUIDED MINING OF ASSOCIATION RULES

Metarules allow users to specify the syntactic form of rules that they are interested in mining. The rule forms can be used as constraints to help improve the efficiency of the mining process. Metarules may be based on the analyst's experience, expectations, or intuition regarding the data, or automatically generated based on the database schema.

Suppose that as a market analyst for HTC, you have access to the data describing customers (such as customer age, address, and credit rating) as well as the list of customer transactions. You are interested in finding associations between customer traits and the items that customers buy. However, rather than finding all of the association rules reflecting these relationships, you are particularly interested only in determining which pairs of customer traits promote the sale of cellphone. A metarule can be used to specify this information describing the form of rules you are interested in finding. An example of such a metarule is

$$P_1(X, Y) \wedge P_2(X, W) \Rightarrow \text{buys}(X, \text{“cellphone”})$$

where P_1 and P_2 are predicate variables that are instantiated to attributes from the given database during the mining process, X is a variable representing a customer, and Y and W take on values of the attributes assigned to P_1 and P_2, respectively.

Typically, a user will specify a list of attributes to be considered for instantiation with P_1 and P_2, otherwise, a default set may be used. In general, a meta rule forms a hypothesis regarding the relationships that the user is interested in probing or confirming. The data mining system can then search for rules that match the given metarule. For instance, the rule given below matches or complies with the metarule given above.

$$\text{age}(X, \text{“30}\ldots\text{39”}) \wedge \text{income}(X, \text{“41000}\ldots\text{60000”}) \Rightarrow \text{buys}(X, \text{“cellphone”})$$

Mining Guided by Additional Rule Constraints

Rule constraints specifying set/subset relationships, constant initiation of variables, and aggregate functions can be specified by the user. These may be used together with, or as an alternative to, metarule-guided mining. In this section, we examine rule constraints as to how they can be used to make the mining process more efficient. Let us see an example where rule constraints are used to mine hybrid-dimension association rules.

Suppose that HTC has a sales multidimensional database with the following interrelated relations:

sales(customer _name, item_name, transaction_id)

lives (customer _name, region, city)

item(item_name, category, price)

transaction (transaction_id, day, month, year)

where lives, item, and transaction are three dimension tables, linked to the fact table sales via three keys, customer _name, item_name, and transaction_id, respectively.

Our association mining query is to "Find the sales of what cheap items (where the sum of the prices is less than Rs.100) that may promote the sales of what expensive items (where the minimum price is Rs.500) in the same category for Chennai customers in 1999. This can be expressed in the DMQL data mining query language as follows, where each line of the query has been enumerated to aid in our discussion.

(1) mine associations as

(2) lives(C,_, "Chennai") $\wedge$ sales^{+}(C, ?{I}, {S}) $\Rightarrow$ sales^{+}(c, ?{J}, {T})

(3) from sales

(4) where S.year = 1999 and T.year = 1999 and I.category = J.category

(5) group by C, I.category

(6) having sum(I.price) <= 100 and min(J.price)>= 500

(7) with support threshold = 1 %

(8) with confidence threshold = 50%

Knowledge type and data constraints are applied before mining. The remaining constraint types could be used after mining, to filter out discovered rules. This, however, may make the mining process very inefficient and expensive.

Rule constraints can be classified into the following five categories with respect to frequent itemset mining: (1) antimonotone, (2) monotone, (3) succinct, (4) convertible, and (5) inconvertible.

A list of these types and their association with commonly used SQL-based constraints is given below:

Constraint	Antimonotone	Monotone	Succinct
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$\min(S) \leq v$	no	yes	yes
$\min(S) \geq v$	yes	no	yes
$\max(S) \leq v$	yes	no	yes
$\max(S) \geq v$	no	yes	yes

Student Activity 1

1. Write short note on:

   a. Mining Multilevel Association Rules from Transactional Databases

   b. Mining Multilevel Association Rules from Relational Data Bases

   c. Constraint – Based Association mining

11.10 SUMMARY

- One of the reasons behind maintaining any database is to enable the user to find interesting patterns and trends in the data.
- The goal of database mining is to automate this process of finding interesting patterns and trends.
- Item sets have minimum support are called frequent item sets.
- The discovery of interesting association relationships among huge amount of business transactions records can help in many business decision making processes such as catalog design, cross marketing and loss leader analysis.
- Market basket analysis may help branch mangers to design different store layouts, take decisions on frequently purchases in order to encourage the sale of items.
- Association rule mining is a two step process i.e. find all frequent itemsets and generate strong association rule from frequent itemsets.
- Association rule that involve two or more dimensions or predicates can be refereed to as multi-dimensional association rule.
- Multi-dimensional association rule with no repeated predicate are called inter dimension association rules.
- Due to ever increasing use of data warehousing and OLAP technology, it is possible that a data cube containing the dimensions of interest to the users may already exist fully materialized.
- Three common binning strategies are equi-width binning, equi-depth binning and homogeneity based binning.

11.11 KEYWORDS

Association rules: Association rules that involve two or more dimensions or predicates can be referred to as multidimensional association rules.

Interdimension association rules: Multidimensional association rules with no repeated predicates are called interdimension association rules.

Knowledge type constraints: These specify the type of knowledge to be mined, such as association.

Data constraints: These specify the set of task-relevant data.

Dimension/level constraints: These specify the dimension of the data, or levels of the concept hierarchies, to be used.

11.12 REVIEW QUESTIONS

1. What is association rule mining? Explain with the help of suitable example.
2. Give description of Market Basket Analysis for Association Rule mining.
3. Explain the Road Map for Association Rule mining.

4. Give a short example to show that items in a strong association rule may actually be negatively correlated.

5. Explain Apriori Algorithm for finding frequent itemsets using candidate generation. Also mention how we can improve the efficiency of Apriori Algorithm?

6. What are various approaches to mining multilevel association rule? Elaborate with the help of suitable example.

11.13 FURTHER READINGS

R. Agrawal, T. Imielinski, A. Swami, "Mining Associations between Sets of Items in Massive Databases'', Proc. of the ACM SIGMOD Int'l Conference on Management of Data, Washington D.C., May 1993, 207-216.

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo, "Fast Discovery of Association Rules", Advances in Knowledge Discovery and Data Mining, Chapter 12, AAAI/MIT Press, 1995.

R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", Proc. of the 20th Int'l Conference on Very Large Databases, Santiago,

R. Srikant, Q. Vu, R. Agrawal, "Mining Association Rules with Item Constraints", Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, California, August 1997.

R. Srikant, R. Agrawal: "Mining Generalized Association Rules", Proc. of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland,

Jong Soo Park, Ming-Syan Chen, Philip S. Yu, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules", IEEE Transactions on Knowledge and Data Engineering, vol. 9, no. 5, pp. 813-825, Sept/Oct 1997.

Mohammed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, "New Algorithms for Fast Discovery of Association Rules", 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97), pp 283-286, Newport Beach, California, August, 1997.

Mohammed Zaki, Mitsunori Ogihara, Srinivasan Parthasarathy, and Wei Li, "Parallel Data Mining for Association Rules on Shared-memory Multi- processors", Supercomputing'96, Pittsburg, PA, Nov 17-22, 1996.

Mohammed Zaki, Srinivasan Parthasarathy, Wei Li, "A Localized Algorithm for Parallel Association Mining", 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp 321-330, Newport, Rhode Island,

A. Amir, R. Feldman, and R. Kashi, "A New and Versatile Method for Association Generation", Principles of Data Mining and Knowledge Discovery, First European Symposium, PKDD'97, Komorowski and Zytkow (eds.), Springer LNAI 1263, pp. 221-231, Trondheim, Norway, 1997.

