MODERN DATA BASE MANAGEMENT (DSCSC31)

(BSC COMPUTER SCIENCE-III)

ACHARYA NAGARJUNA UNIVERSITY

CENTRE FOR DISTANCE EDUCATION

NAGARJUNA NAGAR,

GUNTUR

ANDHRA PRADESH

Lesson- 1

THE DATABASE ENVIRONMENT

1.0 Objective

To be able to

- Explain why database will continue to grow in number and into the next century.
- Defining data, metadata, database, database management system
- Name several limitations of conventional file processing systems.
- Identify five categories of databases, and several key decisions that must be made for each category.
- Explain at least six advantages of the database approach, compared to traditional file processing.
- List and briefly describe nine components of a typical database environment.
- Briefly describe the evolution of database systems

Structure

1.1 Introduction

Any organization uses a computer to store and process information because it hopes for speed, accuracy, efficiency, economy etc., beyond what could be achieved using clerical methods. The objectives of using a Database Management System (DBMS) must in essence be the same although the justifications may be more indirect. Early computer applications were based on existing clerical methods and stored information was partitioned in much the same way as manual files. But the computer's processing speed gave a potential for RELATING data from different sources to produce valuable management information, provided that some standardization could be imposed over departmental boundaries. In this Lesson we introduce the basic concepts of databases and database management system (DBMS). We describe traditional file management systems and some of their shortcomings that led to the database approach.

1.2 Basic Concepts and Definitions

We define a database as an organized collection of logically related data. A database may be of any size and complexity. For example, a salesperson may maintain a small database of customer contacts on her laptop computer that consists of a few megabytes of data. A large corporation may build a very large database consisting of several terabytes of data on a large mainframe computer that is used for decision support applications (Winter, 1997). Very large data warehouses contain more than a peta byte of data .

Data

Historically, the term data referred to known facts that could be recorded and stored on computer media. For example in a salesperson's database, the data would include facts such as customer name, address, and telephone number. This definition now needs to be expanded to reflect a new reality. Databases today are used to store objects such as documents, photographic images, sound, and even video segments, in addition to conventional textual and numeric data. For example, the salesperson's database might include a photo image of the customer contact. It might also include a sound recording or video clip of the most recent conversation with the customer. To reflect this reality, we use the following broadened definition: Data consist of facts, text, graphics, images, sound, and video segments that have meaning in the user's environment.

We have defined a database as an organized collection of related data. By organized we mean that the data are structured so as to be easily stored, manipulated, and retrieved by users. By related we mean that the data describe a domain of interest to a group of users and that the users can use the data to answer questions concerning that domain.

Data Versus Information

The terms data and information are closely related, and in fact are often used interchangeably. However, it is often useful to distinguish between data and information. We define information as data that has been processed in such a way that it can increase the knowledge of the person who uses it. For example, consider the following list of facts:

Baker, Kenneth D.	324917628
Doyle, Joan E.	476193248
Finkle, Clive R.	548429344
Lewis, John C.	551742186
McFerran, Debra R.	409723145
Sisneors, Michael	392416582

These facts satisfy our definition of data, but most persons would agree that the data are useless in their present form. Even if we guess that this is a list of persons' names together with their Social Security numbers, the data remain useless since we have no idea what the entries mean. Notice what happens when we place the same data in a context, as shown in Figure 1.1a. By adding a few additional data items and providing some structure, we recognize a class roster for a particular course. This is useful information to some users, such as the course instructor and the registrar's office.

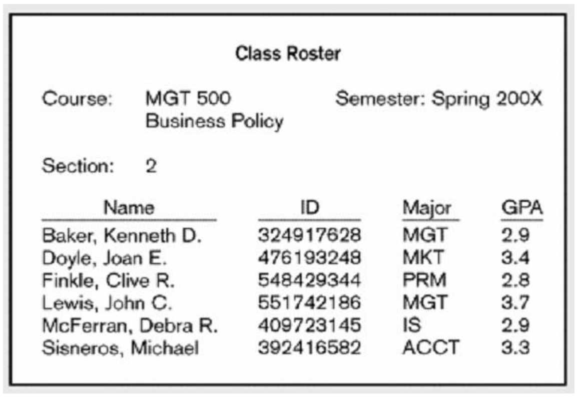
Another way to convert data into information is to summarize them or otherwise process and present them for human interpretation. For example, Figure 1.1b shows summarized student enrollment data presented as graphical information. This information could be used as a basis for deciding whether to add new courses or to hire new faculty members.

In practice, databases today may contain either data or information (or both), according to our definitions. For example, a database may contain an image of the class roster document shown in Figure 1.1a. Also, data are often preprocessed and stored in summarized form in databases that are used for decision support. Throughout this text we use the term *database* without distinguishing its contents as data or information.

As we have indicated, data only become useful when placed in some context. The primary mechanism for providing context for data is metadata. Metadata are data that describe the properties or characteristics of other data. Some of these properties include data definitions, data structures, and rules of constraints.

Some sample metadata for the Class Roster (Figure 1.1a) are listed in Table 1.1. For each data item that appears in the Class Roster, the metadata show the data item name, the data type, length, minimum and maximum allowable values (where appropriate), and a brief description of each data item. Notice the distinction between data and metadata: Metadata are once removed from data. That is, metadata describe the properties of data but do not include that data. Thus, the metadata shown in Table 1.1 do not include any sample data from the Class Roster of Figure 1.1a. Metadata allow database designers and users to understand what data exist, what the data mean, and what the fine distinctions are between seemingly similar data items. The management f metadata is at least as crucial as managing the associated data since data without clear meaning can be confusing, misinterpreted, or erroneous.

Figure 1.1 Converting data to information (a) Data in context



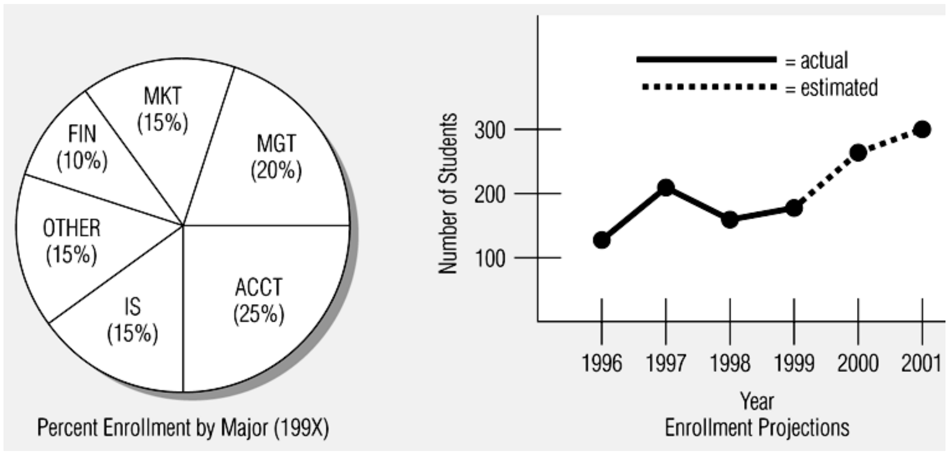
Class Roster

Course: MGT 500 Business Policy Semester: Spring 200X

Section: 2

Name	ID	Major	GPA
Baker, Kenneth D.	324917628	MGT	2.9
Doyle, Joan E.	476193248	MKT	3.4
Finkle, Clive R.	548429344	PRM	2.8
Lewis, John C.	551742186	MGT	3.7
McFerran, Debra R.	409723145	IS	2.9
Sisneros, Michael	392416582	ACCT	3.3

(b) Summarized data



Meta Data

As we have indicated, data only become useful when placed in some context. The primary mechanism for providing context for data is metadata. Metadata are data that describe the properties or characteristics of other data. Some of these properties include data definitions, data structures, and rules or constraints.

Some sample metadata, for the class roster (Figure 1.1a) are listed in Table1.1. For each data item that appears in the class Roster, the metadata show the data item name, the data type, length, minimum and maximum allowable values, and a brief description of each data item. Notice the distinction between data and metadata: metadata are once removed from data. That is, metadata describe the properties of data but do not include that data. Thus, the metadata shown in Table 1.1 do not include any sample data from the Class Roster of a Figure 1.1a. Metadata allow database designers and users to understand what data exist, what the data mean, and what the fine distinctions are between seemingly similar data items. The management of metadata is at least as crucial as managing the associated data since data without clear meaning can be confusing, misinterpreted, or erroneous.

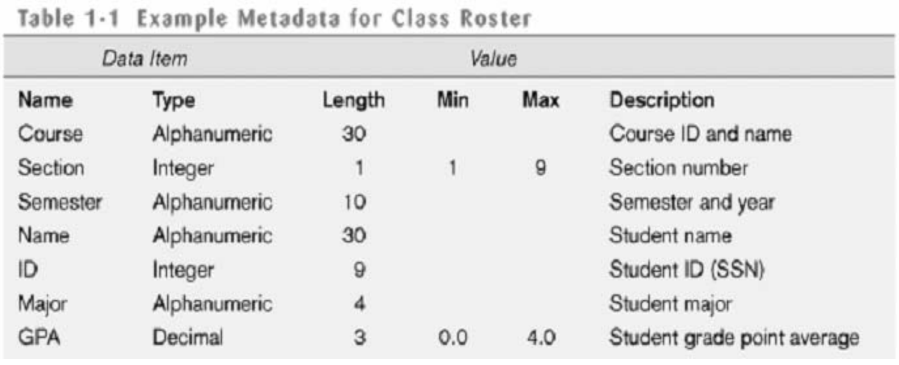


Table 1-1 Example Metadata for Class Roster

Data Item		*Value*			
Name	**Type**	**Length**	**Min**	**Max**	**Description**
Course	Alphanumeric	30			Course ID and name
Section	Integer	1	1	9	Section number
Semester	Alphanumeric	10			Semester and year
Name	Alphanumeric	30			Student name
ID	Integer	9			Student ID (SSN)
Major	Alphanumeric	4			Student major
GPA	Decimal	3	0.0	4.0	Student grade point average

1.3 Traditional File Processing System

In the beginning of computer-based data processing, there were no databases. To be useful for business applications, computers must be able to store, manipulate, and retrieve large files of data. So, an organization's information was stored as groups of records in separate files. Computer file processing systems were developed for this purpose. Although these systems have evolved over time, their basic structure and purpose have changed little over several decades.

As business applications became more complex, it became evident that traditional file processing systems had a number of shortcomings and limitations (described below). As a result, these systems have been replaced by database processing systems in more critical business applications today. Nevertheless, you should have at least some familiarity with file processing systems for the following reasons:

1. File processing systems are still widely used today, especially for backing up database systems.
2. Understanding the problems and limitations inherent in file processing systems can help us avoid these same problems when designing database systems.

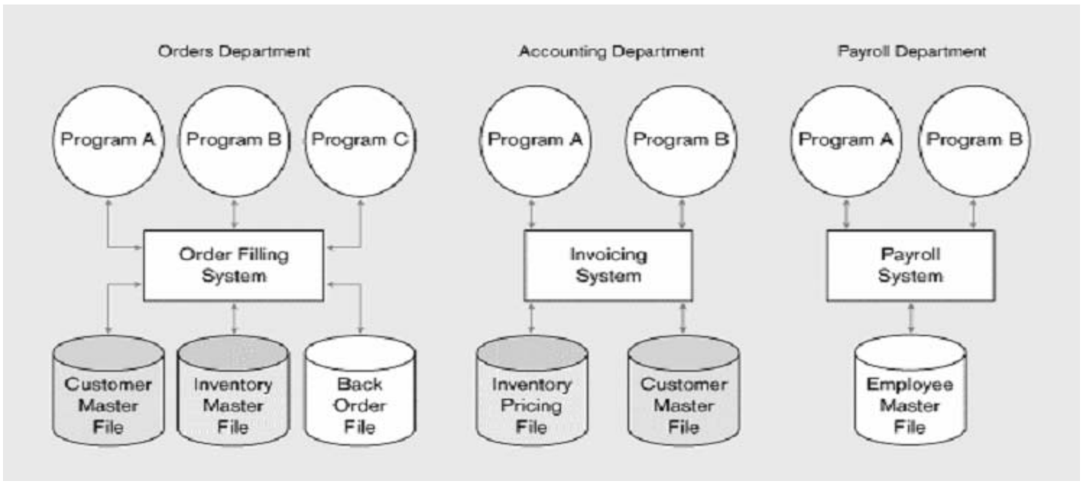
In the remainder of this section we describe file processing systems and discuss their limitations by means of a realistic case example. In the next section we use the same case example to introduce and compare database processing systems.

1.4 File Processing Systems at Pine Valley Furniture Company

Pine Valley Furniture Company manufactures high-quality, all-wood furniture and distributes it to retail stores nationwide. Among the firm's several product lines are computer desks, entertainment centers, dinette sets, bookcases, and wall units. Customers submit orders to Pine Valley Furniture by any of several means; telephone, mail, fax, or electronic forms via the Internet. The company employs a total of about 100 persons at the present time and is experiencing rapid growth.

Early computer applications at Pine Valley Furniture used the traditional file processing approach. This approach to information systems design focused on the data processing needs of individual departments, instead of evaluating the overall information needs of the organization. The information systems group typically responds to users' requests for new systems by developing (or acquiring) new computer programs for individual applications such as inventor control, accounts receivable, or human resource management. Each application program or system that is developed is designed to meet the needs of the particular requesting department or user group. Thus there is no overall map, plan, or model to guide the growth of applications, Three of the computer applications based on the file processing approach are shown in Figure 1.2. The systems illustrated are Order Filling, Invoicing, and Payroll. The figure also shows the major data files associated with each application. A *file* is a collection of related records.

Figure 1.2 Three file Processing systems at Pine Valley Furniture



1.4.1. Disadvantages of File Processing System

Program data dependence*:* In traditional file processing, file descriptions are to be specified with in each application program that access those files. If file structured is changed , every program that access that file is to be modified.

Ex : A custom file may be accessed by various programs.

Suppose it is decided to change the customer address field length from 30 to 40 characters, the customer file description in each program is to be modified. It is difficult to locate and modify the field description in all programs.

Duplication of data: In traditional file processing system the applications are developed independently. For example ORDER file system contains INVENTORY master file, the INVOICING systems have the inventory-pricing file. These two files having the common data descriptions like unit price, quantity on hand. This duplication is wasteful since it requires additional storage space. It causes the loss of data integrity and it also causes loss of metadata integrity.

Limited data sharing*:* Traditional file processing allows limited chance of sharing the data outside their own application. For example in the accounting department have access to the INVOICING system and its file, but they do not have access to the ORDER FILLING system or the PAYROLL SYSTEM and their files.

Lengthy development times: In Traditional file processing Each new application program requires declaration of all entities, and their formats at the beginning. This will causes in increase programming time and which is not required in fastest business world.

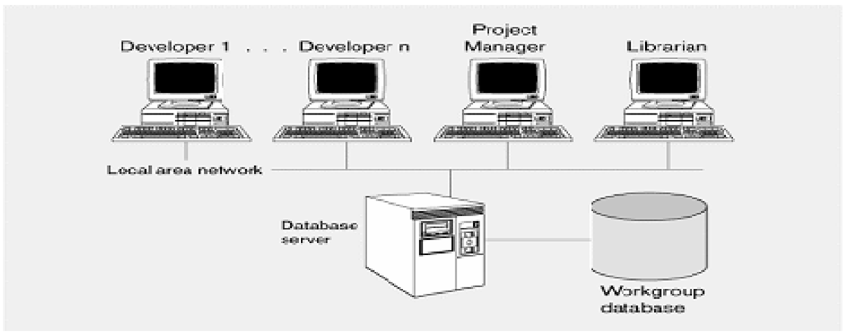
Excessive program maintenance*:* In Traditional file processing all the departments in an organization maintains their programs independently. So, every dept. should bare 80% of their budget amount for their program maintenance . Automatically this may causes the little opportunity for developing new application.

1.5 The Range of Database Applications

Personal Databases: Personal databases are designed to support one user. For example the simple database application that store customer information and the details of contacts of each customer. These databases reside on PC's or PDA (Personal Digital Assistant). This information can be transferred from one device to another for backup and work purpose.

Workgroup Databases: A workgroup is a relatively small team of people who are on the same project or application. The workgroup may contain 25 persons. A workgroup database is designed to support the collaborate efforts of such a team. The workgroup needs a database that will track each item as it is developed and allow the data to be easily shared by the team members each member of the workgroup has a computer and computers are linked with LAN. The database stored in a central device called database server. Each member of a workgroup has access to the shared data.

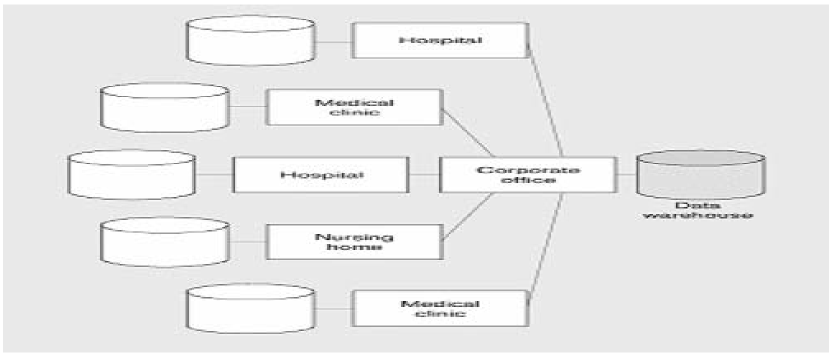
Figure 1.8 Workgroup database with local area network



Department Databases : A department is a functional unit within an organization. The organization may have persona, marketing, manufacturing and accounting. Generally it is having 25 to 100 persons, which is larger than a workgroup. It is designed to support the various functions and activities of a department. They are the most common of five types of databases. For example the personal database that is designed to track data concerning employees, jobs, skills and job assignments. Once we store the data, we can query the database to obtain the answers.

Enterprise Database: Enterprise databases are intended to support an organization - wide operations and decision making. The organization may have several enterprise databases. The enterprise database provides needed information from many departments.

Figure 1.9 An enterprise data warehouse



The evolutions on enterprise databases are

1) ERP systems 2) Data warehouse implementations.

ERP Systems : Enterprise resource planning systems (ERP) : A business management system that integrates all functions of the enterprise, such as manufacturing, sales, finance, marketing, inventory, accounting, and human resources. ERP systems are software applications that provide the data necessary for the enterprise to examine and manage its activities.

Data Warehouse : An integrated decision support database whose context is derived from the various operational databases.

Internet, Intranet And Extract Databases: The business through internet has resulted an important changes in long established business models. Internet businesses improves customer information and services, which estimates traditional market channels. When the data is web enabled, the web browser interface allows users to ask unique and specific questions and receive answers based on current information. The answer to the question is automated. There is no need to go through a series of options over the telephone and wait for a human to offer assistance.