Charu Aggarwal and Philip Yu, "Mining Large Itemsets for Association Rules", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998.

R. Ng, L. V. S. Lakshmanan, J. Han and A. Pang, "Exploratory Mining and Pruning Optimizations of Constrained Associations Rules", Proc. of 1998 ACM-SIGMOD Conf. on Management of Data, Seattle, Washington, June 1998.

Sergey Brin, Rajeev Motwani, Craig Silverstein, "Beyond Market Baskets: Generalizing Association Rules to Correlations", Proceedings of 1997 ACM SIGMOD, Montreal, Canada, June 1997.

Sergey Brin, Rajeev Motwani, Dick Tsur, Jeffrey Ullman, "Dynamic Itemset Counting and Implication Rules for Market Basket Data", Proceedings of 1997 ACM SIGMOD, Montreal, Canada, June 1997.

Dick Tsur, Jeffrey Ullman, Chris Clifton, Serge Abiteboul, Rajeev Motwani, Svetlozar Nestorov,

Arnie Rosenthal, "Query Flocks: a Generalization of Association-Rule Mining", Proceedings of 1998 ACM SIGMOD, Seattle,

K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Computing Optimized Rectilinear Regions for Association Rules", Proceedings of the Third Conference on Knowledge Discovery and Data Mining (KDD'97), pages 96-103, Los Angels, August 1997

K. Wang, W. Tay, B. Liu, "Interestingness-based Interval Merger for Numeric Association Rules" , Proc. International Conference on Knowledge Discovery and Data Mining (KDD-98), August 1998, New York City.

K. Alsabti, S. Ranka and V. Singh. "A One-Pass Algorithm for Accurately Estimating Quantiles for Disk-Resident Data", In Proc. of VLDB'97

R. Agrawal and J. C. Shafer, "Parallel Mining of Association Rules: Design, Implementation and Experience", IEEE Transactions on Knowledge Data and Engg., 8(6):962-969, December 1996.

Eui-Hong (Sam) Han, George Karypis and Vipin Kumar, "Scalable Parallel Data Mining for Association Rules", Proc. of 1997 ACM-SIGMOD International Conference on Management of Data, May 1997.

T. Shintani and M. Kitsuregawa, "Hash Based Parallel Algorithm for Mining Association Rules", Proceedings of IEEE Fourth International Conference on Parallel and Distributed Information Systems, pp.19-30, 1996.

H. Mannila and H. Toivonen, "Multiple uses of frequent sets and condensed representations", Proc. Second International Conference on Knowledge Discovery and Data Mining (KDD'96), 189-194, Portland, Oregon, August 1996.

H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hdtvnen, and H. Mannila, "Pruning and grouping discovered association rules", In MLnet Workshop on Statistics, Machine Learning, and Discovery in Databases, pp. 47-52, Heraklion, Crete, Greece, April 1995.

Jose Borges and Mark Levene, "Mining Association Rules in Hypertext Databases", Proc. International Conference on Knowledge Discovery and Data Mining (KDD-98), August 1998, New York City.

Mohammed J. Zaki and Mitsunori Ogihara, "Theoretical Foundations of Association Rules", 3rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), Seattle, WA, June 1998.

UNIT

12

CLASSIFICATION AND PREDICTION

L E A R N I N G O B J E C T I V E S

After studying this unit, you should be able to:

- Define data and information.
- Describe the need for information.
- Describe the characteristics of information and information system.
- Describe various categories of information.
- Describe the value of information.
- Describe various levels of information.

U N I T S T R U C T U R E

12.1 INTRODUCTION

A good amount of hidden information is hidden in databases that can be used for making intelligent business decisions. Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends. While classification predicts categorical labels, prediction models continuous-valued functions. For example, a classification model may be built to categorize bank loan applications as either safe or risky, while a prediction model may be built to predict the expenditures of potential customers on computer equipment given their income and occupation. Many classification and prediction methods have been proposed by researchers in machine learning, expert systems, statistics, and neurobiology. Most algorithms are memory resident, typically assuming a small data size. Recent database mining research has built on such work, developing scalable classification and prediction techniques capable of handling large disk-resident data. These techniques often consider parallel and distributed processing.

Data classification is a two-step process. In the first step, a model is built describing a predetermined set of data classes or concepts. The model is constructed by analyzing database tuples described

by attributes. Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the class label attribute. In the context of classification, data tuples are also referred to as samples, examples, or objects. The data tuples analyzed to build the model collectively form the training data set. The individual tuples making up the training set are referred to as training samples and are randomly selected from the sample population. Since the class label of each training sample is provided, this step is also known as supervised learning (i.e., the learning of the model is "supervised" in that it is told to which class each training sample belongs). It contrasts with unsupervised learning (or clustering), in which the class label of each training sample is not known, and the number or set of classes to be learned may not be known in advance.

Usually, the learned model is represented in the form of classification rules, decision trees, or mathematical formulae.

In the second step, the model is used for classification. First, the predictive accuracy of the model (or classifier) is estimated. The holdout method is a simple technique that uses a test set of class-labeled samples. These samples are randomly selected and are independent of the training samples. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. For each test sample, the known class label is compared with the learned model's class prediction for that sample. Note that if the accuracy of the model were estimated based on the training data set, this estimate could be optimistic since the learned model tends to overfit the data (that is, it may have incorporated some particular anomalies of the training data that are not present in the overall sample population). Therefore, a test set is used.

If the accuracy of the model is considered acceptable, the model can be used to classify future data tuples or objects for which the class label is not known. (Such data are also referred to in the machine learning literature as "unknown" or "previously unseen" data.)

Prediction can be viewed as the construction and use of a model to assess the class of an unlabeled sample, or to assess the value or value ranges of an attribute that a given sample is likely to have. In this view, classification and regression are the two major types of prediction problems, where classification is used to predict discrete or nominal values, while regression is used to predict continuous or ordered values. In our view, however, we refer to the use of prediction to predict class labels as classification, and the use of prediction to predict continuous values (e.g., using regression techniques) as prediction.

Classification and prediction have numerous applications including credit approval, medical diagnosis, performance prediction, and selective marketing.

Suppose that we have a database of customers on the HTC mailing list. The mailing list is used to send out promotional literature describing new products and upcoming price discounts. The database describes attributes of the customers, such as their name, age, income, occupation, and credit rating. The customers can be classified as to whether or not they have purchased a computer at HTC.

Suppose that new customers are added to the database and that you would like to notify these customers of an upcoming computer sale. To send out promotional literature to every new customer in the database can be quite costly. A more cost-efficient method would be to target only those new customers who are likely to purchase a new computer. A classification model can be constructed and used for this purpose.

Now suppose instead that you would like to predict the number of major purchases that a customer will make at HTC during a fiscal year. Since the predicted value here is ordered, a prediction model can be constructed for this purpose.

12.2 CLASSIFICATION AND PREDICTION ISSUES

Preparing the Data for Classification and Prediction

The following preprocessing steps may be applied to the data in order to help improve the accuracy, efficiency, and scalability of the classification or prediction process.

- ***Data cleaning:*** This refers to the preprocessing of data in order to remove or reduce noise

(by applying smoothing techniques, for example) and the treatment of missing values (e.g., by replacing a missing value with the most commonly occurring value for that attribute, or with the most probable value based on statistics). Although most classification algorithms have some mechanisms for handling noisy or missing data, this step can help reduce confusion during learning.

- ***Relevance analysis:*** Many of the attributes in the data may be irrelevant to the classification or prediction task. For example, data recording the day of the week on which a bank loan application was filed is unlikely to be relevant to the success of the application. Furthermore, other attributes may be redundant. Hence, relevance analysis may be performed on the data with the aim of removing any irrelevant or redundant attributes from the learning process. In machine learning, this step is known as feature selection. Including such attributes may otherwise slow down, and possibly mislead, the learning step.

  The time spent on relevance analysis, when added to the time spent on learning from the resulting reduced feature subset, should ideally be less than the time that would have been spent on learning from the original set of features. Hence, such analysis can help improve classification efficiency and scalability.

- ***Data transformation:*** The data can be generalized to higher-level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous-valued attributes. For example, numeric values for the attribute income may be generalized to discrete ranges such as low, medium, and high. Similarly, nominal-valued attributes, like street, can be generalized to higher-level concepts, like city. Since generalization compresses the original training data, fewer input/output operations may be involved during learning.

The data may also be normalized, particularly when neural networks or methods involving distance measurements are used in the learning step. Normalization involves scaling all values for a given attribute so that they fall within a small specified range, such as -1.0 to 1.0, or 0.0 to 1.0. In methods that use distance measurements, for example, this would prevent attributes with initially large ranges (like, say, income) from outweighing attributes with initially smaller ranges (such as binary attributes).

Classification and prediction methods can be compared and evaluated according to the following criteria:

- ***Predictive accuracy:*** This refers to the ability of the model to correctly predict the class label of new or previously unseen data.
- ***Speed:*** This refers to the computation costs involved in generating and using the model.
- ***Robustness:*** This is the ability of the model to make correct predictions given noisy data or data with missing values.
- ***Scalability:*** This refers to the ability to construct the model efficiently given large amounts of data.
- ***Interpretability:*** This refers to the level of understanding and insight that is provided by the model.

Student Activity 1

1. List down some of the Classification and Prediction issues.

12.3 INDUCTION OF DECISION TREES

Decision Trees

- A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision.
- For example, we might have a decision tree to help a financial institution decide whether a person should be offered a loan.

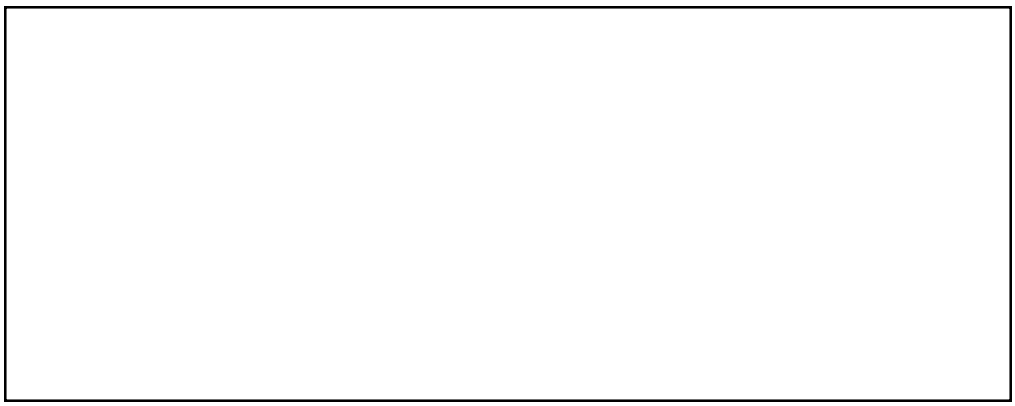


Figure 12.1

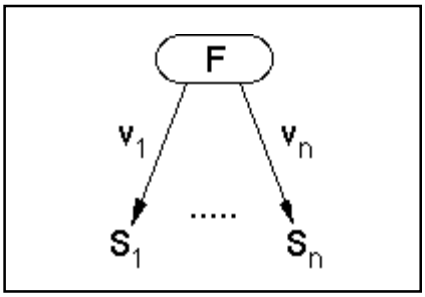
- We wish to be able to induce a decision tree from a set of data about instances together with the decisions or classifications for those instances.

Example Instance Data

size:	small	medium	large
color:	red	blue	green
shape:	brick	wedge	sphere pillar
%% yes			
medium	blue	brick	
small	red	sphere	
large	green	pillar	
large	green	sphere	
%% no			
small	red	wedge	
large	red	wedge	
large	red	pillar	

- In this example, there are 7 instances, described in terms of three features or attributes (size, color, and shape), and the instances are classified into two classes %% yes and %% no.
- We shall now describe an algorithm for inducing a decision tree from such a collection of classified instances.
- Originally termed CLS (Concept Learning System) it has been successively enhanced.
- At the highest level of enhancement that we shall describe in these notes, the system is known as ID3 - later versions include C4, C4.5 and See5/C5.0 (latest version release 1.20, May 2004).
- ID3 and its successors have been developed by Ross Quinlan.

Tree Induction Algorithm

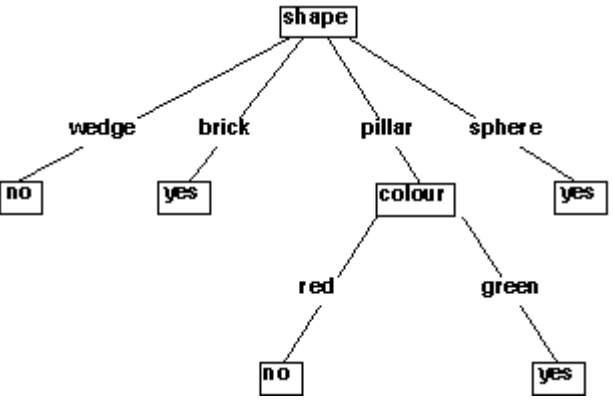


- The algorithm operates over a set of training instances, C.
- If all instances in C are in class P, create a node P and stop, otherwise select a feature or attribute F and create a decision node.
- Partition the training instances in C into subsets according to the values of V.
- Apply the algorithm recursively to each of the subsets C.

Output of Tree Induction Algorithm

This can easily be expressed as a nested if-statement

```
if (shape == wedge)
        return no;
```




```
}
if (shape == sphere)
        return yes;
```

Student Activity 2

1. Describe the various steps of decision tree classification.

12.4 CHOOSING ATTRIBUTES AND ID3

- The order in which attributes are chosen determines how complicated the tree is.
- ID3 uses information theory to determine the most informative attribute.
- A measure of the information content of a message is the inverse of the probability of receiving the message:

  information1(M) = 1/probability(M)

- Taking logs (base 2) makes information correspond to the number of bits required to encode a message:

  information(M) = -log2(probability(M))

Information

- The information content of a message should be related to the degree of surprise in receiving the message.
- Messages with a high probability of arrival are not as informative as messages with low probability.
- Learning aims to predict accurately, i.e. reduce surprise.
- Probabilities are multiplied to get the probability of two or more things both/all happening. Taking logarithms of the probabilities allows information to be added instead of multiplied.

Entropy

- Different messages have different probabilities of arrival.
- Overall level of uncertainty (termed entropy) is:

  $-\sum i\ Pi\ log2Pi$

- Frequency can be used as a probability estimate.
- E.g. if there are 5 positive examples and 3 negative examples in a node the estimated probability of positive is 5/8 = 0.625.