Summary of Database Applications:

TYPE	NO. OF USERS	ARCHITECTURE	SIZE OF DATA
Personal	1	PC, PDA	MB
Workgroup	5.25	CLIENT / SERVER (Two Tier)	MB/GB
Department	25-100	CLIENT / SERVER (Three Tier)	GB
Enterprise	>100	CLIENT / SERVER (Distributed parallel)	GB/ Terabytes
		WEB SERVER & APPLICATION SERVER	MB / GB

1.6 Advantages Of The Database Approach

Program-Data Independence

The separation of data description (metadata) from the application programs that use the data leads to **data independence.** Data descriptions are stored in a central location called the *repository.* Organization's data can change and evolve without changing the application programs that process the data.

Minimal Data Redundancy

Traditionally, information systems were developed using a file-processing approach. Each application had its own files, and data was not shared among applications, resulting in a great deal of **data redundancy,** or repetition of the same data value.

The database approach was developed to minimize data redundancy by creating separate files for each entity. Files are referred to as **tables,** and a **database** is a collection of related tables. Data files are integrated into a single logical structure. While not completely eliminating redundancy, the designer can control the type and amount of redundancy.

Improved Data Consistency

By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example a customer address is stored only once, we cannot disagree on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage space that results from redundant data storage.

Improved Data Sharing

Database is designed as a shared resource. Authorized users are granted permission to use the database, and provided with user views to facilitate this use.

Improved Productivity of Application Development:

Reduction of the cost and time for developing new business applications. The programmer can concentrate on the specific functions required for the new application and DBMS provides a number of high-level productivity tools such as forms and report generators and high-level languages that automate some of the activities of database design and implementation.

Enforcement of Standards

Standards include naming conventions, data quality standards, and uniform procedures for accessing, updating, and protecting data.

Improved Data Quality

Concern with poor quality of data is a common theme in database administration today. The database approach provides a number of tools and processes to improve data quality. Two of the more important are the following:

1. Database designers can specify integrity constraints that are enforced by the DBMS/ A constraint is a rule that cannot be violated by database users.
2. One of the objectives of a data warehouse environment is to clean up (or “scrub”) operational data before they are placed in the data warehouse.

Improved Data Accessibility and Responsiveness: End users without programming experience can retrieve and display data (using SQL).

Reduced Program Maintenance: Data are independent of the application programs that use them, and either one can be changed without a change in the other.

1.7 Costs and Risks of the Database Approach

As with any business decision, the database approach entails some additional costs and risks that must be recognized and managed when implementing this approach.

New, Specialized Personnel: Frequently, organizations that adopt the database approach need to hire or train individuals to design and implement databases, provide database administration services, and manage a staff of new people. Further, because of the rapid changes in technology these new people will have to be retrained or upgraded on a regular basis. This personnel increase may be more than offset by other productivity gains, but an organization should not minimize the need for these specialized skills, which are required to obtain the most from the potential benefits.

Installation and Management Cost and Complexity: A multi-user database management systems is a large and complex suite of software that has a high initial cost, requires a staff of trained personnel to install and operate, and also has substantial annual maintenance and support costs. Installing such a system may also require upgrades to the hardware and data communications systems in the organization. Substantial training is normally required on an ongoing basis to keep up with new releases and upgrades. Additional ore more sophisticated and costly database software may be needed to provide security and to ensure proper concurrent updating of shared data.

Conversion Costs: The term legacy systems are widely used to refer to older applications in an organization that are based on file processing and/or older database technology. The cost of converting these older systems to modern database technology—measured in terms of dollars, time, and organizational commitment—may often seem prohibitive to an organization.

Need for Explicit Backup and Recovery: **A shared corporate database must be accurate and available at all times. This requires that comprehensive procedures be developed and used for providing backup copies of data and for restoring a database when damage occurs. A modern database management system normally automates many more of the backup and recovery tasks than a file system.**

Organizational Conflict : A shared database requires a consensus on data definitions and ownership as well as responsibilities for accurate and maintenance. Experience has shown that conflicts on data definitions, data formats and coding, rights to update shared data, and associated issues are frequent and often difficult to resolve. Handling these issues requires organizational commitment to the database approach, organizationally astute database administrators, and a sound evolutionary approach to database development.

If strong top management support of and commitment to the database approach is lacking, end-user development of stand-alone databases is likely to proliferate. These databases do not follow the general database approach that we have described, and they are unlikely to provide the benefits described earlier.

1.8 Components of Database Environment

Computer-Aided Software Engineering tools(CASE): Automated tools used to design databases and application programs.

Repository : A generalized knowledge base for all data definitions, data relationships, screen and report formats and other system components. A repository contains an extended set of metadata important for managing databases as well as other components of system.

Database Management System: It is a software application that is used to create , maintain and provide controlled access to user databases.

Database: It is an organized collection of logically related data. The difference between the Repository and Database is, Repository contains definitions of data, where as database contains occurrences of data.

Applications Programs: Computer programs that are used to create and maintain the database and provide the information to the user.

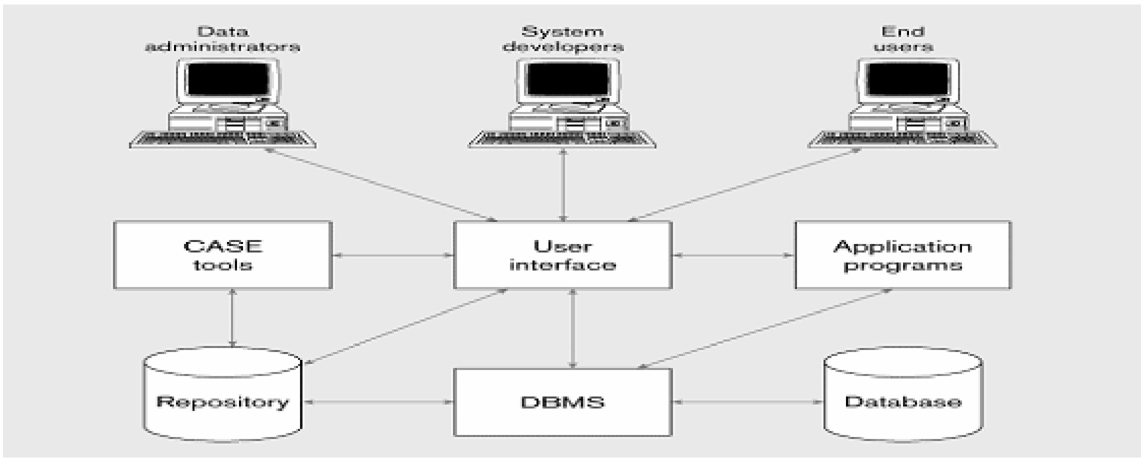
User Interface: Languages, Menus and other facilities by which user interact with various system components, such as CASE tools, application programs, the DBMS and the repository.

Data Administrators: Persons who are responsible for the overall information resources of an organization. Data administrators uses the CASE tools to improve the productivity of the database design and the Repository.

System Developers: System analysts and programmers design new application programs. These system developers use CASE tools for system requirements analysis and program design.

End Users: The persons, who add, delete and modify data in the database. All user interactions with the database must be routed through the DBMS.

Figure 1.10 Components of the database environment



1.9 Summary

Databases are used to store, manipulate, and retrieve data in every type organization. A course of modern database management is one of the most important courses in the information systems curriculum. Metadata are data that describe the properties or characteristics of other data. A database management system is a general-purpose commercial software system used to define, create, maintain, and provide controlled access to the database.

Computer file processing systems were developed early in the computer era so that computers could store, manipulate large and retrieve the large files of data. Database applications can be described in the following categories, personal databases, workgroup databases, departmental databases, enterprise databases and Internet databases.

1.10 Technical Terms

Database: An organized collection of logically related data.

Data: Facts, text, graphics, images, sound, and video segments that have meaning in the users environment**.**

Information: Data that have been processed in such a way as to increase the knowledge of the person who uses the data.

Database Management System: A Database Management System (DBMS) is a software package to facilitate the creation and maintenance of a computerized database. A Database System (DB) is a DBMS together with the data itself.

Query: A specific set of instructions for extracting particular data from a database.

Metadata: Data is useful when placed in some context. **Metadata** are data that describe the properties or characteristics, such as definitions, structures, and rules or constraints, of other data.

Data Processing: Systematically performing a series of actions with data. May be done by manual, mechanical, electromechanical, or electronic (primarily computer) means.

Security: The process of protecting information from unauthorized use. An example is the use of credit card numbers on the Internet to purchase merchandise and services.

Data warehouse: An integrated decision support database whose content is derived from the various operational databases.

1.11 Questions

1. **Define the following key Terms**

   a) data b) information c) metadata d) constraint

2. Contrast the following terms :
   a) Data dependence, data independence
   b) repository; database
   c) data warehouse, data mining
3. Why has the definition of data been expanded in today's environment?
4. How are relationships between tables expressed In a relational database?
5. What does the term data independence mean and why is it an important goal?
6. List five additional costs or risks associated with the database approach.
7. List the major components in a database system environment.
8. List potential benefits of the database approach.

1.12 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe
(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.
Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson- 2

THE DATA DEVELOPMENT PROCESS

2.0 Objective

To be able to

- Define the terms Enterprise data modeling, information systems architecture(ISA), information engineering top-down planning, business functions, Functional decomposition, SDLC, CASE.
- Describe the life cycle of systems developments project, with emphasis on the purpose of database analysis, design, and implementation activities.
- Explain the prototyping approach to database and application development.
- Explain the prototyping approach to database and application development.
- Explain the roles if individuals who design , implement, use and administer databases.
- Explain the differences between conceptual, external, and physical schemas and the reasons for a three-schema architecture for databases.
- Explain the three-tired location architecture for databases and database processing.

Structure

2.1 Introduction

This lesson presents an overview of the general steps followed in the analysis, design, implementation, and administration of databases. This lesson also emphasizes the need to coordinate database development with al the other activities in the development of a complete information system.

Finally, this lesson gives you a general understanding of the key steps and types of skill.

2.2 Database Development with in Information Systems Develop ment

Generally database development begins with Enterprise data modeling. The purpose of Enterprise Data Modeling is to create an overall picture or explanation of organizational data but not for design of database. Enterprise Data Modeling is nothing but a graphical model that shows the high-level entities for the organizational database relationships among these entities.

2.2.1 Information System Architecture (ISA)

It is a conceptual blue print or plan that express desired structure for information system in an organization. An enterprise data model is only one part or blue print for an information system Architecture in an organization. The information system architecture consists of six key components.

1. **data**
2. **Process** : Manipulate data
3. **Network** : Transports data around organization, between organizations and its key business partners.
4. **People:** **W**ho perform process who are source and who receives data or information.
5. **Events and points in time:** **W**hen processes are performed.
6. **Reasons**: For events and rules that govern the processing of data.

2.2.2 Information Engineering

INFORMATION ENGINEERING: A formal top-down methodology that uses data orientation to create and maintain information systems. The data orientation is nothing but explanation of how databases are identified and defined. Information engineering includes four phases, which are

1. **Planning**
2. **Analysis**
3. **Design**
4. **Implementation**

Planning phase of Information System: The goal to align information technology with business activities of the organization. The phase includes three steps

Table 2.1 Information Engineering Planning Phase

Step	Explanation
1	Identify strategic planning factors
	a. Goals
	b. Critical success factors
	c. Problem Areas
2.	Identify corporate planning objects
	a. Organizational Units
	b. Locations
	c. Business functions
	d. Entity types
	e. Information systems
3.	Develop an Enterprise model
	a. Functional decomposition
	b. E-R diagrams
	c. Planning matrixes

2.2.3 Information Systems Planning

The goal of information systems planning is to align information technology with the business strategies of the organization. Such as alignment is important in order to achieve the maximum benefits from investments in information systems and technologies. As depicted in Table 2.1, the planning phase of the information engineering methodology includes three steps, which are in the following three sections.

Identifying Strategic Planning Factors: Strategic planning factors are organizational goals, critical success factors and problem areas. These are

Table 2.2 Examples Results of Information Engineering Planning Phase

Planning factor	Examples
Goals	Maintain 10% per year growth rate Maintain 15% return on investment Avoid employees layoffs Be a responsible corporate citizen

Critical success Factors	High-quality products On-time deliveries of finished products High productivity of employees
Problem areas	Inaccurate sales forecasts Increasing competition Stock outs of finished products

Identifying Corporate Planning Objects: The corporate planning objects define the business scope. There are five key planning objects; these are organizational units, Organizational locations, Business functions, Entity types and Information systems.

Table 2.3 Example Corporate Planning Objects

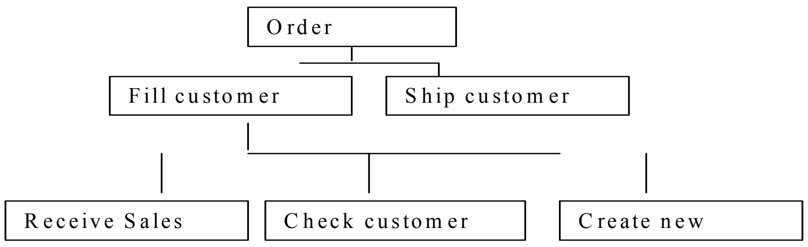
Planning object	**Description & example**
Organizational units	Various departments of the organization Ex. Sales, manufacturing, Accounting
Organizational locations	Places where business operations occur Ex. Corporate Head quarters, plant etc.,
Business functions	A related group of business processes that supports some aspects of an enterprise. Ex. Marketing & sales (Order fulfillment & shipment), Material Management (INVENTORY), Finance and Accounting etc.,
Entity types	Category of data about people, places and things. Ex. Customer, product, work center, order invoice, etc.,
Information systems	Application software for handling set of data Ex. Transaction processing system (order processing, payroll etc) Management information system (sales management, Inventory control etc)

Developing an Enterprise Model: An enterprise Model consists of a functional breakdown (decomposition) of each business function, Entity-relationship diagram and various planning matrixes.

Functional decomposition: is the process of breaking down the functions of an organization in order to simplify problems and identify components

Figure 2.2 Example process decomposition of an order fulfillment function

Ex.



E-R diagram describes entities and their relationships of an organization. It also describes business rules of various entities.

Planning Matrixes: The other relationships between various planning objects are also depicted during enterprise modeling. A common format for interrelationship between planning objects is matrixes. Widely used **planning matrixes** are

Location-to-function	Which business functions are being performed at which locations
Unit-to-function	Does which business unit perform which business functions
Information system-to-data entry	Which data are stored, used, updated or deleted with in each function
Information system-to objective	Which information systems support each business objective

2.3 Database Development Process

Information systems planning, based on information engineering, is one source of database development projects. Such projects develop new databases often to meet strategic organizational needs, such as improved customer support, better production and inventory management, or more accurate sales forecasting, Many database development projects arise, however, more in bottom-up fashion.

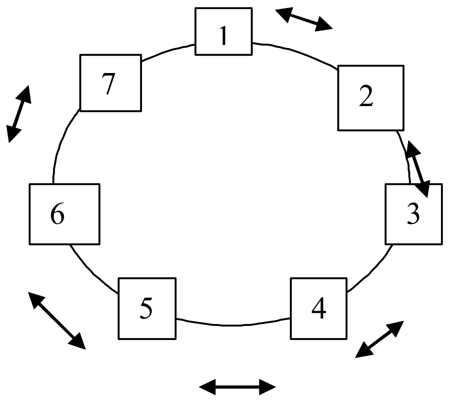
In this case, projects are requested by information system users, who need certain information to do their job, or from other information systems professional, who see a need to improve data management in the organization. Even in the bottom-up case, enterprise data modeling still must be done in order to understand whether existing databases can provide the desired data, and if not, what new databases, data entities, and attributes need to be added to the current organizational data resource.

Whether identified from strategic or operational information needs, each database development projects usually focus on one database. Some database projects concentrate only on defining, designing, and implement ting a database as a foundation for subsequent information systems development. In most cases, however, a database and the associated information processing functions are developed together as part of a comprehensive information systems development project.

2.3.1. System Development Life Cycle

A traditional process for conducting an information system development project is called System Development Life Cycle (SDLC). The SDLC is a complete set of steps that team of information system professionals, including database designers and programmers, follow in an organization to specify, develop, maintain and replace information system. This can be viewed as cascade steps, i.e. the total information is developed in different modules, output from one module is used as input in subsequent modules.

Figure 2.4 Systems Development life Cycle (SDLC)



1	**Project identification & selection**
2	**Project initiation & planning**
3	**Analysis**
4	**Logical design**
5	**Physical design**
6	**Implementation**
7	**Maintenance**

1. Project Identification And Selection: To develop a preliminary understanding of the business system that has caused the request for a new or an enhanced information system.

2. Project Initiation And Planning: To state business system and how information systems might help solve a problem or make an opportunity possible.

3. Analysis: To analyze the business system to determine requirements, to structure those requirements and to select among competing system features.

4. Logical Design: To elaborate and create a structure or syntax for all information requirements.

5. Physical Design: To develop all technology, knowledge and organizational specification.

6.Implementation: To write programs, build data files, test and install the new system, train users, and finalize documentation.

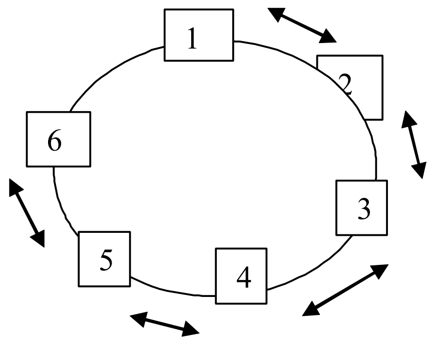
7.Maintenance: To monitor the operation and use usefulness of a system, and to modify and enhance the system.

When we compare each phase of the SDLC includes activity related to database development. So database management issues are privilege throughout the systems development process. The database development activities typically included in the phases of SDLC.

Fig. 2.5 Databases development activities during the system development life cycle(SDLC)

SDLC phase	Database activity
1. Project Identification And selection	Enterprise data modeling
2. Project initiation and Planning	Conceptual data modeling

3. Analysis	Conceptual data modeling
4. Logical design	Logical database design
5. Physical design	Physical database design
6. Implementation	Database implementation
7. Maintenance	Database maintenance



1	Enterprise modeling
2	Conceptual data modeling
3	Logical database design
4	Physical database design and definition
5	Database implementation
6	Database maintenance

Enterprise Modeling

When we compare Enterprise modeling with project identification and selection, we could set the range and general contents of organizational databases.

— Analyze current data processing

— Analyze the general business functions and their database needs

— Identify need for new data and databases in support of business

Conceptual Data Modeling

When compare conceptual data modeling with project initiation planning and analysis phases.

- Identify scope of database requirements for business data function supported by database.
- It analyzes overall data requirements for business data functions supported by database.
- If develop preliminary conceptual data model, including entities and relationships
- It compare preliminary conceptual data model with enterprise data model.

- It develop detailed conceptual data model, including all entities, relationships, attributes and business rules.
- It make conceptual data model consistent with other models of information system.
- It populate repository with all conceptual database specifications.

Logical Database Design

When we compare logical database design with logical design phase

- It analyze in detail the transactions, forms, displays, and inquiries
  (Database views) required by the business functions supported by the Database.
- It integrates database views into conceptual data model.
- It identify data integrity and security requirements, and populate repository.
- It creates a stable and well-defined structure for the database.

Physical Database Design And Definition

When we compare physical database design and definition with physical design phase,

- It define database to DBMS offenly created from repository.
- It takes decisions on physical organization of data.
- It designs database processing programs.

Database Implementation

When we compare database implementation with system implementation phase,

- It completes database documentation and training materials.
- It install database and convert data from prior systems.

Database Maintenance

When we compare database maintenance with system development maintenance phase,

- It analyze database and database applications to ensure that evolving information requirements are met .
- It tune database for improved performance.
- It fix errors in database and database applications and recover database when it is contaminated.

2.3.2 Alternative IS Development Approaches

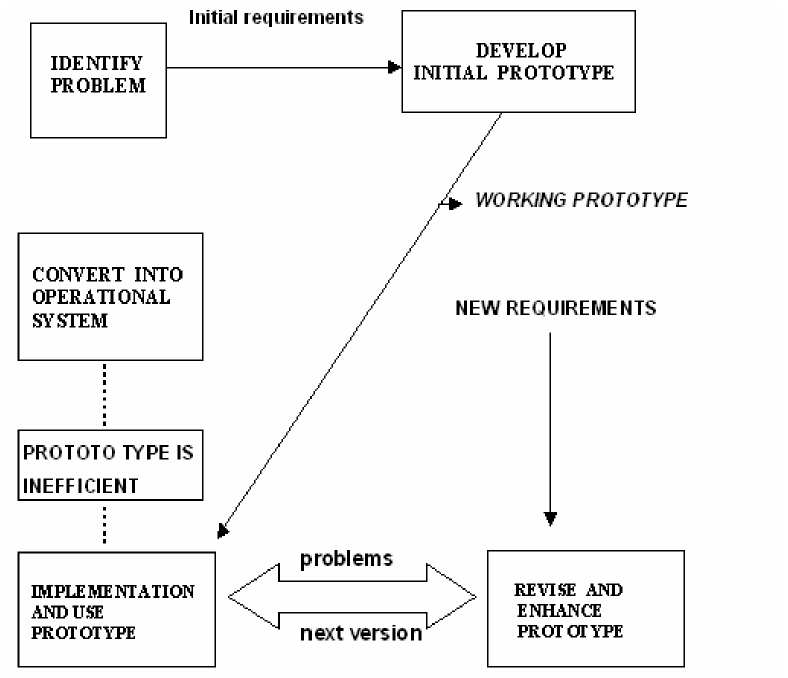
The SDLC is often criticized for the length of time needed until a working system is produced, which occurs only at the end of the process. Increasingly, organizations use more rapid application development (RAD) methods, which follow an iterative process of rapidly repeating analysis, design, and implementation steps.

Prototyping is a separate alternative development approach. Here prototype is come from protocol. Protocol is a set of rules which describe the method in which information may be transferred between two computer systems.

Prototyping: An iterative process of systems development in which requirements are converted to a working system that is continually revised through close work between Analysts and users.

The SDLC is a methodical, highly structured approach, which Includes many checks and balances to insure that each step produces accurate results. During the development of the initial prototype, one simultaneously designs the displays and reports the user wants while understanding any new database requirements and defining a database to be used by the prototype. Repeat database implementation and maintenance activities as new versions of the prototype are produced.

Figure 2.6 The prototyping methodology and database development process


IDENTIFY PROBLEM
Initial requirements
DEVELOP INITIAL PROTOTYPE
WORKING PROTOTYPE
CONVERT INTO OPERATIONAL SYSTEM
NEW REQUIREMENTS
PROTOTO TYPE IS INEFFICIENT
IMPLEMENTATION AND USE PROTOTYPE
problems
next version
REVISE AND ENHANCE PROTOTYPE

2.3.3 The Role of CASE and a Repository

At several points in the preceding section we mentioned the role of CASE tools in information systems development.

Computer-aided Software Engineering (CASE): Software tools that provide automated support for some portion of the systems development process.

Repository: A knowledge base of information about the facts that an enterprise must be able to access and the process it must perform to be successful.

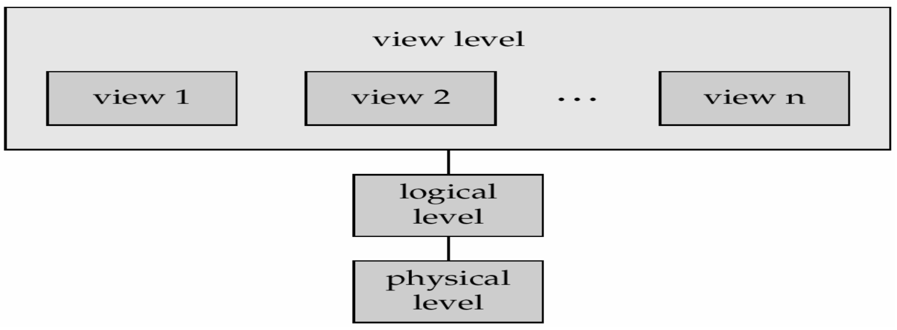
2.4 Three-Schema Architecture For Database Development

A Database is a collection of interrelated files and set of programs that allow users to access and modify these files. The main objective of three-schema architecture is to separate database from application programs.

A commonly used view of data approach is the three-level architecture suggested by ANSI/ SPARC (American National Standards Institute/Standards Planning and Requirements Committee). Under this approach, a database is considered as containing data about an *enterprise*.

The three levels of the architecture are three different views of the data. The design of each such level is considered as a schema and hence the whole design is referred as three-schema architecture. With the three schemas, program data independency can corresponding operation independency can be achieved. This process is also referred to as data abstraction.

- ☐ Internal Level or Internal Schema or Physical Level.
- ☐ Conceptual Level or Conceptual Schema or Logical Level.
- ☐ External Level or External Schema or View Level.



Three Levels Of Data Abstraction

The three level database architecture allows a clear separation of the information meaning (conceptual view) from the external data representation and from the physical data structure layout.

External Schema

The highest level of data abstraction is nothing but external schema. It describes only a part of the entire database. It is basically presentation level. Several users of the database have their own view of data. Each has a separate design of approach. In general, the end users and even the application programmers are only interested in a subset of the database. For example, a department head may only be interested in the departmental finances and student enrolments but not the library information. The librarian would not be expected to have any interest in the information about academic staff. The payroll office would have no interest in student enrolments.

Logical or Conceptual Schema

The conceptual view is the information model of the enterprise and contains the view of the whole enterprise without any concern for the physical implementation. In this level, we describe what data are to be stored and what relationships exist among the data. The database administrators use this level of abstraction.

Internal Schema

The lowest level of data abstraction describes how the data is actually stored. In this level, complex data structures of low level are described in detail. In other words, the internal view is the view about the actual physical storage of data. It tells us what data is stored in the database and how.

2.5 Three-Tiered Database Location Architecture

The data for given information system may reside in multiple locations or tiers of computers, in order to balance various organizational and technical factors. Different types of processing of data from a database may occur at different locations in order to take advantage of the processing speed, ease of programming on different computer platforms. The most commonly considered tiers are

1.Client Tier: A computer, which concentrates on managing the user-system interface and localized data. This tier also called presentation tier.

2.Application/web Tier: Processes HTTP protocol, scripting tasks, perform calculations and provides access to data. This tier also called process services tier. Most application server vendors including SUN, IBM and ORACLE have adopted JAVA 2 ENTERPRISE EDITION (J2EE) as their standard for application servers.

3.Enterprise Tier: This Tier performs sophisticated calculations and manages the merging of data from multiple sources across the organization. This tier also called data services tier. A limited view of the client/server architecture considers only the client Tier and a general server tier.

2.6 Summary

Database development begins with enterprise data modeling, where the range and general contents of organizational databases are established. Enterprise data modeling is part of an overall process that develops information systems architecture for an organization. Information systems planning must consider organization goals, critical success factors, and problem areas of the organization. When developing the new system, the project team follows a systems development process, such as systems development life cycle or prototyping. SDLC having overlapping phases, and feedback may occur that causes a project to return to a prior phase. In prototyping, a database and its applications are iteratively refined through a close interaction of systems developers and users. In system development process, CASE tools are used to develop new data models.

A modern database and the applications that use it may be located on multiple computers, for that we have client/server architecture for data processing.

2.7 Technical Terms

Top-down Planning: A generic information systems planning methodology that attempts to gain a broad understanding of the information system needs of the entire organization.

Business function: A related group of business processes that support some aspect of the mission of an enterprise.

Functional decomposition: An iterative process of breaking down the description of a system into finer and finer detail in which one function is described in greater detail by a set of other, supporting functions.

System Development Life Cycle (SDLC): The traditional methodology used to develop, maintain, and replace information systems.

Conceptual schema: A detailed technology in development specification of the overall structure of a database.

Physical Schema: Specifications for how data from a conceptual schema are stored in a computer's secondary memory.

Client/server architecture: A local area network-based environment in which database software on a server performs database commands sent to it from client workstations and application programs on each client concentrate on user interface functions.

2.8 Questions

1. Define ISA,SDLC ?
2. List and explain the four steps of information engineering.
3. List and defines the five key corporate planning objects.
4. Explain the use of information system planning matrixes in information systems development.
5. In which of the seven phases of the SDLC do database development activities occur?
6. Define a three-tired database architecture.

2.9 References

Modern Database System Concepts :
Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:
Ramez Elmasri and Shamkant B.Navathe
(Person Education Asia) 2002.

Database System Concepts:
Abraham Silberschatz.Henry F.Korth and S. Sudarshan.
Tata McGraw Hill 2002

Database Application Design and Development
Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:
Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:
Raghurama Krishna and Johannes Gherkin

- **Y.SURESH BABU.**, M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College,
GUNTUR.

Lesson-3

MODELING DATA IN THE ORGANIZATION

3.0 Objective

To be able to

- State reasons why many system developers believe that data modeling is the most important part of the system development process
- Write good names and definitions for entities, relationships, and attributes.
- Distinguish unary, binary, and ternary relationships and give a common example of each.
- Model each of the following constructs in an E-R diagram: composite attribute, multi-valued attribute, derived attribute, associative entity, identify relationship, and minimum and maximum cardinality constraints.
- Draw an E-R Diagram to represent common business situation.
- Convert a many-to-many relationship to an associative entity type.
- Model simple time-dependent data using time stamps in an E-R diagram.

Structure

3.1 Introduction

In this lesson we formalize data modeling based on the powerful concept of business rules and we describe the entity-relationship (E-R) data model. Business rules are derived from policies, procedures events, functions, and other business objects, and state constraints on the organization. Explained the guidelines for the clear naming and definition of data objects in a business. In terms of conceptual data modeling, names and definitions must be provided for entity types, attributes, and relationships.

The E-R model is used to construct a conceptual data model, which is representation of the structure and constraints of a database that is independent of the software and its associated data model that will be used to implement the database. We introduced three important concepts associated with relationships: the degree of a relationship, cardinality of relationship, and participation constraints in a relationship.

3.2 Modeling Rules of the Organization

Business rules are important in data modeling. Business rules are derived from policies, procedures, events, functions, and business objects of the organization. Business rules and policies are for creating, updating and removing data in an information processing and storage system.

- Business rules are not universal. Different Universities may have different policies for student advising and may include different types of people as students.

Ex: 1. Every student in the university must have a faculty adviser.

2. A student is any person who has applied for admission.

3.2.1 Overview of Business Rules

A business rule is a statement that defines some aspect of the business. It is intended to control or influence the behavior of the business.

A student may register for a course only if he/she has successfully completed the

prerequisites for that course.

3.2.2 Scope of Business Rules

Most organizations have rules and/or policies that may not impact organization's database.

E.g.,

1.RULE "Friday is the business casual dress day" But it has no impact on the database.

2.RULE: "A student may register for a sections of courses only if he has completed the prerequisites for that course"

This may be processed against the database.

The scope of the business Rules are divided into two, these are

1.Good Business Rules

2.Gathering Business Rules

Good Business Rules

Rules may be stated in natural language, a structured data model or other information systems documents. A business Rule will have certain characteristics.

1.Declarative — what, not how

2.Precise – clear, agreed-upon meaning

3.Atomic – one statement

4.Consistent – internally and externally

5.Expressible – structured, natural language

6.Distinct – non-redundant

7.Business-oriented – understood by business people

Declarative: A Business rule is a statement of policy is enforced or conducted; the rule does not describe a process or implementation, but rather describes what a process validates.

Precise: With the related organization, the rule must have only one interpretation among all interested people, and its meaning must be clear.

Atomic: A business rule marks one statement, not several; no part of the rule can stand on its own as a rule.

Consistent: A business rule must be internally consistent and must be consistent with other rules.

Expressible: A business rule must be able to stated in natural language.

Distinct: Business rules are not redundant, but a business rule may refer to other rules.

Business-Oriented: A business rule is stated in terms business people can understand.

Gathering Business Rules

Business functions appear in descriptions of business functions, events, policies, units, stakeholders and other objects.

These descriptions found in interviews, by asking questions about the who, what, when, where, why and how of the organization.

3.2.3 Data Names and Definitions

Data objects must be named and defined before they can be used in model of organizational data. In E-R notation you have to give clear and distinct names and definitions for entities, relationships and attributes.

Data Names:

Naming entities, relationships, and attributes must follow general guidelines.

- *Relate to Business*, not technical characteristics
- *Be meaningful* is being self-documenting avoid using words such as “has”, “is”,”it”,”and”.”or” etc.
- *Be unique* for every distinct data object.
- *Readable* is the name is structural naturally (ex empname, but not name employee)
- *Composed of words taken from an approved list* is choose vocabulary (ex maximum, but not high limit) alias names and abbreviations (CUST for customer)
- *Repeatable*, meaning that different people or the same person at different times should develop exactly or almost the same name (stdob for student and empdob for employee)

Data definitions: A definition is an explanation of a term or fact. A term is a word that has a specific meaning for the business. These terms must be defined carefully and concisely. A fact is an association between two or more terms.

3.3 The E-R Model

An entity-relation model (or E-R model) is a detailed, logical representation of the data for an organization or for a business area. The E-R model is Expressed in terms of entities in the business environment, the relationships among those entities , and the attributes of both the entities and their relationships. An E-R model is normally expressed as an entity-relations ship diagram, which is a graphical representation of an E-R model.

3.3.1Sample E-R Diagram

To jump-start your understanding of E-R diagrams, Figure 3.2 presents a simplified E-R diagram for a small furniture manufacturing company, Pine Valley Furniture. This company can purchase items from a number of different suppliers, who then ship the items to the manufacture. The items are assembled into products that are sold to customers who order the products. Each customer order may include one or more lines corresponding to the products appearing on that order.

The diagram in Figure 3.1 shows the entities and relationships for the company. Entities are represented by the rectangle symbol, while relationships between entities are represented by the diamond connected by lines to the related entities. The entities in Figure 3.1 are

Customer A person or organization who has ordered or might order products.

Product A type of furniture made by Pine Valley Furniture, which may be ordered by customers. Note that a product is not a specific bookcase since individual bookcase do not need to be tracked.

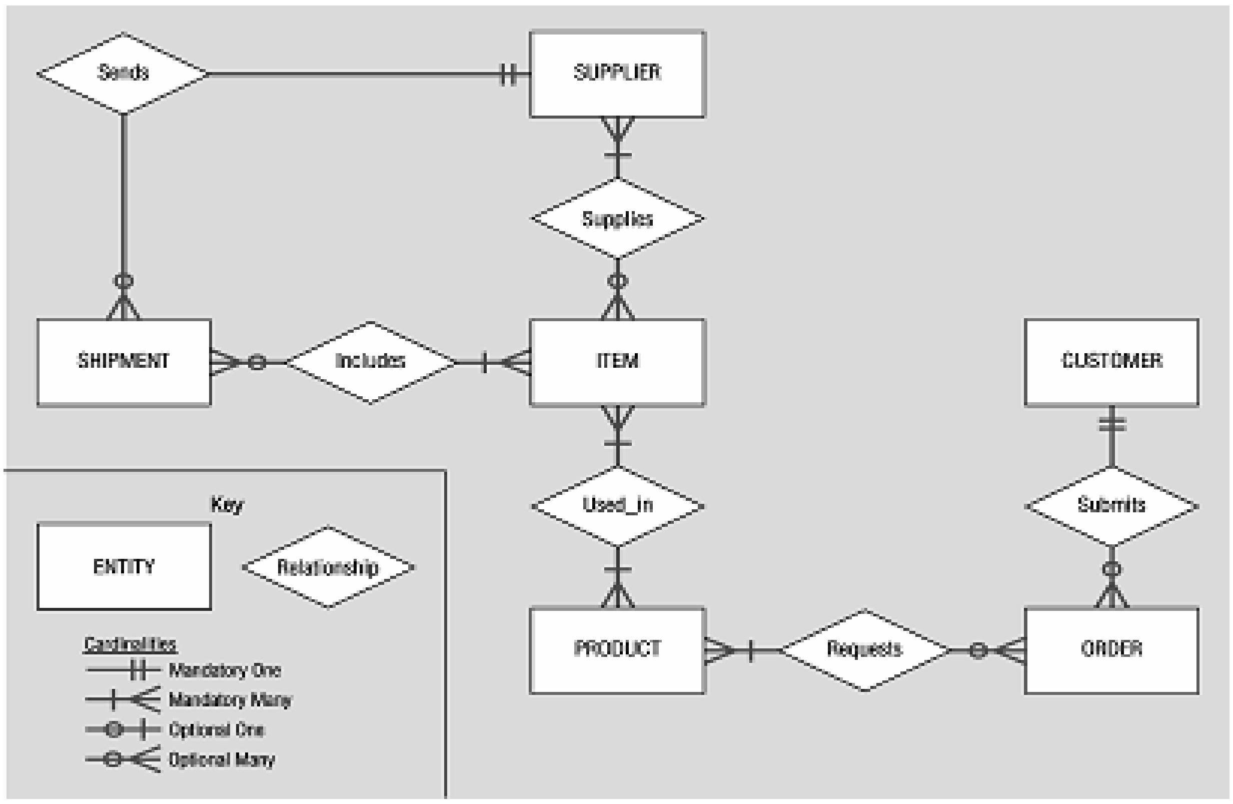
Order The transaction associated with they sale of one or more products to a customer and identified by a transaction number from sales or accounting.

Item A type of components that goes into making one or more products and can be supplied by one or more suppliers.

Supplier Another company that may provide items to Pine Valley Furniture.

Shipment The transaction associated with items received in the same package by Pine Valley Furniture from a supplier. All items in a shipment appear on one bill-of0lading document.