Information and Learning

- We can think of learning as building many-to-one mappings between input and output.
- Learning tries to reduce the information content of the inputs by mapping them to fewer outputs.
- Hence we try to minimise entropy.
- The simplest mapping is to map everything to one output.
- We seek a trade-off between accuracy and simplicity.

Splitting Criterion

- Work out entropy based on distribution of classes.
- Trying splitting on each attribute.
- Work out expected information gain for each attribute.
- Choose best attribute.

Example Part 1

- Initial decision tree is one node with all examples.
- There are 4 positive examples and 3 negative examples
- i.e. probability of positive is 4/7 = 0.57; probability of negative is 3/7 = 0.43
- Entropy is: – (0.57 * log 0.57) – (0.43 * log 0.43) = 0.99
- Evaluate possible ways of splitting.

Example Part 2

- Try split on size which has three values: large, medium and small.
- There are four instances with size = large.
- There are two large positives examples and two large negative examples.
- The probability of positive is 0.5
- The entropy is: – (0.5 * log 0.5) – (0.5 * log 0.5) = 1

Example Part 3

- There is one small positive and one small negative
- Entropy is: – (0.5 * log 0.5) – (0.5 * log 0.5) = 1
- There is only one medium positive and no medium negatives, so entropy is 0.
- Expected information for a split on size is:

$$\left(1 \times \frac{4}{7}\right) + \left(1 \times \frac{2}{7}\right) + \left(0 \times \frac{1}{7}\right) = 0.86$$

Example Part 4

- The expected information gain is: 0.99 – 0.86 = 0.13
- Now try splitting on color and shape.
- Color has an information gain of 0.52
- Shape has an information gain of 0.7
- Therefore split on shape.
- Repeat for all subtree

Summary of Splitting Criterion 1

Some people learn best from an example; others like to see the most general formulation of an algorithm. If you are an "examples" person, don't let the following subscript-studded presentation panic you.

Assume there are k classes C1, ... Ck (k = 2 in our example).

to decide which attribute to split on:

- for each attribute that has not already been used
  - Calculate the information gain that results from splitting on that attribute
  - Split on the attribute that gives the greatest information gain.

Summary of Splitting Criterion 2

to calculate the information gain from splitting N instances on attribute A:

- Calculate the entropy E of the current set of instances.
- for each value aj of the attribute A(j = 1, ..., r)
  - Suppose that there are Jj,1 instances in class C1,…, Jj,k instances in class Ck, for a total of Jjinstances with A = aj.
  - Let qj,1 = Jj,1/Jj, ..., qj,k = Jj,k/Jj;
  - The entropy Ej associated with A = aj is –qj,1 log2(qj,1) ... –qj,k log2(qj,k)

- Now compute E – (J1/N)E1 ... –(Jr/N)Er - this is the information gain associated with a split on attribute A.

Summary of Splitting Criterion 3

to calculate the entropy E of the current set of instances

- Suppose that of the N instances classified to this node, I1 belong to classC1, ..., Ik belong to classCk,
- Let p1 = I1/N, ..., pk = Ik/N,
- Then the initial entropy E is –p1 log2(p1) –p2 log2(p2) ... –pk log2(pk).

Using the iProlog Implementation of ID3

```
% cat example.data
table object(
        texture(smooth, wavy, rough),
        temperature(cold, cool, warm, hot),
        size(small, medium, large),
        class(yes,  no)
)!
object(smooth, cold, large, yes).
object(smooth, cold, small, no).
object(smooth, cool, large, yes).
object(smooth, cool, small, yes).
object(smooth, hot, small, yes).
object(wavy, cold, medium, no).
object(wavy, hot, large, yes).
object(rough, cold, large, no).
object(rough, cool, large, yes).
object(rough, hot, small, no).
object(rough, warm, medium, yes).
```

Using the iProlog Implementation of ID3 - ctd

```
% prolog example.data
iProlog ML (21 March 2003)
: id(object)?
id0
: pp id0!
object(Texture, Temperature, Size, Class) :-
 (Temperature = cold ->
 (Texture = smooth ->
```

(Size = small -> Class = no

| Size = large -> Class = yes)

| Texture = wavy -> Class = no

| Texture = rough -> Class = no)

| Temperature = cool -> Class = yes

| Temperature = warm -> Class = yes

| Temperature = hot ->

(Texture = smooth -> Class = yes

| Texture = wavy -> Class = yes

| Texture = rough -> Class = no)).

:

Windowing

- ID3 can deal with very large data sets by performing induction on subsets or windows onto the data.
    1. Select a random subset of the whole set of training instances.
    2. Use the induction algorithm to form a rule to explain the current window.
    3. Scan through all of the training instances looking for exceptions to the rule.
    4. Add the exceptions to the window
- Repeat steps 2 to 4 until there are no exceptions left.

Noisy Data

- Frequently, training data contains “noise” - i.e. examples which are misclassified, or where one or more of the attribute values is wrong.
- In such cases, one is like to end up with a part of the decision tree which considers say 100 examples, of which 99 are in class C1 and the other is apparently in class C2 (because it is misclassified).
- If there are any unused attributes, we might be able to use them to elaborate the tree to take care of this one case, but the subtree we would be building would in fact be wrong, and would likely misclassify real data.
- Thus, particularly if we know there is noise in the training data, it may be wise to “prune” the decision tree to remove nodes which, statistically speaking, seem likely to arise from noise in the training data.
- A question to consider: How fiercely should we prune?

Expected Error Pruning

- Approximate expected error assuming that we prune at a particular node.
- Approximate backed-up error from children assuming we did not prune.
- If expected error is less than backed-up error, prune.

(Static) Expected Error

- If we prune a node, it becomes a leaf labelled, C.

- What will be the expected classification error at this leaf?

(This is called the Laplace error estimate - it is based on the assumption that the distribution of probabilities that examples will belong to different classes is uniform.)

S is the set of examples in a node

k is the number of classes

N examples in S

C is the majority class in S

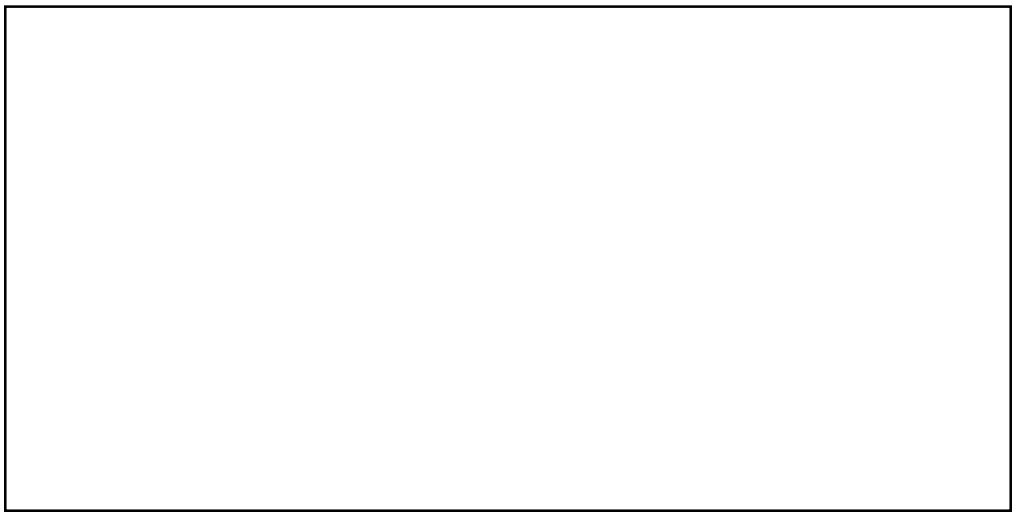
n out of N examples in S belong to C

Backed-up Error

- For a non-leaf node
- Let children of Node be Node1, Node2, etc

- Probabilities can be estimated by relative frequencies of attribute values in sets of examples that fall into child nodes.

Pruning Example



Error Calculation for Pruning Example</FONT< h3>

- Left child of b has class frequencies [3, 2]

- Right child has error of 0.333, calculated in the same way
- Static error estimate E(b) is 0.375, again calculated using the Laplace error estimate formula, with N=6, n=4, and k=2.

- Backed-up error is:

  5/6 and 1/6 because there are 4+2=6 examples handled by node b, of which 3+2=5 go to the left subtree and 1 to the right subtree.

- Since backed-up estimate of 0.413 is greater than static estimate of 0.375, we prune the tree and use the static error of 0.375

BAYESIAN CLASSIFICATION

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

Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. It is made to simplify the computations involved and, in this sense, is considered naive. Bayesian belief networks are graphical models, which unlike naive Bayesian classifiers, allow the representation of dependencies among subsets of attributes. Bayesian belief networks can also be used for classification.

Bayes Theorem

Let X be a data sample whose class label is unknown. Let H be some hypothesis, such as that the data sample X belongs to a specified class C. For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the observed data sample X.

P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X. For example, suppose the world of data samples consists of fruits, described by their color and shape. Suppose that X is red and round, and that H is the hypothesis that X is an apple. Then P(H|X) reflects our confidence that X is an apple given that we have seen that X is red and round. In contrast, P(H) is the prior probability, or a priori probability, of H. For our example, this is the probability that any given data sample is an apple, regardless of how the data sample looks. The posterior probability, P(H|X), is based on more information (such as background knowledge) than the prior probability, P(H), which is independent of X.

Similarly, P(X|H) is the posterior probability of X conditioned on H. That is, it is the probability that X is red and round given that we know that it is true that X is an apple. P(X) is the prior probability of X. Using our example, it is the probability that a data sample from our set of fruits is red and round.

Bayes theorem states that:

NAIVE BAYESIAN CLASSIFICATION

The naive Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Each data sample is represented by an n-dimensional feature vector, $X = (x_1 > x_2, \ldots, x_n)$, depicting n measurements made on the sample from n attributes, respectively, $A_1, A_2, \ldots, A_n$.

2. Suppose that there are m classes, $C_1, C_2, \ldots, C_m$. Given an unknown data sample, X (i.e., having no class label), the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naive Bayesian classifier assigns an unknown sample X to the class Ci if and only if $P(C_i|X) > P(C_j|X)$ for $1 \leq j < m$, $j \neq i$.

Thus we maximize $P(C_i|X)$. The class C_i for which $P(C_i|X)$ is maximized is called the maximum posteriori hypothesis. By Bayes theorem,

$$P(C_i/X) = \frac{P(X/C_i)P(C_i)}{P(X)}.$$

3. As P(X) is constant for all classes, only $P(X|C_i)P(C_j)$ need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1) = P(C_2) = \ldots = P(C_m)$, and we would therefore maximize $P(X|C_i)$. Otherwise, we maximize $P(X|C_i)P(C_i)$. Note that the class prior probabilities may be estimated by $P(C_i) =$ where si is the number of training samples of class C_i, and s is the total number of training samples.

4. Given data sets with many attributes, it would be extremely computationally expensive to compute $P(X|C_i)$. In order to reduce computation in evaluating $P(X|C_i)$, the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the sample, that is, there are no dependence relationships among the attributes. Thus,

$$P(X/C_i) = \prod_{k=1}^{n} P(x_k|C_i).$$

The probabilities $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$ can be estimated from the training samples, where

(a) If A_k is categorical, then $P(x_k|C_i) = \frac{s_{ik}}{s_i}$, where s_{ik} is the number of training samples of class C_i having the value x_k for A_k, and s_i is the number of training samples belonging to C_i.

(b) If A_k is continuous-valued, then the attribute is typically assumed to have a Gaussian distribution so that

$$P(x_k|C_i) = g(x_k, \mu C_i, \sigma C_i) = \frac{1}{\sqrt{2\pi}\sigma C_i} e^{-\frac{(x_k - \mu C_i)^2}{2\sigma C_i^2}},$$

where $g(x_k, \mu C_i, \sigma C_i)$ is the Gaussian (normal) density function for attribute A_k, while mC_i, and $sC_i$are the mean and standard deviation, respectively, given the values for attribute Ak for training samples of class C_i.

5. In order to classify an unknown sample X, $P(X|C_i)P(C_i)$ is evaluated for each class C_i. Sample X is then assigned to the class C_i if and only if

$P(X|C_i)P(C_i) > P(X|C_j)P(C_j)$ for $1 \le j \le m, j \ne i$.

In other words, it is assigned to the class C_i for which $P(X|C_i)P(C_i)$ is the maximum.

In theory, Bayesian classifiers have the minimum error rate in comparison to all other classifiers. However, in practice this is not always the case owing to inaccuracies in the assumptions made for its use, such as class conditional independence, and the lack of available probability data. However, various empirical studies of this classifier in comparison to decision tree and neural network classifiers have found it to be comparable in some domains.

Bayesian classifiers are also useful in that they provide a theoretical justification for other classifiers that do not explicitly use Bayes theorem. For example, under certain assumptions, it can be shown that many neural network and curve-fitting algorithms output the maximum posteriori hypothesis, as does the naive Bayesian classifier.

Let us assume that we wish to predict the class label of an unknown sample using naive Bayesian classification, given the same training data as cited earlier for decision tree induction. The data samples are described by the attributes age, income, student, and credit_rating. The class label attribute, buys_computer, has two distinct values (namely, {yes, no}). Let C_1 correspond to the class buys_computer = "yes" and C_2 correspond to buys_computer = "no'. The unknown sample we wish to classify is

X = (age = "<=30", income = "medium", student = "yes", credit _rating="fair")

We need to maximize $P(X|C_i)P(C)$, for i = 1,2. $P(C_i)$, the prior probability of each class, can be computed based on the training samples:

P(buys_computer = "yes") = 9/14 = 0.643

P(buys_computer = "no") = 5/14 = 0.357

To compute $P(X|C_i)$, for i = 1,2, we compute the following conditional probabilities:

P(age = "<30" | buys_computer = "yes")= 2/9=0.222

P(age = "<30" | buys_computer = "no")= 3/5=0.600

P(income = "medium" | buys_computer = "yes")= 4/9=0.444

P(income = "medium" | buys_computer = "no") = 2/5=0.400

P(student = "yes" | buys_computer = "yes") = 6/9=0.667

P(student = "yes" | buys_computer = "no") = 1/5=0.200

P(credit_rating = "fair" | buys_computer = "yes") = 6/9=0.667

P(credit_rating = "fair" | buysJomputer = "no")= 2/5=0.400

Using the above probabilities, we obtain

P{X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X| buys_computer = "no") = 0.600 x 0.400 x 0.200 x 0.400 = 0.019

P(X| buys_computer = "yes")P(buys_computer = "yes") = 0.044 x 0.643 = 0.028

P(X| buys_computer = "no")P(buys_computer = "no") = 0.019 x 0.357 = 0.007

Therefore, the naive Bayesian classifier predicts buys_computer = "yes" for sample X.