Figure 3.1 Sample E-R diagram :



3.3.2 E-R Model Notation

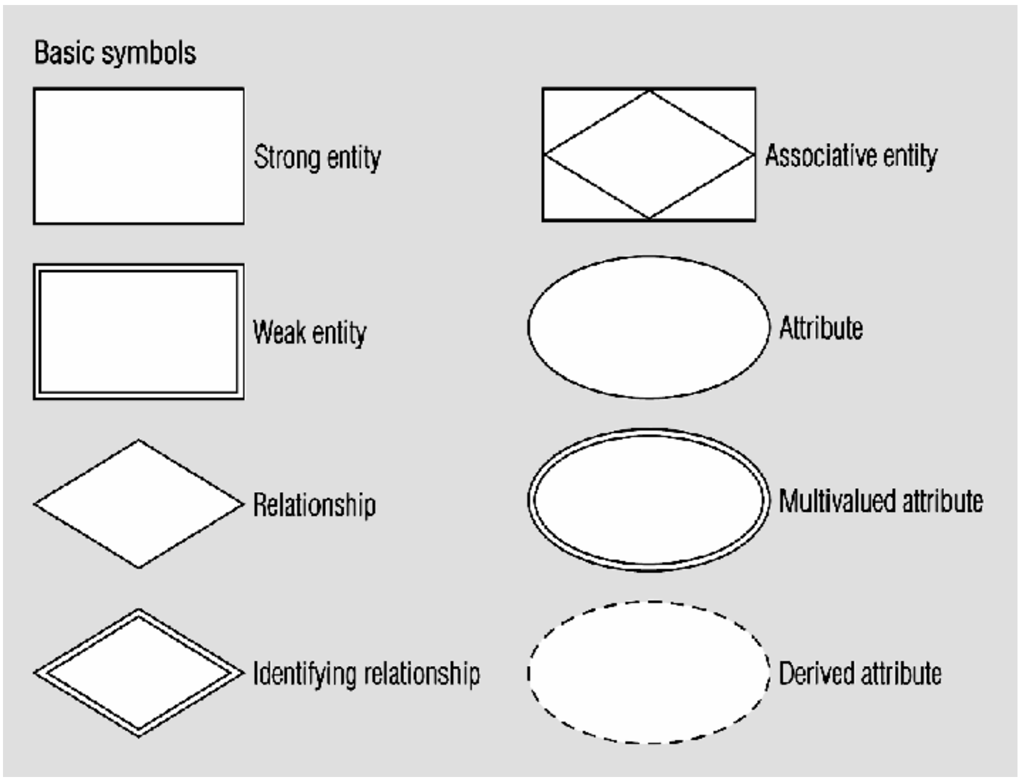
The basic notation we use for E-R diagrams is shown in figure 3.2, As indicated in the previous section, there is no industry standard notation. However , we believe that the notation in

Figure 3.2 combines most of the desirable features of the different notations that are commonly used, and also allows us to accurately model most situations that are encountered in practice. We introduce additional notation for enhanced entity-relationship models.

3.4 Entity-Relationship Model Constructs

The basic constructs of the entity-relationship are entities, relationships and attributes. As shown in Figure 3.2 the model allows numerous variations for each of these constructs. The richness of the E-R model allows designers to model real-world situations accurately and expressively, which helps account for the popularity of the model.

Figure 3.2 Basic E-R notation



3.4.1 Entities

Entity: A person, place, object event or concept in the user environment about which the organization wishes to maintain data.

Person:	EMPLOYEE, STUDENT, PATIENT
Place:	STORE, WARESHOUSE, STATE
Object:	MACHINE, BUILDING, AUTOMOBILE
Event:	SALE, REGISTRATION, RENEWAL
Concept:	ACCOUNT, COURSE, WORK CENTRE

There is an important distinction between entity types and entity instances. In an E-R diagram the entity name is placed inside the box representing the entity type. (See Figure 3.1).

Entity Type: A collection of entities that share common properties or characteristics.

Entity Instance: A single occurrence of an entity type.

Strong Entity: An entity that exists independently of other entity types.

Entity Type:	EMPLOYEE
Attributes :	
EMPLOYEE NUMBER	CHAR(10)
NAME	CHAR(25)
ADDRESS	CHAR(30)
CITY	CHAR(20)

Two instances of EMPLOYEE

111-12-6789	222-23-5678
RAMA KRISHNA	PRASAD
JKC COLLEGE	KOTHAPET
GUNTUR	GUNTUR

Example STUDENT, EMPLOYEE, COURSE

Weak Entity: An entity type whose existence depends on some other entity type.

3.4.2 Attributes

Attribute: An attribute is a property or characteristic of an entity type that is of interest to the organization.

Examples:

STUDENT: Student_ID, Student_Name, Home_Address, Phone_number

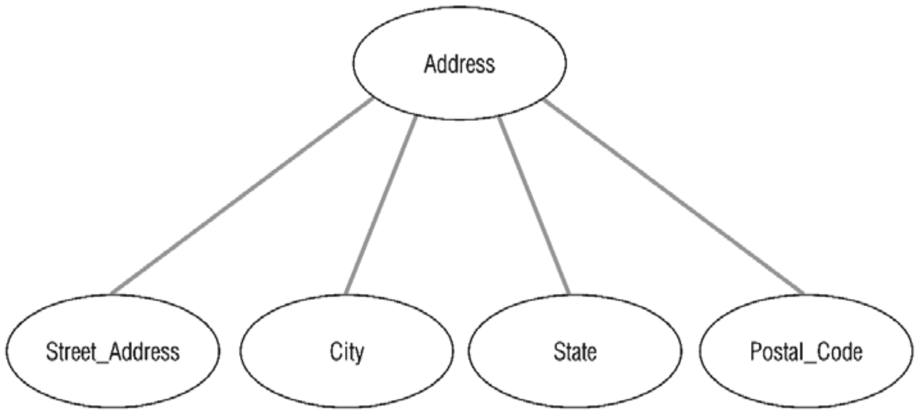
AUTOMOBILE: Vehicle_Id, Color, Weight,and Horsepower

EMPLOYEE: Employee_Id, Employee_Name, Skill

In naming an attribute, we use an initial capital letter followed by lowercase letters. If an attribute name consists of two words, we use an underscore character to connect the words and we start each word with a capital letter.

Composite Attribute: An attribute that can be broken down into component parts. The most common example is Address, which can usually be broken down into the Street_Address, City, State and Postal_Code.

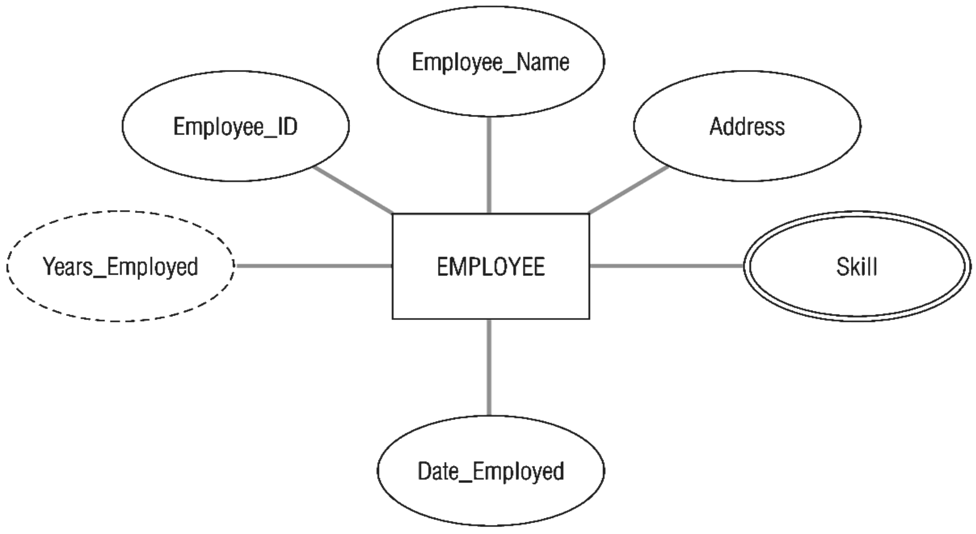
Figure 3.7 A composite attribute



Multivalued Attribute: A multivalued attribute is an attribute that may take on more than one value for a given entity instance. For example, the employee entity type in given picture has an attribute name Skill, whose values record the skill (or skills) for that employee.

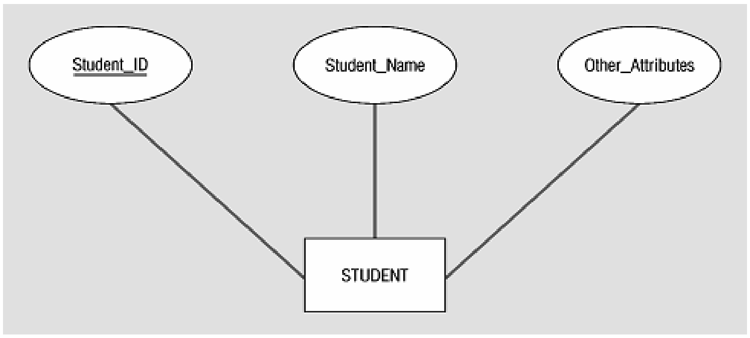
Derived Attribute: An attribute whose values can be calculated from related attribute values. For example, in an organization, the EMPLOYEE entity type has a Date_Employed attribute. If users need to know how many years a person has been employed using Date_Employed and today's date. So, Years_Employed is a derived attribute (which is shown in the below picture).

Figure 3.8 Entity with multivalued attribute and derived attribute



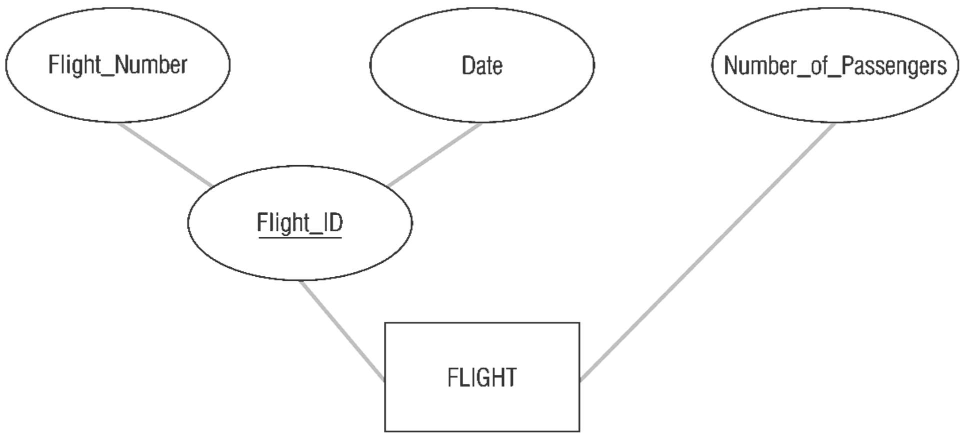
Identifier (Key Attribute): An identifier is an attribute (or combination of attributes) that uniquely identifies individual instances of an entity type. The identifier for STUDENT entity is Student_Id. An attribute such as Student_Name is not a candidate identifier, since many students may potentially have the same name. Each entity instance must have a single value for the attribute and the attribute must be associated with the entity. We underline identifier names on the E-R diagrams, which are shown in below picture.

Figure 3.9 Simple and composite key attributes (a) Simple key attribute



Composite Identifier (Composite Key Attribute): A composite identifier is an identifier that consists of a composite attribute. For example the entity FLIGHT with the composite identifier Flight_Id in turn has component attributes **Flight_Number** and **Date.** This combination is required to uniquely identify individual occurrences of FLIGHT. We use the convention that the composite attribute **(Flight_Id**) is underlined to indicate it is the identifier, while the component attributes are not underlined.

b) Composite key attribute



3.5 Relationships

A **relationship type** is a meaningful relation between or among entity types.

A relationship is an association among one or more entities.

Note: Most of our work centers on relationship types.

You can use the diamond shape to represent a relationship or you can write the relationships one the relationship line at each end:

Consider an example of a company that tracks the training courses its employees are taking. The analysts developed the following Employee-Course ERD.

3.5.1 Basic Concepts and Definitions in Relationships

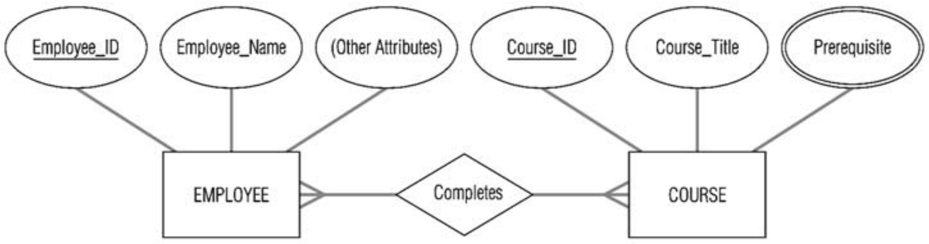
Relationship: Relationship is an association among the instances of one or more entity types.

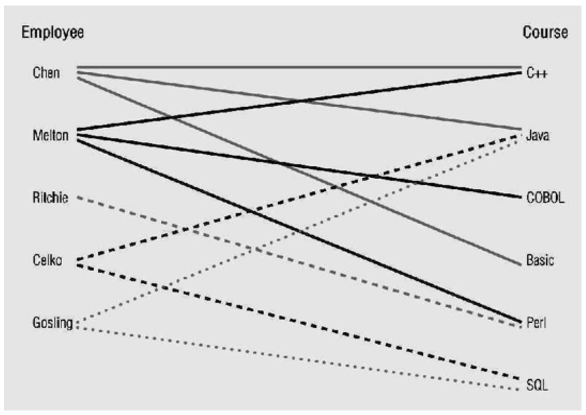
Relationship Type: It is a meaningful association between (or among) entity types. A relationship type is denoted by a diamond symbol containing the name of the relationship.

Figure 3.10 Relationship type and instances

(a) Relationships type

(b) Relationship instances

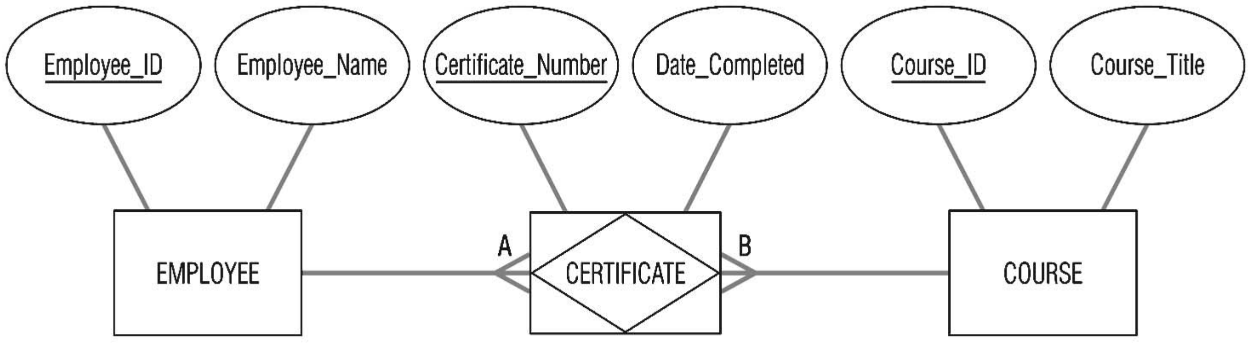




Relationship Instance: It is an association between (or among) entity instances where each relationship instance includes exactly one entity from each participating entity type.

Associative Entity: It associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.

Figure 3.11 An associative entity (a) Attribute on a relationship



The associative entity **CERTIFICATE** is represented with the diamond relationship symbol enclosed within the entity box.

Analysts use the relations symbols on the diagram while others do not. For example, the relation “Takes” makes sense when we read the ERD from left to right, as: “ an employee Takes courses”, but it doesn’t make sense when we say “Course Takes employee”. Thus, the use of relations is optional.

The above ERD represents a ‘many-to-many’ relation, stating “an employee takes many courses and a course may be attended by many employees”. In order to understand this relationship, let’s review the relationship instances, showing the occurrences of the relations between entities. The relationship instances below shows that employee named Chen takes C++, Java and Basic. Similarly, the course C++ was attended by many students: Chen and Melton.

3.5.3. Attributes or Relationship

Attribute may be associated with a many-to-many relationship, as well as with an entity. For example, suppose the organization wishes to record the date when an employee compiletes each course. This attributes is named Date_Complted.

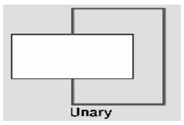
Where the attributes Date_Completed should be placed on the E-R diagram? Refferening to Figure 3.10a, you will notice that Date_Completed has not been associated with either the EMPLOYMEE or COURSE entity. That is because Dtae_ Completed is a property of the relationship Completeds, rather than a property of either entity. In other words, for each instance of the relationship Completes, there is a value for Date_Completed. One such instance shows that the employee named Melton completed the course titled C++ on 06/2000.

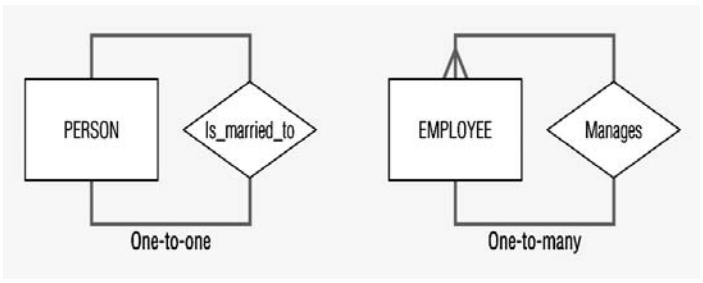
3.5.2. Degree of a Relationship

The degree of relationship is the number of entity types that participate in that relationship. The three most common relationship degrees in E-R models are

1. **Unary Relationship**
2. **Binary Relationship**
3. **Ternary Relationship**

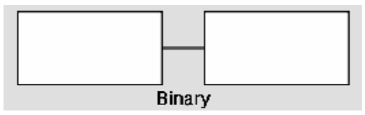
1.Unary Relationship: A relationship between the instances of a single entity type. This is also called as a recursive relationship.

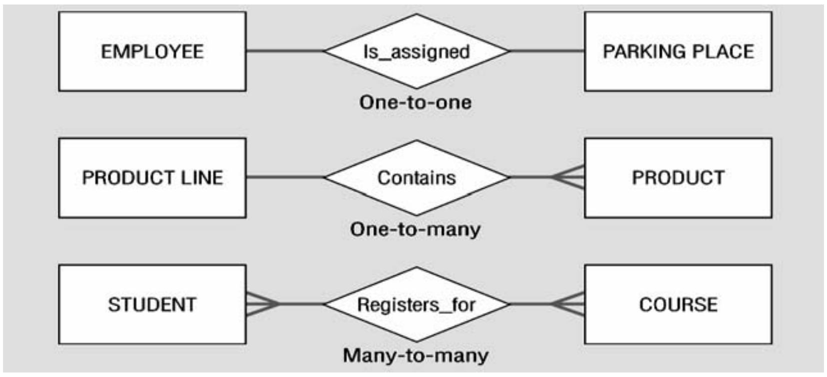




Unary relationships are also called ***recursive*** relationships. In the above example “Is_married-to” is shown a one-to-one relationship between instances of the PERSON entity type.

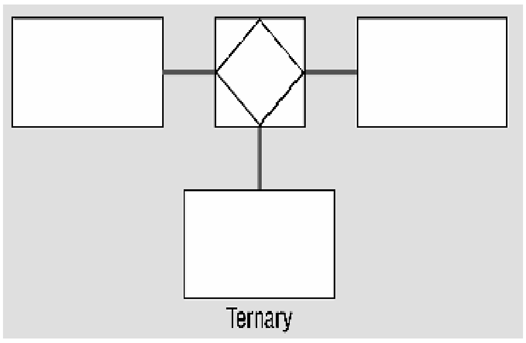
2.Binary Relationship: A binary relationship is a relationship between the instances of two entity types.