K-NEAREST NEIGHBORS (K-NN) CLASSIFICATION

In k-nearest-neighbor classification, the training dataset is used to classify each member of a "target" dataset. The structure of the data is that there is a classification (categorical) variable of interest ("buyer," or "non-buyer," for example), and a number of additional predictor variables (age, income, location...). Generally speaking, the algorithm is as follows:

1. For each row (case) in the target dataset (the set to be classified), locate the k closest members (the k nearest neighbors) of the training dataset. A Euclidean Distance measure is used to calculate how close each member of the training set is to the target row that is being examined.
2. Examine the k nearest neighbors - which classification (category) do most of them belong to? Assign this category to the row being examined.
3. Repeat this procedure for the remaining rows (cases) in the target set.
4. Additional to this XLMiner™ also lets the user select a maximum value for k, builds models parallelly on all values of k upto the maximum specified value and scoring is done on the best of these models.

Of course the computing time goes up as k goes up, but the advantage is that higher values of k provide smoothing that reduces vulnerability to noise in the training data. In practical applications, typically, k is in units or tens rather than in hundreds or thousands.

See also

- Using k-Nearest Neighbors Classification in XLMiner™
- Example - k-Nearest Neighbors Classification

k-nearest neighbor algorithm

From Wikipedia, the free encyclopedia

(Redirected from)

Jump to: navigation, search

The correct title of this article is k-nearest neighbor algorithm. The initial letter is shown capitalized due to technical restrictions.

In pattern recognition, the k-nearest neighbor algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space.

The training examples are mapped into multidimensional feature space. The space is partitioned into regions by class labels of the training samples. A point in the space is assigned to the class c if it is the most frequent class label among the k nearest training samples. Usually Euclidean distance is used.

The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the actual classification phase, the same features as before are computed for the test sample (whose class is not known). Distances from the new vector to all stored vectors are computed and k closest samples are selected. The new point is predicted to belong to the most numerous class within the set.

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by parameter optimization using, for example, cross-validation. The special case where the class is predicted to be the class of the closest training sample (i.e. when k = 1) is called the nearest neighbour algorithm.

The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the features scales are not consistent with their relevance. Much research effort has been placed into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes.

The algorithm is easy to implement, but it is computationally intensive, especially when the size of the training set grows. Many optimizations have been proposed over the years; these generally seek to reduce the number of distances actually computed. Some optimizations involve partitioning the feature space, and only computing distances within specific nearby volumes. Several different types of nearest neighbour finding algorithms include:

- Linear scan
- Kd-trees
- Balltrees
- Metric trees
- Locality-sensitive hashing (LSH)

The nearest neighbor algorithm has some strong consistency results. As the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the

Bayes error rate (the minimum achievable error rate given the distribution of the data). k-nearest neighbor is guaranteed to approach the Bayes error rate, for some value of k (where k increases as a function of the number of data points).

12.6 CLASSIFICATION BY BACK-PROPAGATION

Backpropagation is a neural network learning algorithm. The field of neural networks was originally kindled by psychologists and neurobiologists who sought to develop and test computational analogues of neurons. Roughly speaking, a neural network is a set of connected input/output units where each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input samples. Neural network learning is also referred to as connectionist learning due to the connections between units.

x_1	x_2	x_3	w_{14}
1	0	1	0.2

Neural networks involve long training times and are therefore more suitable for applications where this is feasible. They require a number of parameters that are typically best determined empirically, such as the network topology or "structure:' Neural networks have been criticized for their poor interpretability, since it is difficult for humans to interpret the symbolic meaning behind the learned weights. These features initially made neural networks less desirable for data mining.

Advantages of neural networks, however, include their high tolerance to noisy data as well as their ability to classify patterns on which they have not been trained. In addition, several algorithms have recently been developed for the extraction of rules from trained neural networks. These factors contribute towards the usefulness of neural networks for classification in data mining.

Algorithm: Neural network learning for classification, using the backpropagation algorithm.

Input: The training samples, samples; the learning rate, l; a multilayer feed-forward network, network.

Output: A neural network trained to classify the samples. Method:

(1) Initialize all weights and biases in network;

(2) while terminating condition is not satisfied {

(3) for each training sample X in samples {

(4) // Propagate the inputs forward:

(5) for each hidden or output layer unit j {

(6) $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit j with respect to the previous layer, i

(7) $O_j = \frac{1}{1+e^{-1}}$; } // compute the output of each unit j

(8) //Backpropagate the errors:

(9) for each unit j in the output layer

(10) Errj = $O_i(1 - O_j)(T_j - O_j)$; //compute the error

(11) for each unit j in the hidden layers, from the last to the first hidden layer

(12) $Err_j = O_j(l - O_j)$ .....; //compute the error with respect to the next higher layer, k

(13) for each weight W_{ij} in network I

(14) $\Delta w_{ij} = (l)Err_j O_i$; //weight increment

(15) $w_{ij} = w_{ij} + \Delta w_{ij}$;} //weight update

(16) for each bias θ_j in network {

(17) $\Delta \theta_j = (l)Err_j$; //bias increment

(18) $\theta_j = \theta_j + \theta_j$; } //bias update

(19) } }

Sample calculations for learning by the backpropagation algorithm shows a multilayer feed-forward neural network. Let the learning rate be 0.9. The initial weight and bias values of the network are given in Table below, along with the first training sample, X = (1,0, 1), whose class label is 1.

The net input and output calculations:

Unit j	Net input, I_j	Output, O_j
4	0.2+0-0.5-0.4=-0.7	$1/(1+e^{0.7})$=0.332
5	-0.3+0+0.2+0.2=0.1	$1/(1+e^{-0.7})$=0.525
6	(-0.3) (0.332) - (0.2) (0.525)+0.1= - 0.105	$1/(1+e^{0.105})$=0.474

Calculation of the error at each node.

Unit j Err;

6 (0.474)(1 - 0.474)(1 - 0.474) = 0.1311

5 (0.525)(1 - 0.525)(0.1311)(-0.2) = -0.0065

4 (0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087

Calculations for weight and bias updating.

Weight or bias	New value
w_{24}	-0.3 + (0.9)(0.1311)(0.332) = -0.261
w_{56}	-0.2 + (0.9)(0.1311)(0.525) = -0.138
w_{14}	0.2 + (0.9)(-0.0087)(1) = 0.192
w_{15}	-0.3 + (0.9)(-0.0065)(1) == -0.306
w_{24}	0.4 + (0.9)(-0.0087)(0) = 0.4
w_{25}	0.1 + (0.9)(-0.0065)(0) = 0.1
w_{34}	-0.5 + (0.9)(-0.0087)(1) = -0.508
w_{35}	0.2 + (0.9)(-00065)(1) = 0.194
θ_6	0.1 + (0.9)(0.1311) = 0.218
θ_5	0.2 + (0.9)(-0.0065) = 0.194
θ_4	-0.4 + (0.9)(-0.0087) = -0.408

Several variations and alternatives to the backpropagation algorithm have been proposed for classification in neural networks. These may involve the dynamic adjustment of the network topology and of the learning rate or other parameters, or the use of different error functions.

A major disadvantage of neural networks lies in their knowledge representation. Acquired knowledge in the form of a network of units connected by weighted links is difficult for humans to interpret. This factor has motivated research in extracting the knowledge embedded in trained neural networks and in representing that knowledge symbolically. Methods include extracting rules from networks and sensitivity analysis.

Various algorithms for the extraction of rules have been proposed. The methods typically impose restrictions regarding procedures used in training the given neural network, the network topology, and the discretization of input values.

Fully connected networks are difficult to articulate. Hence, often the first step towards extracting rules from neural networks is network pruning. This consists of removing weighted links that do not result in a decrease in the classification accuracy of the given network.

12.7 CLASSIFICATION BY ASSOCIATION RULES

Association rule mining is an important and highly active area of data mining research. Recently, data mining techniques have been developed that apply concepts used in association rule mining to the problem of classification. In this section, we study three methods in historical order. The first two, ARCS and associative classification, use association rules for classification. The third method, CAEP, mines "emerging patterns" that consider the concept of support used in mining associations.

The first method mines association rules based on clustering and then employs the rules for classification. The ARCS, or Association Rule Clustering System, mines association rules of the form $A_{quan1} \wedge A_{quan2} => A_{cat}$, where A_{quan1} and A_{quan2} are tests on quantitative attribute ranges (where the ranges are dynamically determined), and Acat assigns a class label for a categorical attribute from the given training data. Association rules are plotted on a 2-D grid. The algorithm scans the grid, searching for rectangular clusters of rules. In this way, adjacent ranges of the quantitative attributes occurring within a rule cluster may be combined. The clustered association rules generated by ARCS were applied to classification, and their accuracy was compared to C4.5. In general, ARCS was empirically found to be slightly more accurate than C4.5 when there are outliers in the data. The accuracy of ARCS is related to the degree of discretization used. In terms of scalability, ARCS requires a constant amount of memory, regardless of the database size. C4.5 has exponentially higher execution times than ARCS, requiring the entire database, multiplied by some factor, to fit entirely in main memory.

The second method is referred to as associative classification. It mines rules of the form condset => y, where condset is a set of items (or attribute-value pairs) and y is a class label. Rules that satisfy a prespecified minimum support are frequent, where a rule has support s ifs% of the samples in the given data set contain condset and belong to class y. A rule satisfying minimum confidence is called accurate, where a rule has confidence c if c% of the samples in the given data set that contain condset belong to class y. If a set of rules has the same condset, then the rule with the highest confidence is selected as the possible rule (PR) to represent the set.

The third method, CAEP (classification by aggregating emerging patterns), uses the notion of item set support to mine emerging patterns (EPs), which are used to construct a classifier. Roughly speaking, an EP is an itemset (or set of items) whose support increases significantly from one class of data to another. The ratio of the two supports is called the growth rate of the EP. For example, suppose that we have a data set of customers with the classes buys_computer = "yes", or C_1, and buys_computer = "no", or C_2. The item set {age = "$\leq$ 30"; student = "no"} is a typical EP, whose support increases from 0.2% in C_1 to 57.6% in C_2 at a growth rate of $\frac{57.6\%}{0.2\%} = 28.8$.

Note that an item is either a simple equality test on a categorical attribute, or a membership test checking whether a numerical attribute is in an interval. Each EP is a multiattribute test and can be very strong at differentiating instances of one class from another. For instance, if a new sample X contains the above EP, then with odds of 99.6% we can claim that X belongs to C_2. In general, the differentiating power of an EP is roughly proportional to its growth rate and its support in the target class.

12.8 OTHER CLASSIFICATION METHODS

Other than the methods discussed earlier, there are many more methods available for classification. Some of them are briefly discussed below.

k-Nearest Neighbor Classifiers

Nearest neighbor classifiers are based on learning by analogy. The training samples are described by n-dimensional numeric attributes. Each sample represents a point in an n-dimensional space.

In this way, all of the training samples are stored in an n-dimensional pattern space. When given an unknown sample, a k-nearest neighbor classifier searches the pattern space for the k training samples that are closest to the unknown sample. These k training samples are the k "nearest neighbors" of the unknown sample. "Closeness" is defined in terms of Euclidean distance, where the Euclidean distance between two points, $X = (x_1, x_2, \ldots, x_n)$ and $Y=(y_1, y_2, \ldots, y_n)$ is

$$d(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$

The unknown sample is assigned the most common class among its k nearest neighbors. When k = 1, the unknown sample is assigned the class of the training sample that is closest to it in pattern space.

Nearest neighbor classifiers are instance-based or lazy learners in that they store all of the training samples and do not build a classifier until a new (unlabeled) sample needs to be classified. This contrasts with eager learning methods, such as decision tree induction and backpropagation, which construct a generalization model before receiving new samples to classify. Lazy learners can incur expensive computational costs when the number of potential neighbors (i.e., stored training samples) with which to compare a given unlabeled sample is great. Therefore, they require efficient indexing techniques. As expected, lazy learning methods are faster at training- than eager methods, but slower at classification since all computation is delayed to that time. Unlike decision tree induction and backpropagation, nearest neighbor classifiers assign equal weight to each attribute. This may cause confusion when there are many irrelevant attributes in the data.

Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown sample. In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown sample.

Case-Based Reasoning

Case-based reasoning (CBR) classifiers are instanced-based. Unlike nearest neighbor classifiers, which store training samples as points in Euclidean space, the samples or "cases" stored by CBR are complex symbolic descriptions. Business applications of CBR include problem resolution for customer service help desks, for example, where cases describe product-related diagnostic problems. CBR has also been applied to areas such as engineering and law, where cases are either technical designs or legal rulings, respectively.

When given a new case to classify, a case-based reasoner will first check if an identical training case exists. If one is found, then the accompanying solution to that case is returned. If no identical case is found, then the case-based reasoner will search for training cases having components that are similar to those of the new case. Conceptually, these training cases may be considered as neighbors of the new case. If cases are represented as graphs, this involves searching for sub graphs that are similar to subgraphs within the new case. The case-based reasoner tries to combine the solutions of the neighboring training cases in order to propose a solution for the new case. If incompatibilities arise with the individual solutions, then backtracking to search for other solutions may be necessary. The case-based reasoner may employ background knowledge and problem-solving strategies in order to propose a feasible combined solution.

Challenges in case-based reasoning include finding a good similarity metric (e.g., for matching subgraphs), developing efficient techniques for indexing training cases, and methods for combining solutions.

Genetic Algorithms

Genetic algorithms attempt to incorporate ideas of natural evolution. In general, genetic learning starts as follows. An initial population is created consisting of randomly generated rules. Each rule can be represented by a string of bits. As a simple example, suppose that samples in a given training set are described by two Boolean attributes, A_1 and A_2, and that there are two classes, C_1 and C_2. The rule "IF A_1 AND NOT A_2 THEN C_2" can be encoded as the bit string "100", where the two leftmost bits represent attributes A_1 and A_2, respectively, and the rightmost bit represents the class. Similarly, the rule "IF NOT A_1 AND NOT A_2 THEN C_1" can be encoded as "001". If an

attribute has k values, where k > 2, then k bits may be used to encode the attribute's values. Classes can be encoded in a similar fashion.

Based on the notion of survival of the fittest, a new population is formed to consist of the fittest rules in the current population, as well as offspring of these tules. Typically, the fitness of a rule is assessed by its classification accuracy on a set of training samples.

Offspring are created by applying genetic operators such as crossover and mutation. In crossover, substrings from pairs of rules are swapped to form new pairs of rules. In mutation, randomly selected bits in a rule's string are inverted.

The process of generating new populations based on prior populations of rules continues until a population P "evolves" where each rule in P satisfies a prespecified fitness threshold.

Genetic algorithms are easily parallelizable and have been used for classification as well as other optimization problems. In data mining, they may be used to evaluate the fitness of other algorithms.