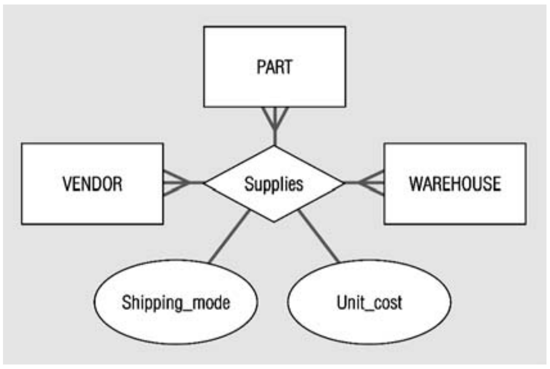




In the above example an employee is assigned one parking place, each parking place is assigned is assigned to one employee.

1. **Ternary Relationship:** A Ternary relationship is a simultaneous relationship among the instances of three entity types.





In the above example vendors can supply various parts to warehouses. The relationship supplies is used to record the specific parts that are supplied by a given vendor to a particular warehouse.

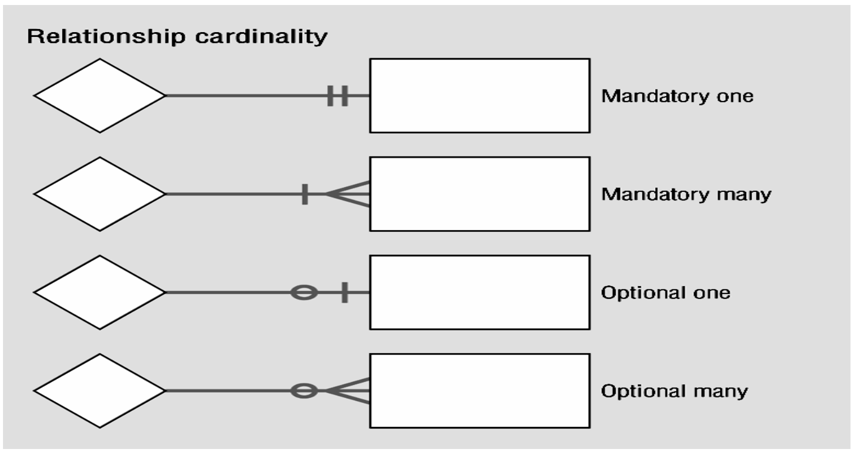
3.5.4. Cardinality Constraints

Cardinality Constraints: Cardinality Constraint Specifies the number of instances of one entity that can (or must) be associated with each instance of another entity.

Minimum Cardinality: The Minimum number of instances of one entity that may be associated with each instance of another entity.

Maximum Cardinality: The Maximum number of instances of one entity that may be associated with each instance of another entity.

As a relationship line is followed from an entity to another, near the related entity two symbols will appear. The first of those is the optionality indicator. A circle (™) indicates that the relationship is optional—the minimum number of relationships between each instance of the first entity and instances of the related entity is zero. One can think of the circle as a zero, or a letter O for "optional." A stroke (**|**) indicates that the relationship is mandatory—the minimum number of relationships between each instance of the first entity and instances of the related entity is one.



The second symbol indicates cardinality. A stroke (|) indicates that the maximum number of relationships is one. A "crows-foot" () indicates that many such relationships between instances of the related entities might exist.

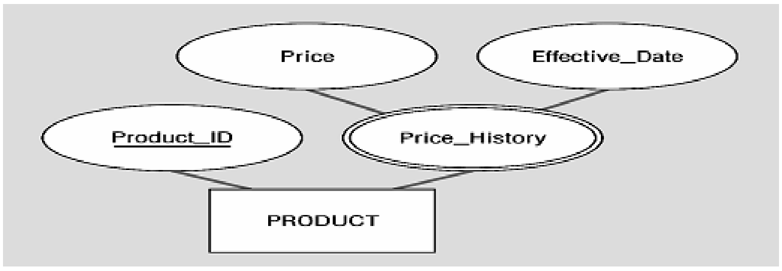
The following diagram indicates all of the possible combinations:

3.5.5 Modeling Time - Depender Data

Database contents vary over time. Fox example, in a database that contains product information, the unit rice for each product may be changed as material and labor costs and market conditions. A **time-stamp** is simply a time value that is associated over time when we need to maintain a history of those data values. Time stamps may the value becomes valid or stops being valid, or the time when critical actions were performed. The use of time stamping (given example) is often adequate for modeling time-dependent data.

Figure 3.19 , shows , we can conceptualize this requirements as a series of prices and the effective data for each price. This results in the multi-valued attribute named Price-History. The components of Price-History are Price and Effective-Date. An important characteristics of such a composite, multi-valued attribute is that the component attributes go together. Thus , in Figure , each Price is paired with the corresponding Effective-Date.

Figure 3.19 Simple example of time stamping



3.5.6 Multiple Relationships

An organization may wish to model more than one relationship between the same entity types.

The ERD demonstrates the relationship between employees and department in a company. The left relation states: "An employee supervises 1 or many employees, and an Employee is supervised by 1 employee".

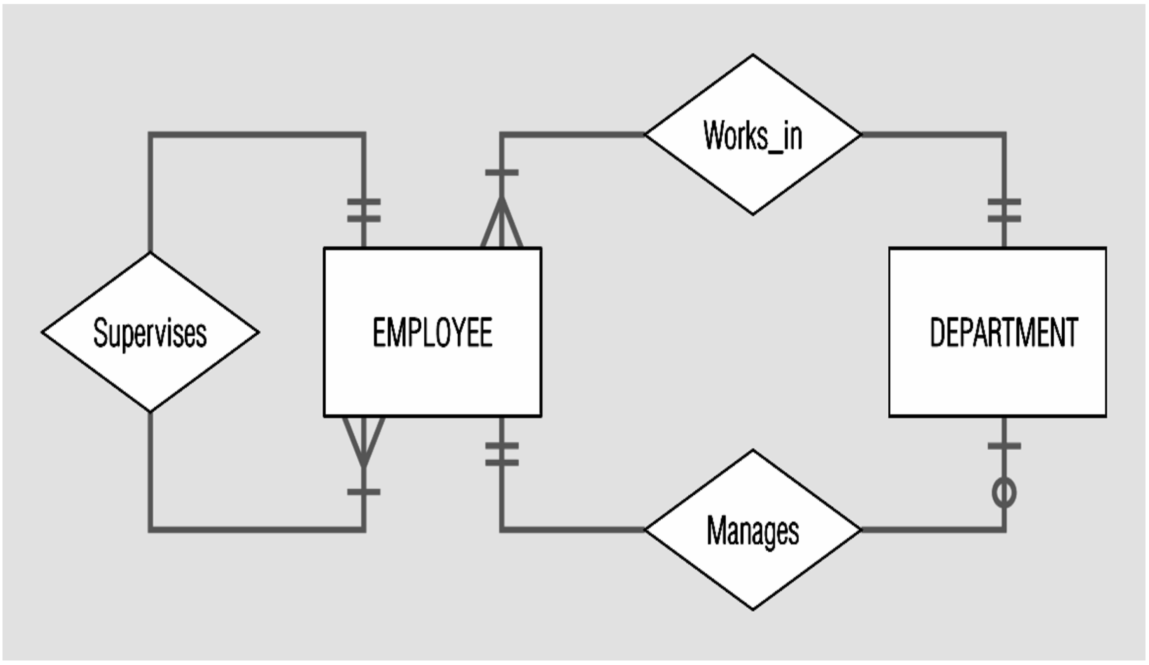


Figure 3.21 Examples of multiple relationships

The right relation states: "An employee manages zero or 1 department and the employee works in only one department". Also a department is managed by one employee and a department has one or many employees.

3.5.7 Naming and Defining Relationship

In addition to the general guidelines for naming data objects, there are a few special guidelines for naming relationship, which follows.

- A relationship name is verb phase . Relationships represented access being taken, usually in the present tense. A relationship name states the action taken, no the result of the action. The name states the essence of the interaction between the participating entity types, not the process involved.

- You should avoid vague names, such as Has of Is_related_to. Use descriptive verb phrases, often taken from the action verbs found in the definition of the relationship.

There are also some specific guidelines of defining relationships, which follows:

- A relation definition explains what actions are being taken and possibly why it is important.
- It may also be important to give examples to clarity the action.
- The definition should explain any optional participation.
- A relationship definition should also explain the reason for any explicit maximum cardinality other than many.
- A relationship between should explain any mutually exclusive relationships.
- A relationship definition should explain any restriction on participation in the relationship
- A relationship definition should explain the extent of history that is kept in the relationship.
- A relationship definition should explain whether an entity instance involved in a relationship instance can transfer participation to another relationship instance.

3.6 E-R Modeling Example: Pine Valley Furniture Company

Developing an entity-relationship diagram can proceed from one of two perspectives. With a top-down perspective, the designer proceeds from basic descriptions of the business, including its policies, process, and environment. This approach is most appropriate for developing a high-level E-R diagram with only the major entities and relationships and with a limited set of attributes. With a bottom-up approach, the designer proceeds from detailed discussions with users and from a detailed study of documents, screens, and other data sources. This approach is necessary for developing a detailed, “fully attributed” E-R diagram.

In this section we develop a high-level E-R diagram for Pine valley Furniture Company, based largely on the first of these approaches. We reference the customer invoice to add some detail and illustrate how the two approaches can be combined.

From a study of the business processes at Pine Valley Furniture Company, we have identified the following entity types. An identifier is also suggested for each entity, together with selected important attributes.

The company sells a number of different furniture products. These products are grouped into several product lines. The identifier for a product is Product-ID, while the identifier for a product line is Product-Line-ID. Referring to the customer invoice, we identify the following additional attributes for product: Product_ Description, Product- Finish, and Standard_Price. Another attribute for product line is Product_Line_Name. A product line may group any number of products, but must group at least one product. Each product must belong to exactly one product line.

Customers submit for products. The identifier for an order is Order_ID, and another attributes is Order_ID. A customer may submit any number of orders, but need not submit any orders. Each order is submitted by exactly one customer. The identifier for a customer is Customer_ID. Other attributes include Customer_ Name, Customer Address, and Postal code.

A given customer order must request at least one product and only one product per order line item. Any product sold by Pine Valley Furniture may not appear on any order line item, or may appear on one or more order line items. An attribute associated with each order line item is ordered_Quantity, which is the number of units requested.

Pine Valley Furniture has established sales territories for its customers. Each customer does business in one or more of these sales territories. The identifier for a sales territory is Territory_ID and an attribute of a Territory_Names. A sales territory may have any number of customers, or may not have any customers doing business.

Pine Valley Furniture Company has several salespersons. The identifier for a salesperson is Salesperson_ID. Other attributes include Salesperson_Name, Sales_Telephone, and Salesperson_Fax. A salesperson serves exactly one sales territory. Each sales territory is served by one or more salespersons.

3.7 Summary

This lesson has described the fundamentals of modeling data in the organization Business rules derived from policies, procedures, events, functions, and other business objects, state constraints that govern the organization and , hence, how data are handled and stored. An E-R model is usually expressed in the form of an E-R diagram, which is a graphical representation of E-R model. The basic constructs of an E-R model are entity types, relationships, and related attributes.

A relationship type is a meaningful association between entity types. A relationship instance is an association between entity instances. The most common relationship types are unary, binary and ternary. Cardinality constraint is a constraint that specifies the number of instances of entity B that may be associated with each instance of entity A.

3.8 Technical Terms

Business Rule: A statement that defines or constrains some aspect of the business . It is intended to assert business structure or to control or influence the behavior of the business.

Term: A word or phrase that has a specific meaning for the business.

Fact: An association between two or more terms.

Identifying Owner: The entity type on which the weak entity type depends.

Identifying relationship: The relationship between a weak entity type and its owner.

Time Stamp: A time value that is associated with a data value.

Simple attribute : An attribute that can not be broken down into smaller components.

Degree: The number of entity types that participate in a relationship.

3.9 Self Assessment Questions

1.Contrast the following terms stored

- attributes; derived attributes
- degree; cardinality
- strong entity type, weak entity type

2. Give four reasons why a business rules approach is advocated as a new
Paradigms for specify information systems requirements.
3. Explain where you can find business rules in an organization.
4. State three conditions that suggest the designer should model a
relationship as an associative entity type.
5. Given an examples of each of the following
Ternary relationship & Unary relationship
6. Give an example of the use effective dates as attributes of an entity.

3.10 References

Modern Database System Concepts :
Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden
Database Systems:
Ramez Elmasri and Shamkant B.Navathe
(Person Education Asia) 2002.
Database System Concepts:
Abraham Silberschatz.Henry F.Korth and S. Sudarshan.
Tata McGraw Hill 2002
Database Application Design and Development
Michael V.Manino. McGraw Hill Irwin.
Database Management Systems:
Gerald V.Post.Tata McGraw Hill 2002
Database Management Systems:
Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College.
GUNTUR.

Lesson- 4

THE ENHANCED E-R MODEL AND BUSINESS RULES-I

4.0 Objective

To be able to

- Define the following terms: enhanced entity relationship model, subtype, supertype, attribute inheritance subtype discriminator, super type, subtype hierarchy.
- Recognize when to use subtype/supertype relationships in data modeling
- Use both specialization and generalization as techniques for defining supertype/subtype relationships.
- Specify both completeness constraints and disjointness constraints in modeling supertype/ subtype relationship.
- Develop a supertype/subtytpe hierarchy for a realistic business situation.

Structure

4.1 Introduction

The researchers have continued to enhance the E-R model so that it can more accurately represent the complex data encountered in today's business environment. The term enhanced entity-relationship (EER) model is used to identify the model that has resulted from extending the original E-R model with these new modeling constructs.

The most important new modeling construct incorporated in the EER model is supertype/ subtype relationships. Adding new notation for modeling supertype/subtype relationships has greatly improved the flexibility of the basic E-R model. Enhanced E-R diagrams are sued to capture important business rules such as constraints in supertype/subtype relationships.

4.2 Representing Super types and Subtypes

The model that has resulted from extending the original E-R model with new modeling constructs .The most important new modeling construct incorporated in the EER model is

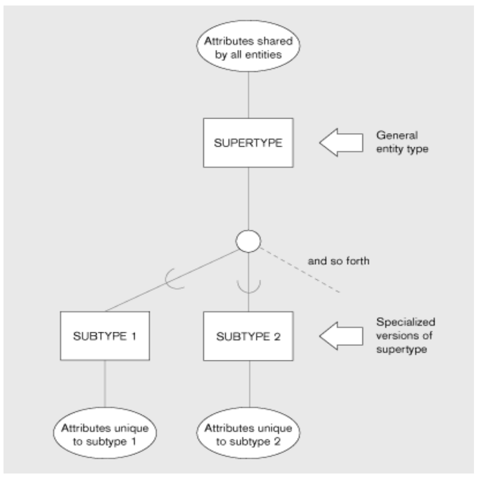
1. **SUPERTYPE**
2. **SUBTYPE**

Subtype: A subgrouping of the entities in an entity type, which has attributes that are distinct from those in other subgroupings. For example STUDENT is an entity type in a university. Two sub-types of STUDENT are GRADUATESTUDENT and UNDERGRADUATE STUDENT.

Supertype: A generic entity type that has a relationship with one or more subtypes.

4.2.1 Basic Concepts and Notation

Figure 4.1 Basic notation for SUPERTYPE/SUBTYPE relationships


Attributes shared by all entities
SUPERTYPE
General entity type
and so forth
SUBTYPE 1
SUBTYPE 2
Specialized versions of supertype
Attributes unique to subtype 1
Attributes unique to subtype 2

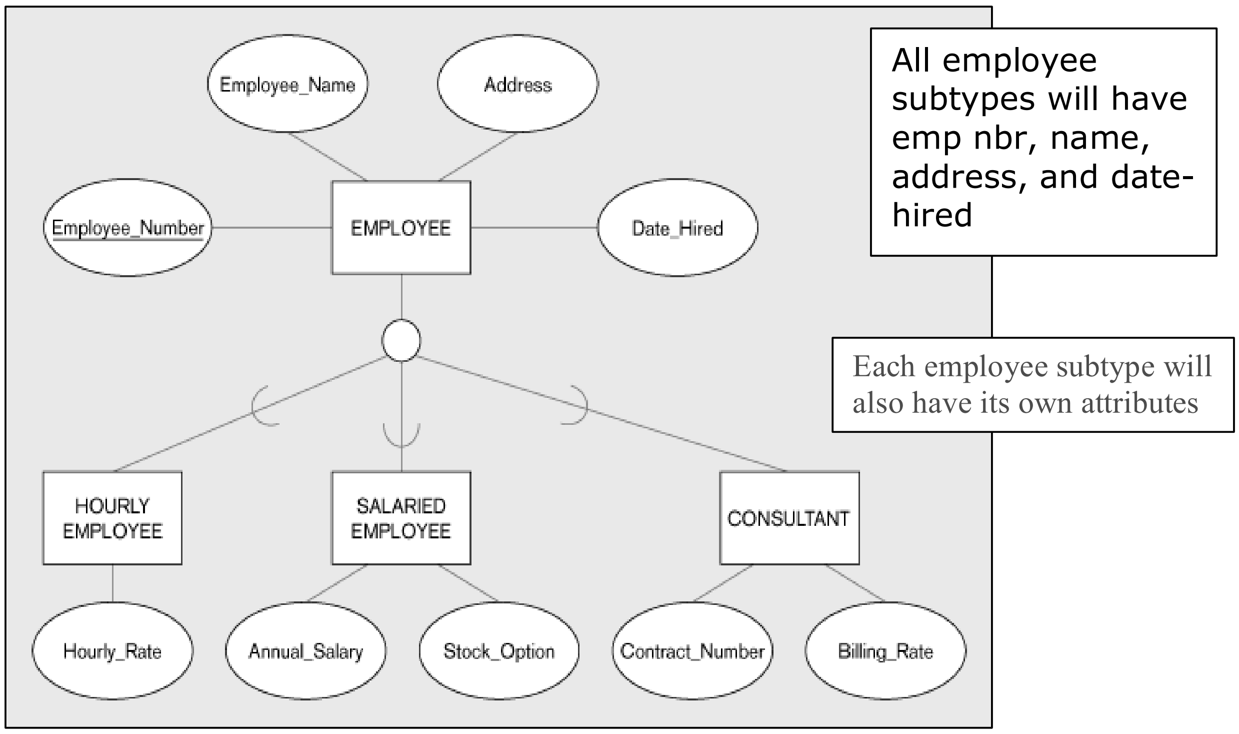
In **the** above notation the super type is connected with a line to a circle; which in turn does a line to each subtype that has been defined connect. The **U-shaped** symbol on each line connecting a subtype to the circle indicates that the subtype is a subset of the super type. It also indicates the direction of the subtype/supertype relationship.

Attributes that are shared by all entities are associated with the supertype. Attributes that are unique to a particular subtype are associated with that subtype.

E.g.: suppose that an organization has three basic types of employees. The types are Hourly employees, Salaried employees, and Contract employees. Some of the important attributes for each of these types of employees are the following:

Hourly Employees	-	employee_number, employee_name, address, date_hired, hourly_rate
Salaried Employees	-	employee_number, employee_name, address, date_hired, annual_salary, stock_option
Contract Employees	-	employee_number, employee_name, address, date_hired, contract_number, billing_rate

Notice that all of the employee types have several attributes in common. In addition each type has one or more attributes distinct from the attributes of the other types.