Rough Set Approach

Rough set theory can be used for classification to discover structural relationships with in imprecise or noisy data. It applies to discrete-valued attributes. Continuous-valued attributes must therefore be discretized prior to its use.

Rough set theory is based on the establishment of equivalence classes within the given training data. All of the data samples forming an equivalence class are indiscernible, that is, the samples are identical with respect to the attributes describing the data. Given real-world data, it is common that some classes cannot be distinguished in terms of the available attributes. Rough sets can be used to approximately or "roughly" define such classes. A rough set definition for a given class C is approximated by two sets-a lower approximation of C and an upper approximation of C. The lower approximation of C consists of all of the data samples that, based on the knowledge of the attributes, are certain to belong to C without ambiguity. The upper approximation of C consists of all of the samples that, based on the knowledge of the attributes, cannot be described as not belonging to C. Decision rules can be generated for each class. Typically, a decision table is used to represent the rules.

Rough sets can also be used for feature reduction (where attributes that do not contribute towards the classification of the given training data can be identified and removed) and relevance analysis (where the contribution or significance of each attribute is assessed with respect to the classification task). The problem of finding the minimal subsets (reducts) of attributes that can describe all of the concepts in the given data set is NP-hard. However, algorithms to reduce the computation intensity have been proposed. In one method, for example, a discernibility matrix is used that stores the differences between attribute values for each pair of data samples. Rather than searching on the entire training set, the matrix is instead searched to detect redundant attributes.

Fuzzy Set Approaches

Rule-based systems for classification have the disadvantage that they involve sharp cutoffs for continuous attributes. For example, consider the following rule for customer credit application approval. The rule essentially says that applications for customers who have had a job for two or more years and who have a high income (i.e., of at least $50K) are approved:

IF (years_employed >= 2) $\wedge$ (income ³ 50,000) THEN credit = "approved"

By this rule, a customer who has had a job for at least two years will receive credit if her income is, say, Rs.50000, but not if it is Rs.49000. Such harsh thresholding may seem unfair. Instead, fuzzy logic can be introduced into the system to allow "fuzzy" thresholds or boundaries to be defined. Rather than having a precise cutoff between categories or sets, fuzzy logic uses truth-values between 0.0 and 1.0 to represent the degree of membership that a certain value has in a given category. Hence, with fuzzy logic, we can capture the notion that an income of Rs.49000 is, to some degree, high, although not as high as an income of Rs.50000.

12.9 SUMMARY

- Prediction can be viewed as the construction and use of a model to assess the class of unlabeled samples or to assess the value or value ranges of an attribute.
- Data Classification is a process in which a model is built describing a pre-determine set of data classes or concepts.
- Classification and prediction methods can be compared and evaluated according to predictive accuracy, speed, robustness, scalability and interpretability.
- In tree pruning approach, a tree is pruned by halting its construction earlier and node becomes a leaf.
- Back Propagation is a neural network learning algorithm that is a set of connected input/ output units where each connection has a weight associated with it.
- The advantage of neural network is high tolerance to noisy data as well as their ability to classify patterns on which they have not been trained.
- A major disadvantage of neural network is that the acquired knowledge in the form of network of units connected by weighted links is difficult for human to intercept.
- In k-nearest-neighbor classification, the training dataset is used to classify each member of a "target" dataset.
- A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision.
- We can think of learning as building many-to-one mappings between input and output.

12.10 KEYWORDS

- ***Predictive accuracy:*** This refers to the ability of the model to correctly predict the class label of new or previously unseen data.
- ***Speed:*** This refers to the computation costs involved in generating and using the model.
- ***Robustness:*** This is the ability of the model to make correct predictions given noisy data or data with missing values.
- ***Scalability:*** This refers to the ability to construct the model efficiently given large amounts of data.
- ***Interpretability:*** This refers to the level of understanding and insight that is provided by the model.

12.11 REVIEW QUESTIONS

1. Compare the advantages and disadvantages of eager classification (e.g. decision tree, Bayesian, neural network) versus lazy classification (e.g. k- nearest neighbor, case-based reasoning).
2. Write an algorithm for k-nearest neighbor classification given k and n, the number of attributes describing each sample.
3. What is boosting? State why it may improve the accuracy of decision tree induction.
4. Write short note on:
   a. Tree Pruning
   b. Bayesian Classification
   c. Classification by Back- Propagation
   d. Genetic Algorithm

12.12 FURTHER READINGS

Adriaans, P. and Zantige, D. Data Mining, Harlow, UK: Addison-Wesley, 1996

Berry, M.J.A. and Linoff, G., Data Mining Techniques for Marketing, Sales, and Customer Support, New York, NY: John Wiley and Sons, 1997

Bishop, C. Neural Networks for Pattern Recognition, Oxford, UK: Clarendon Press, 1995

Breiman, L., Fredman, J., Olshen, R.A., and Stone, C.J., Classification and Regression Trees, Monterey, CA: Wadsworth & Brooks, 1984

Chester, M., Neural Networks: A Tutorial, Englwood cliffs, NJ, Prentice Hall, 1993

Cressie, N., Statistics for Spatial Data, Revised Edition, Wiley, New York, 1993

Fayyad, U.M.; Piatestsky-Shapiro G.; Smyth D.; and Uthurusamy R., Advances in Knowledge Discovery and Data Mining, Cambridge, MA: AAAI Press/MIT Press, 1996

Fukunaga, K., Introduction to Statistical Pattern Recognition, Academic Press, 1990

Groth, R., Data Mining, A Hands-On Approach for business Professionals, Prentice Hall, 1998 [with demo software]

Hand, D.J., Mannila, H., Smyth, P., Principles of Data Mining, MIT Press, [late] 2000

Jain, A. and Dubes, R., Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988

Jordan, M.I., Learning in Graphical Models, MIT Press, 1999

Kargupta, Hillol, and Chan, Philip, Advances in Distributed and Parallel Knowledge Discovery, MIT/AAAI Press, 2000

Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs," New York: Springer-Verlag, 1994

Mitchell, T.M., Machine Learning, New York, NY: McGrawl Hill, 1997

Ripley, B.D., Pattern Recognition and Neural Networks, Cambridge, UK: Cambridge University Press, 1996

Dick Tsur, Jeffrey Ullman, Chris Clifton, Serge Abiteboul, Rajeev Motwani, Svetlozar Nestorov, Arnie Rosenthal, "Query Flocks: a Generalization of Association-Rule Mining", Proceedings of 1998 ACM SIGMOD, Seattle,

K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Computing Optimized Rectilinear Regions for Association Rules", Proceedings of the Third Conference on Knowledge Discovery and Data Mining (KDD'97), pages 96-103, Los Angels, August 1997

K. Wang, W. Tay, B. Liu, "Interestingness-based Interval Merger for Numeric Association Rules" , Proc. International Conference on Knowledge Discovery and Data Mining (KDD-98), August 1998, New York City.

Michael Berry and Gordon Linoff, "Data Mining Techniques (For Marketing, Sales, and Customer Support), John Wiley & Sons, 1997.

Sholom M. Weiss and Nitin Indurkhya, "Predictive Data Mining: A Practical Guide", Morgan Kaufmann Publishers, 1998.

Alex Freitas and Simon Lavington, "Mining Very Large Databases with Parallel Processing", Kluwer Academic Publishers, 1998.

Adriaans, P. and Zantige, D. Data Mining, Harlow, UK: Addison-Wesley, 1996

Berry, M.J.A. and Linoff, G., Data Mining Techniques for Marketing, Sales, and Customer Support, New York, NY: John Wiley and Sons, 1997