The above EER diagram, the EMPLOYEE supertype with its three subtypes. Attributes shared by all employees are associated with the EMPLOYEE entity type. Attributes that are peculiar to each subtype are included with that subtype only.

Attribute Inheritance: Attribute Inheritance is the property by which subtype entities inherit values of all attributes of the supertype.

Relationships and Subtypes

Relationships at the ***supertype*** level indicate that all subtypes will participate in the relationship

The instances of a ***subtype*** may participate in a relationship unique to that subtype. In this situation, the relationship is shown at the subtype level.

Figure 4.3 Supertype / subtype Relation in a Hospital

The hospital entity type PATIENT has two subtypes. OUTPATIENT and RESIDENTPATIENT (Primary key is patient_id). All patient have an admit_date and patient_name attributes. Also a responsible physician cares for every patient.

Each subtype has an attribute, which is unique to that subtype. OUTPATIENT has a checkback_date, while RESIDENT PATIENT has a date_discharged. The example illustrates the principle of relationship inheritance. OUTPATIENT and RESIDENT PATIENT are also instances of PATIENT.

4.2.2 Representing specialization and generalization

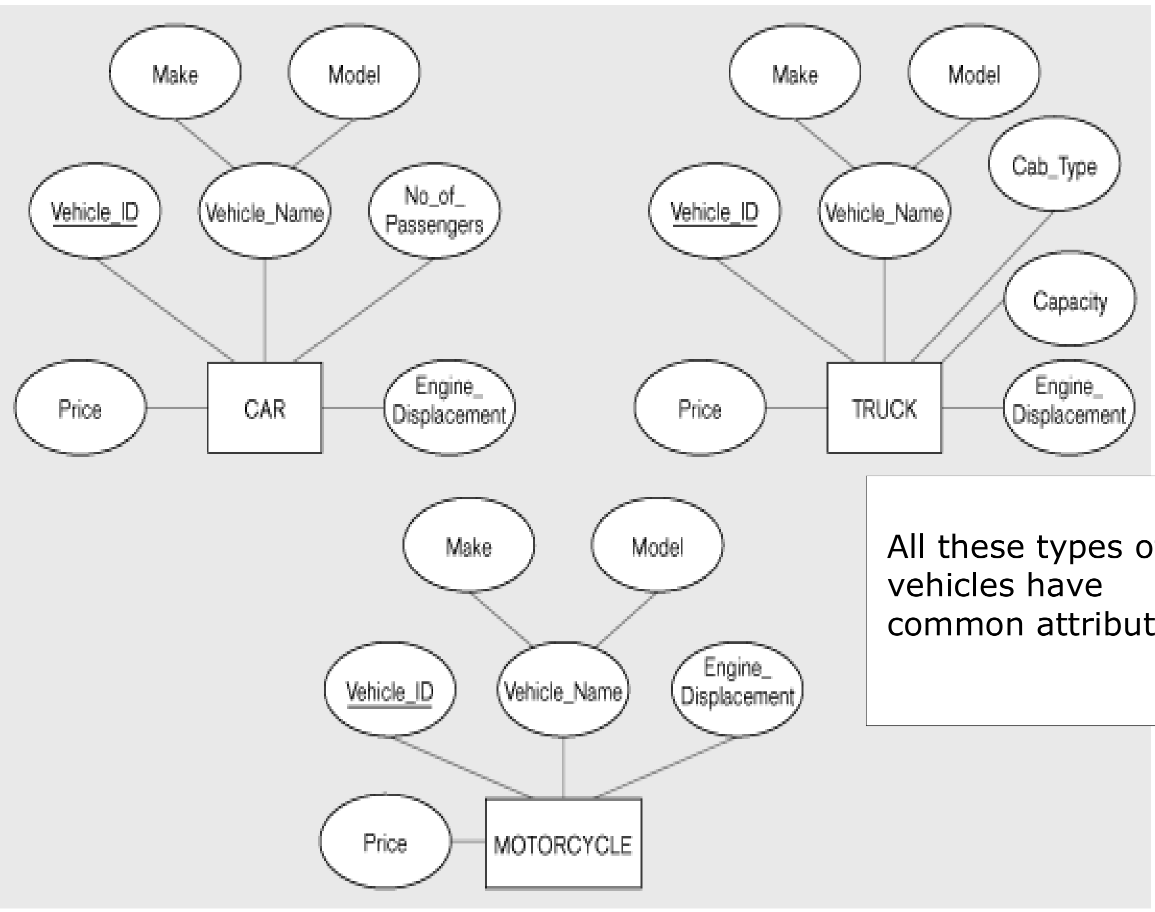
When have described and illustrated the basic principles of supertype/subtype relationships, including the characteristics of " good" subtype. But in developing real world data models, how can you recognize opportunities to exploit these relationships? There are two processes-specialization and generalization – that serve as mental models in developing supertype/subtype relationships.

Generalization

The process of defining a more general entity type from a set of more specialized entity types. Generalization is a BOTTOM-UP process.

Figure 4.4 Example of generalization

a) Three entity types: CAR, TRUCK, and MOTORCYCLE

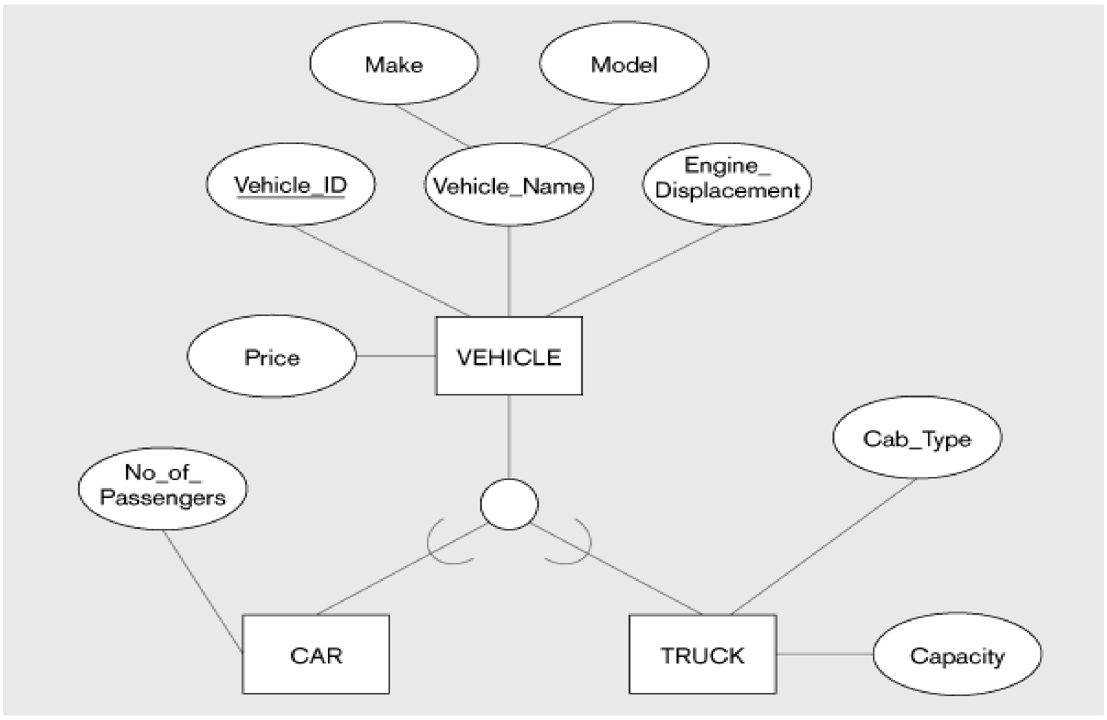


All these types of vehicles have common attributes

MOTORCYTLE.
, vehicle_name,

e represented in
Of passengers,
alization has al-
same time preserve

b) Generalization to VEHICLE supertype

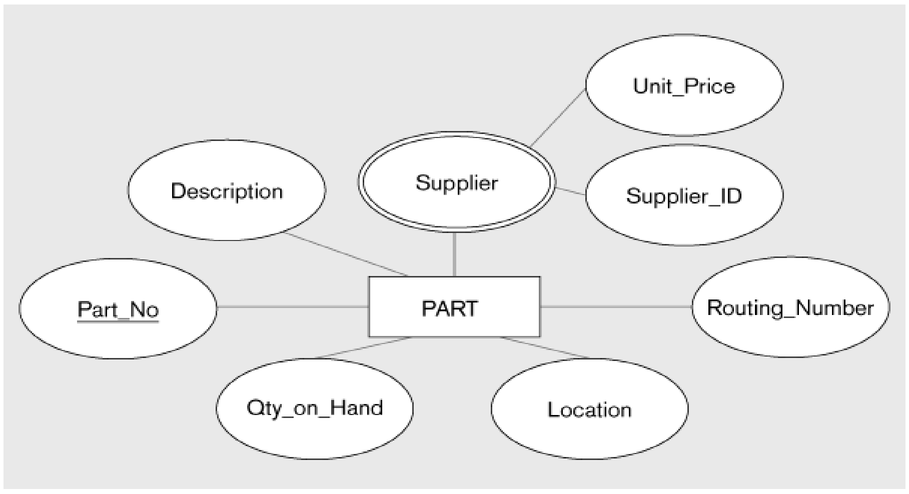


The entity type MOTORCYCLE does not have specific attributes. Thus there is no need to create a MOTORCYTLE subtype. It is also possible to have an instance of that is not a member of any of its subtypes.

Specialization

The process of defining one or more subtypes of the supertype and forming subpertype, subtype relationship. Specialization is a top-down process. Each subtype is formed based on some distinguishing characteristic such as attributes or relationships specific to the subtype.

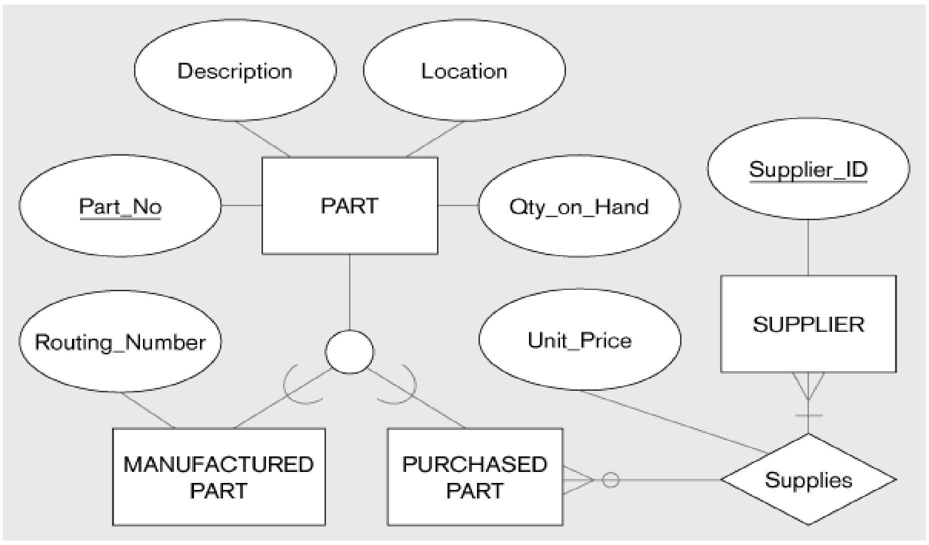
Figure 4.5 Example of specialization: a) Entity type PART



E.g.: In the above EER diagram an entity type PART has been defined with several attributes part_no, description, unit_price, location, qty_on_hand, routing_number and supplier.

The routing_number applies only to manufactured parts, while supplier-id and unit_price apply only to purchased parts. These factors specifies that PART should be specialized by defining he subtypes MANUFACTURED PART and PURCHASED PART. **Routing_no** is associated with MANUFACTURED PART.

b) Specialization to MANUFACTURED PART and PURCHASED PART



We can create a new relationship between PURCHASED PART and SUPPLIER. This relationship allows users to more easily associate purchased parts with their suppliers. The attribute unit_price is associated with the relationship supplier, so that the unit price for a part may vary from one supplier to another.

Specifying constraints in supertype / subtype relationships: The constraints allow us to capture some of the important business rules that apply to these relationships. The **two** most important types the constraints are completeness and disjointness.

Combining Specialization and Generalization

Specialization and generalization are both valuable techniques for developing supertype/ subtype relationships. Which technique you use at a particular time depends on several factors such as the nature of the problem domain, previous modeling efforts, and personal preference. You should be prepared to use both approaches and to alternate back and forth as dictated by the preceding factors.

4.3 Specifying Constraints in Supertype/Subtype Relationships

The constraints allow us to capture some of the important business rules that apply to these relationships. The two most important types the constraints are completeness and disjointness.

4.3.1 Specifying Completeness Constraints

Completeness Constraint: A completeness constraint addresses the question whether an instance of a supertype must also be a member of at least one subtype. The completeness constraint has two possible rules.

a) Total specialization

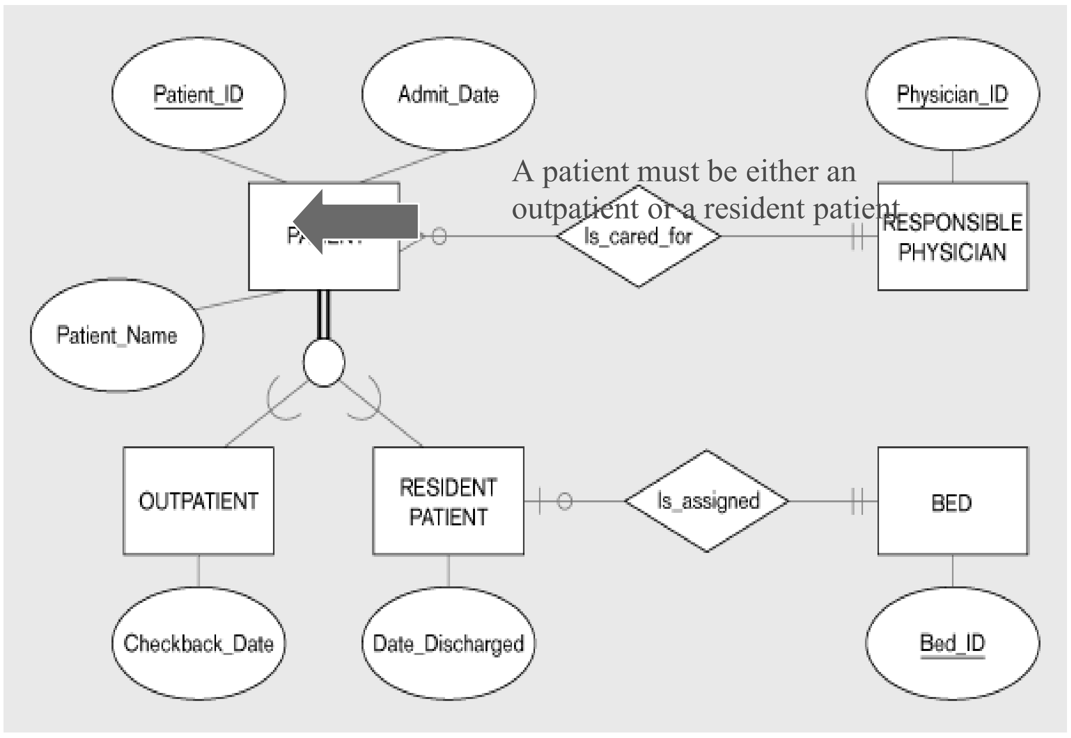
b) Partial specialization

Total Specialization Rule

The total specialization rule specifies that each entity instance of the suypertype must be member of some subtype in the relationship.

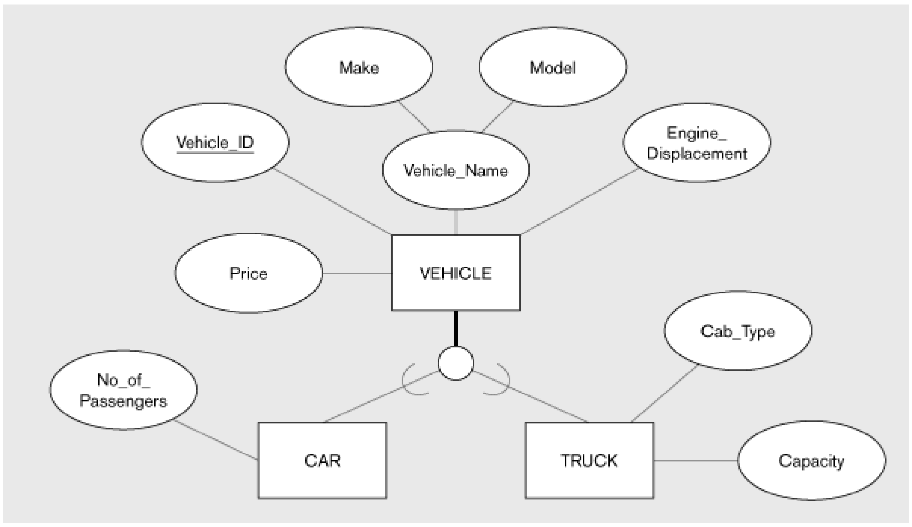
In the below example the business rule is a patient must be either an outpatient or a resident patient. The double line extending from the PATIENT entity type to the circle indicates total specialization.

Figure 4.6 Example of Completeness constraint: a) Total specialization Rule



Partial Specialization Rule: The partial specialization rule specifies that an entity instance of the supertype is allowed not belong to any subtype.

(b) Partial specialization rule



In the above example there are instances in VEHICLE supertype that are not instances of sub-type either CAR or TRUCK. If the vehicle is motorcycle, it cannot appear as an instance of any subtype. The single line from the VEHICLE supertype to the circle indicates partial specialization.

4.3.2 Specifying Disjointness Constraints

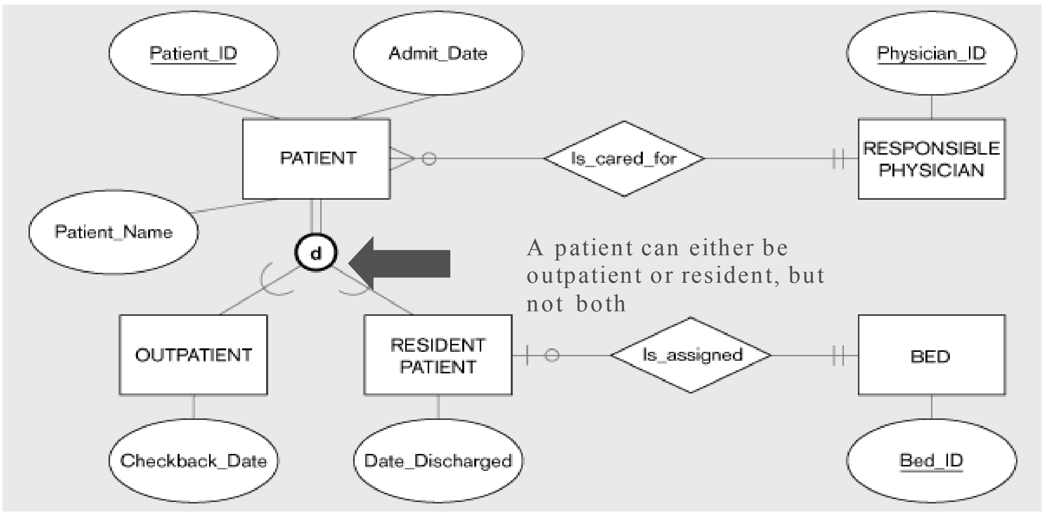
A disjointness constraint addresses the question whether an instance of a supertype may simultaneously be a member of two or more subtypes. The disjoint constraint has two possible rules. Those are

1. **Disjoint Rule**
2. **Overlap Rule**

Disjoint Rule

The disjoint rule specifies that if an entity instance is a member of one subtype, it can not simultaneously be a member of any other subtype.

Figure 4.7 Example of disjiontness constraint: a) Disjoint Rule



In the above example the business rule is, at any given time a patient must be either an outpatient or a RESIDENT patient, but cannot be both. The disjoint rule is specified by the letter d in the circle joining the supertypes.

Overlap Rule: The overlap rule specifies that an entity instance can simultaneously be a member of two or more subtypes.

Fig 4.7 b)Overlap Rule

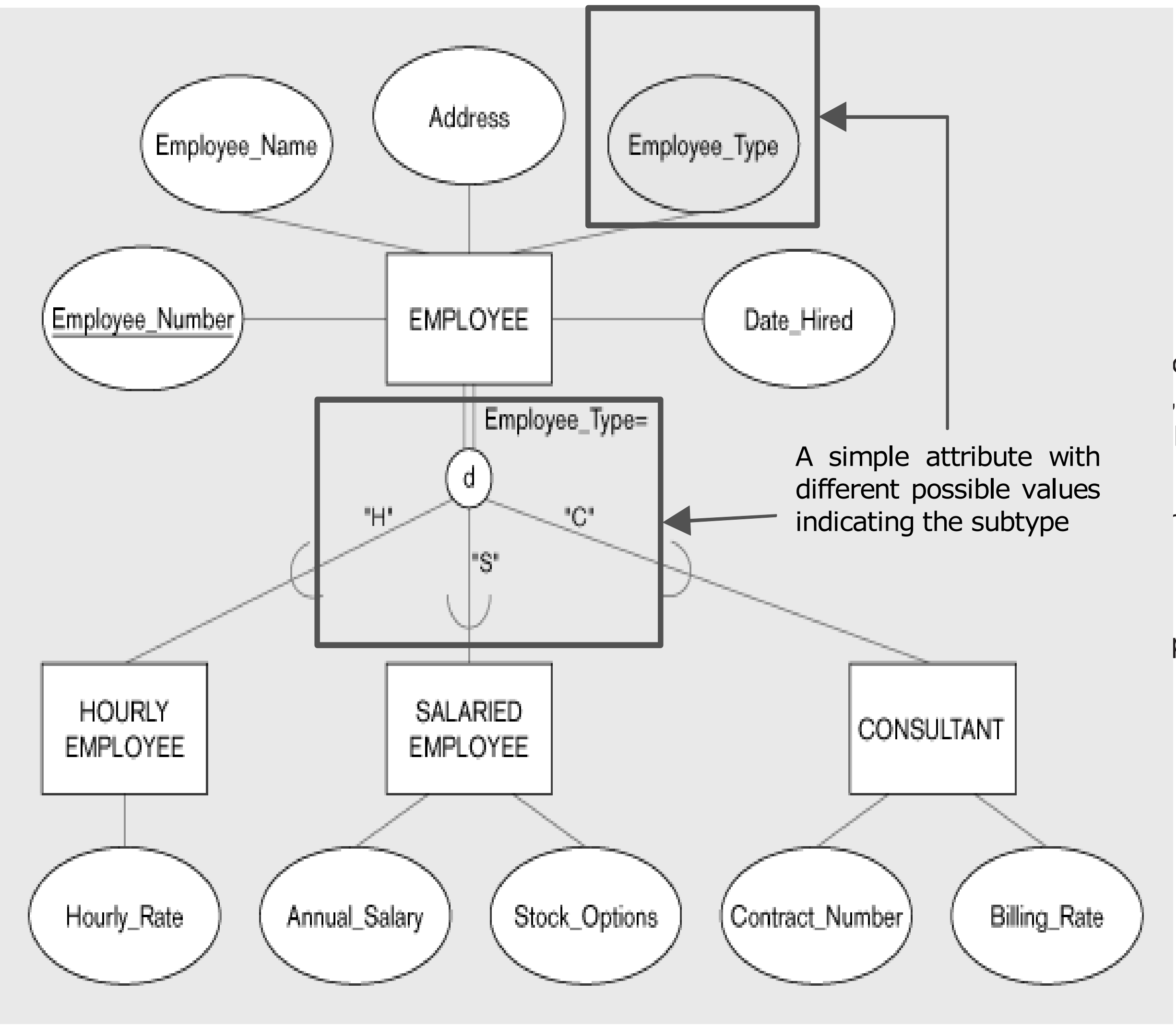
In the above example PART can be simultaneously be a member of manufactured and purchased parts. In this example PART is a particular part number (type of part) not an individual part. The overlap rule is specified by placing the letter ‘O’ in the circle joining the supertype and its subtypes.

4.3.3 Defining Subtype Discriminators

A subtype discriminator is an attribute of the supertype whose values determine the target subtype or subtype(s). The subtype discriminator has two possible rules. Those are **disjoint** and **overlapping.**

a) Disjoint: A *simple* attribute with alternative values to indicate the possible subtypes.

Figure 4.8 Introducing a subtype discriminator (*disjoint* rule)

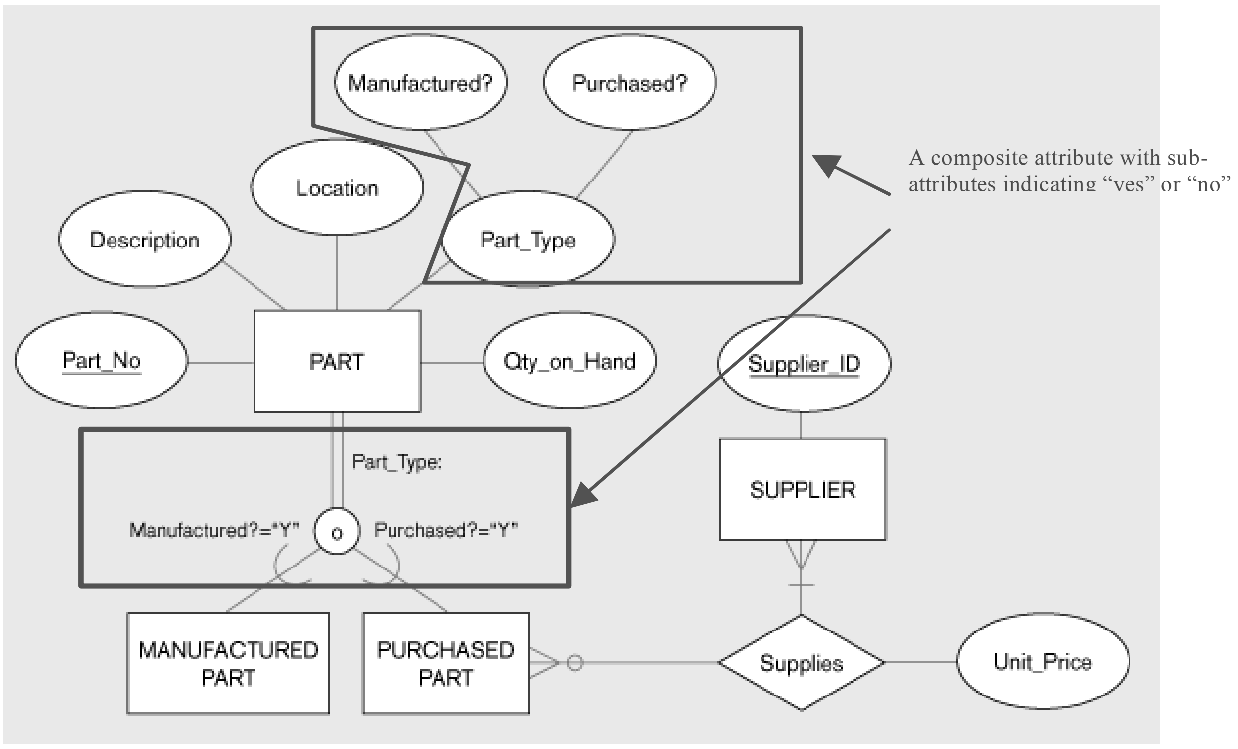


o the supertype
, this attribute is
pending on this

ne supertype to
adjacent to the

pes. Each sub-
the associated

Figure 4.9 Subtype discriminator (overlap rule)



In this example a new attribute named part_type has been added to PART. When a new instance is added to PART, these components are coded as follows,

Type of part	Manufactured?	Purchased?
Manufactured only	"y"	"n"
Purchased only	"n"	"y"
Purchased and manufactured	"y"	"y"

4.3.4 Defining Supertype/Subtype Hierarchies

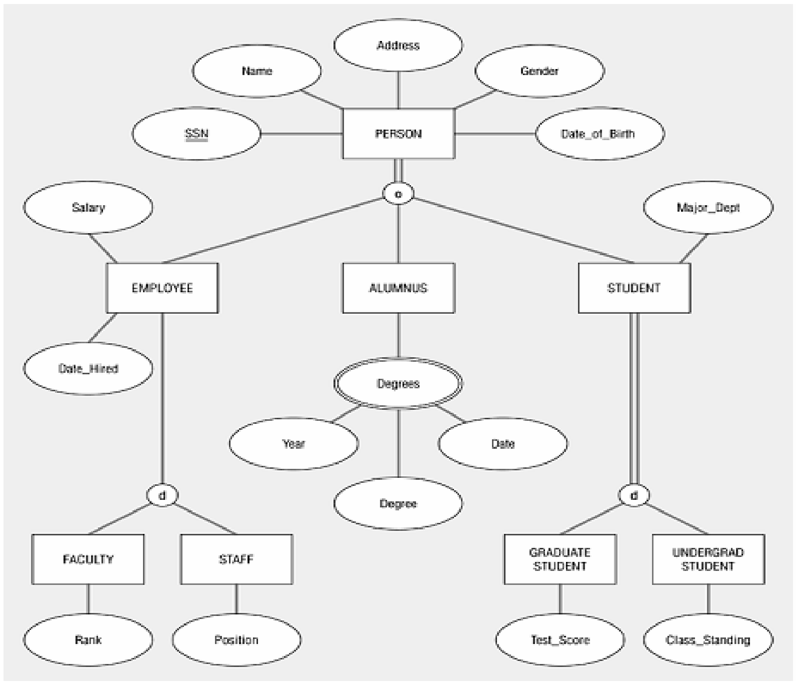
It is possible for any of the subtypes to have other subtypes defined on it. A supertype/subtype hierarchy is a hierarchical arrangement of supertypes and subtypes, where each subtype has only one supertype In modelling the Human resources in a University, the most general entity type would be PERSON (sometimes called the root) Relevant attributes are SSN (Social Security Number – Identifier), Name, Address, Gender and Date_Of_Birth. Next we define all major sub-types of the root, here there are 3, EMPLOYEE, STUDENT and ALUMNUS (already graduated).

A person may belong to more than one subtype (such as ALUMNUS and EMPLOYEE) so the overlap rule is used. Overlap allows for any overlap (3-way) so if certain combinations are not allowed a more refined supertype/subtype hierarchy would have to be developed to eliminate these.

The next step is to evaluate whether any of the subtypes already defined qualify for further specialization. Here employee has two subtypes, FACULTY and STAFF. Because there may be types of employee other than Faculty and Staff, the partial specialization rule is indicated. However, such an employee cannot be both Faculty and Staff at the same time, so the disjoint rule is indicated in the circle

- Two subtypes are defined for student: GRADUATE_STUDENT and UNDERGRADUATE STUDENT
- Notice that total specialization and the disjoint rule are specified

Figure 4.10 Example of supertype/subtype hierarchy



4.7 Summary

The basic E-R model has been extended to include supertype/subtype relationships. The techniques of generalization and specialization are important guides in developing supertype/subtype relationships. The EER notation allows us to capture the important business rules that apply to supertype/subtype relationship.

There are extensions to the E-R notation other than supertype/subtype relationships. One of the other more useful extensions is aggregation, which represents how some entities are parts of other entities. E-R diagrams can become large and complex, including hundreds are entities. Many users and managers do not need to see all the entities, relationships, and attributes to understand the part of the database with which they are most interested.

4.8 Technical Terms

Subtype Discriminator: An attribute of the supertype whose values determine the target subtype or subtype.

Supertype/subtype hierarchy: A hierarchical arrangement of supertypes and subtypes, where each subtype has only one supertype.

4.9 Self Assessment Questions

1. Define the following Terms supertype, subtype discriminator, specialization, entity cluster, disjoint rule ?
2. Contrast the following terms anchor object; corresponding object
3. State two conditions that indicate when a designer should consider using supertype/subtype relation rule.
4. State the reason for entity clustering
5. Give an example of a supertype/subtype relationship.
6. What is the purpose of a subtype discriminator?
7. Give an example of each of the following:
   a. supertype/subtype relationship where the disjoint rule applies
   b. supertype/subtype relationship where the overlap rule applies

4.10 References

Modern Database System Concepts :
Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:
Ramez Elmasri and Shamkant B.Navathe
(Person Education Asia) 2002.

Database System Concepts:
Abraham Silberschatz.Henry F.Korth and S. Sudarshan.
Tata McGraw Hill 2002

Database Application Design and Development
Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:
Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:
Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU, M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson- 5

THE ENHANCED E-R MODEL AND BUSINESS RULES-II

5.0 Objective

To be able to

- Develop an entity cluster to simplify presentation if an E-R diagram.
- Name the various categories of business rules
- Define a simple operational constraint using graphical model or structured English statement.

Structure

5.1 Introduction

The E-R, and especially EER, diagrams can become large and complex, requiring multiple pages for display. Some commercial database include hundreds of entities. Many users and managers specifying requirements for a database do not need to see all the entities, relationships, and attributes to understand the part of the database with which they are most interested. Entity clustering is a way to run a part of an entity-relationship data-model into a more macro-level view of the same data. Entity clustering is a hierarchical decomposition technique, which can make E-R diagrams easier to read. Enhanced E-R diagrams are used to capture important business rules such as constraints in supertype/subtype relationships.

5.2 EER Modeling Example: Pine Valley Furniture

After studying this below figure, you might use some questions to help you clarify the meaning of entities and relationships. Three such areas of questions are:

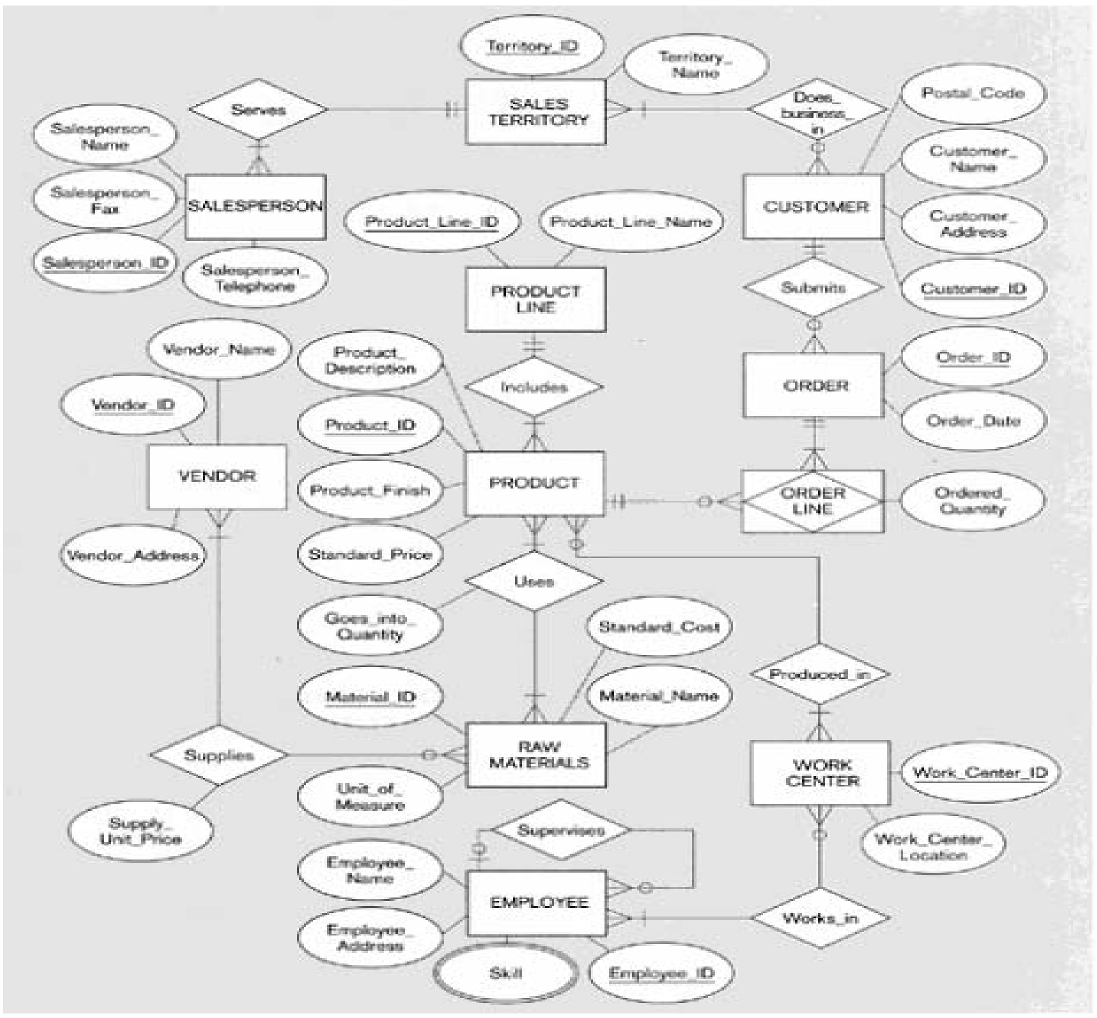
1. Why do all customers not do business in one or more sales territories?
2. Why do some employees not supervise other employees, and why are they not supervised by another employee? Why do some employee not work in a work center?
3. Why do all vendors not supply raw materials to Pine Valley Furniture?

You may have other questions, but we will concentrate on these three of illustrate how supertype/subtype relationships can be used to covey a more specific data model.

After some times investigation into the above questions, we discover the following business rules that apply to how Pine Valley Furniture does business.

1. There are two types of customers- regular and national account. Only regular customers do business in sales territories. A sales territory exists only if it has at least one regular customer associated with it. A national account customer is associated with an account manager. It is possible for a customer to be both a regular and a national customer.
2. Two special types of employee exit – management and union. Only union employee work in work centers, and a management employee supervises union employees. There are other kinds of employee besides management and union. A union employee may be promoted into management, at which time that employee stops being a union employee.

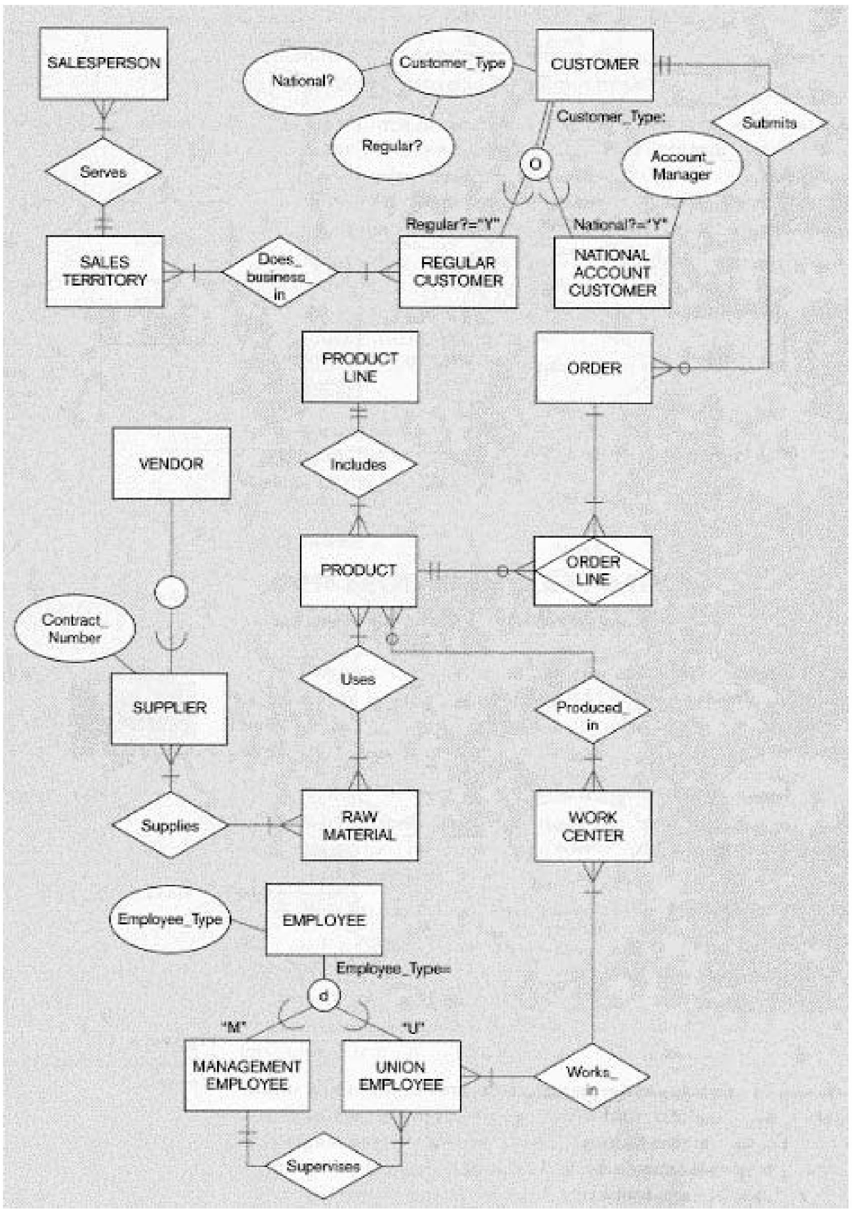
Figure 5.11 E-R diagram for Pine valley Furniture Company



3. Pine Valley Furniture keeps track of many different vendors, not all of which have ever supplied raw materials to the company. A vendor is associated with a contract number once that vendor becomes an official supplier of raw materials.

The above business rules have been used to modify the E-R diagram in Figure 5.11 into the EER diagram in Figure 5.12. Rule 1 means that there is a total, overlapping specialization of CUSTOMER into REGULAR CUSTOMER and NATIONAL ACCOUNT CUSTOMER. A composite attribute of CUSTOMER, Customer_ Type, is used to designate whether a customer instance is a regular customer, a national account, or both. Since only regular customers do business in sales territories, only regular customers are involved in the Does_ business_in relationship, and the minimum cardinality next to regular customer is one, because a sales territory, to exist, must have at least one regular customer.

Figure 5.12 EER diagram for pine Valley Furniture



Rule 2 means that there is partial, disjoint specialization of employee into MANGEMENT EMPLOYEE and UNION EMPLOYEE. An attribute of EMPLOYEE, Employee-Type, discriminates between the two special types of employees. Specialization is partial because there are other kinds of employees besides these two types. Only union employees are involved in the Works_ in relationship, but all union employees work in some work center, so the minimum cardinality of Work-in next to WORK CENTER is now mandatory. Because an employee cannot be both management and union, the specialization is disjoint.

Rule 3 means that there is a partial specialization of Vendor into SUPPLIER because only some vendors become suppliers. A supplier, not a vendor, has a contract number. Because the is only one subtype of VENDOR, there is no reason to specify a disjoint or overlap rule.

Because all suppliers supply some raw material, the minimum cardinality next to RAW MATERIAL in the Suppliers relationship now is one.

The above examples shoe how an E-R diagram can be transformed into an EER diagram once generalization/specialization of entities is understood. Not only are supertype and subtype entities now in the data model, but also additional attributes, including discriminating attributes, are added, minimum cardinalities change, and relationships move from the supertype to a subtype.

This is a good time to emphasize a point made earlier about data modeling. A data model is a conceptual picture of the data model does mot map one-for-one to elements of an implemented database.

Although the EER diagram in figure 5.12 clarifies some questions and makes the data model in figure 5.11 more explicit, it still can be difficult for some people to comprehend. Some people will jot be interested in all types of data, and some may not need to see all the details in the ER-Diagram in order to understand what the database will cover. The next section addresses ho we can simplify a complete and explicit data model for presentation to specific user groups and management.

5.3 Entity Clustering

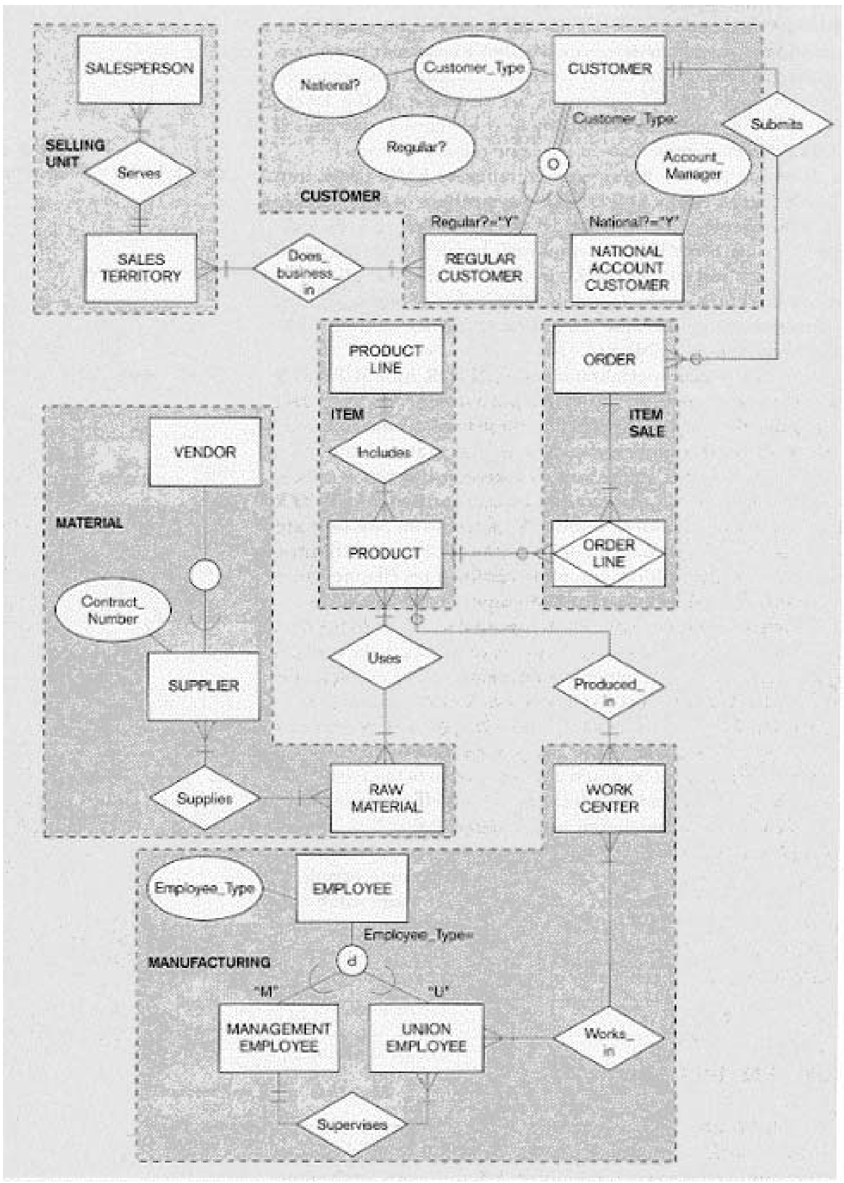
An entity cluster is a set of one or more entity types and associated relationships grouped into a single abstract entity type. Entity clustering is useful way to present a data model for a large and complex organization. Since entity cluster behave like an entity type. Entity clusters and entity types can be further grouped to form a higher-level entity cluster.

Entity-clustering is a hierarchical decomposition of a macro-level view of the data model into finer and finer views, eventually resulting in the full, detailed data model. The first of the following Figs. Shows the completed data model with dashed lines drawn around possible entity clusters, and the second shows the final result of transforming this into an EER diagram of only entity clusters and relationships.

Entity clusters are formed:

- By abstracting a supertype and its subtypes (see CUSTOMER)
- By combining directly related entity types and their relationships (SELLING_UNIT, ITEM, MATERIAL, MANUFACTURING)
- An entity cluster can also be formed by combining a strong entity and its associated weak entity types.

Figure 5.13 Entity clustering for Pine Valley Furniture a) possible entity clusters



5.4 Business Rules Revisited

We have seen that E-R diagrams are a useful means for expressing certain types of business rules. Thus, for example , the participation and disjointness constraints associated with super types and subtypes discussed . However, there are many other types of business rules in an organization that cannot be expressed with this notation.

In this section we develop a general frame work for business rules and show how to express some important types of rules that elude the standard E-R diagram notation.

5.4.1 Classification of Business Rules

A derivation is the calculation of a pay check based on work hours minus deductions.

A **structural assertion** is a statement that expresses some aspects of the static structure of the organization. The E-R diagram represents the structural assertion of data entities and attributes, and the relationships among entities

An **action assertion** is a statement of constraints or control on the action of the organization. It deals with the dynamic aspect of the organization. Examples of action assertion would include: ''A course must have a course name''; ''A student must have a GPA > 2.2 to pass the course'', or ''A professor can't teach a class unless he is qualified to do so''.

A **derivation** is a statement derived from other knowledge in the business, such as a mathematical or logical inference involving literals and facts. For example, a

5.4.2 Starting a Structural Assertion

A Structural assertion says that something of importance to the organization either exists or exists in relationship with other things of interest. A structural assertion can be as simple as the definition of a term, or a fact, which is a statement of a relationship between terms. Four examples of facts are:

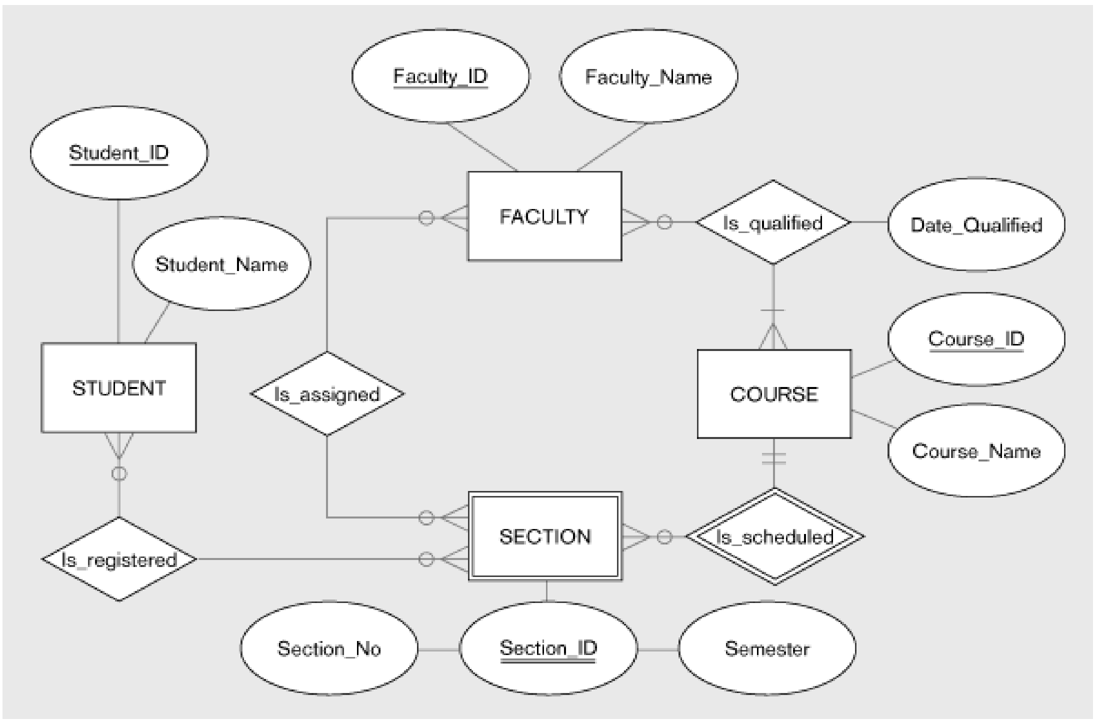
- A course is a module of instruction in a particular subject area. This *definition of the term* course associates two terms: module of instruction and subject area. We assume these are common terms that do not need to be further defined or placed in a business context.
- Student name is an *attribute* of student. His fact is shown in Figure 5-16 by the Student_Name attribute oval.
- A student may register for many sections, and a section may be registered for by many students. This fact states the participation of entity types in a *relationship*. Both student and section are business terms that require definition. This fact is shown in Figure 5-16 by the Is_registered relationship.
- A faculty is an employee of the university. Although not shown in Figure 5-16, this fact designates a *supertype / subtype* relationship between the subtype of faculty and its supertype of employee.

Derived Facts A fact that is derived from business rules using an algorithm or inference.

Student_GPA = Quality_Points/ Total_Hours_Taken where Quality_Points = sum (Credit_Hours * Numerical_Grade)

In this example, Student_GPA is derived from other base and derived facts. Quality_Points and Total_Hours are also derived facts. Student_GPA could be shown in Figure 5-16 in a dashed oval connected to STUDENT. Numerical_Grade as an attribute of Is_registered, and Credit_Hours as an attribute of COURSE.

5.16 Data model segment for class scheduling



5.4.3. Starting an Action Assertion

Whereas a structural assertion deals with the static structure of an organization, an action assertion deals with the dynamic aspects of the organization.

Anchor object: A business rule (a fact) on which actions are limited.

Action: An operation, such as create, delete, update, or read, which may be performed on data objects.

Corresponding object: A business rule (a fact) that influences the ability to perform an action on another business rule.

- A course must have a course name. In this example, the action is updating the course name property of a course.
- A student must have a value of 2.0 or greater for Student_GPA to graduate. In this example, the anchor object is a structural assertion, but it is possible for the anchor object to be another action assertion.
- A student cannot register for a section of a course for which there is no qualified faculty.

Types of Action Assertions There are three ways to classify action assertions:

1. Action assertions can be classified based on the type of result from the assertion. Looking at action assertions in this way yields three types of assertions:

   a. Condition b. Integrity Constraint c. Authorization

2. Action assertions can be classified based on the form of the assertion. Looking at action assertions in this way yields three types of assertions:

a. Enabler b. Timer c. Executive

3. Action assertions can be classified based on the rigor of the assertion. Looking at action assertions in this way yields two types of assertions:

a. Controlling b. Influencing

5.4.5 Representing and Enforcing Business Rules

Most organizations have hundreds (or thousands) of such rules. Action assertions have traditionally been implemented in procedural logic buried deep within *individual* application programs—in a form that is virtually unrecognizable, unmanageable, and inconsistent. This approach places a heavy burden on the programmer, who must know all the constraints that an action may violate and must include checks for each of these constraints. An omission, misunderstanding, or error by the programmer will likely leave the database in an invalid state.

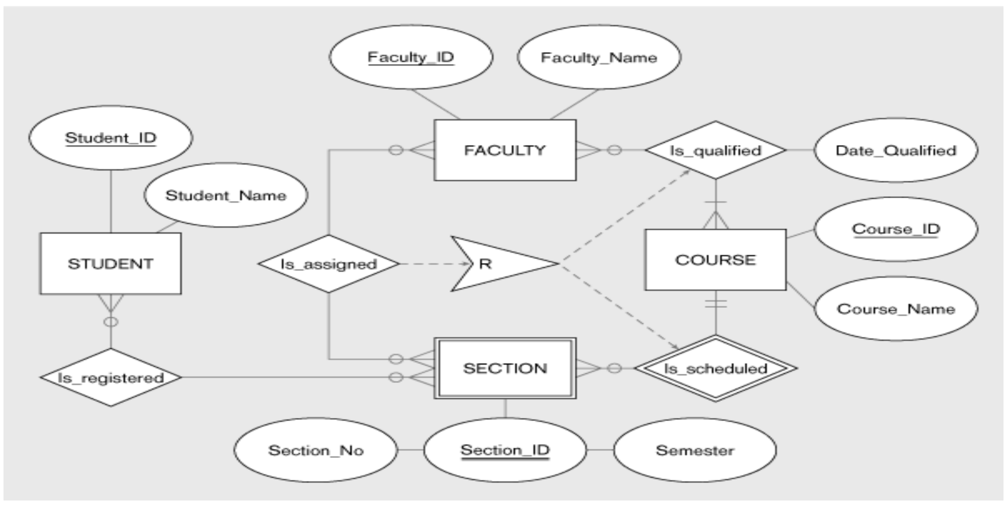
The more modern approach is to declare action assertions at a conceptual level without specifying how the rule will be implemented. Thus, there needs to be a specification language for business rules. We have seen that the EER notation works well for specifying many types of business rules. In fact, the EER notation was invented to allow more business rules to be shown in graphical form than the simpler E-R notation. An alternative to a graphical notation would be a structured grammar. Whether graphical or grammatical, two desirable features for a business rule specification language would be:

1. It should be relatively simple so that end users not only can understand the rule statements, but also can define the rules themselves.
2. The language should be sufficiently structured to be automatically convertible to the computer code that enforces the specification.

In the following section we illustrate both graphical and structured grammar approaches to specifying business rules. The graphical approach we used is adapted from Ross, a leader in the development of business rule specification.

Sample Business Rules In this section we augment the data model for class scheduling. Figure 5.16, with two new business rules.

Figure 5.17 Business Rule 1: For a faculty member to be assigned to teach a section of a course, the faculty member must be qualified to teach the course for which that section is scheduled



Business Rule 1 **For a faculty member to be assigned to teach a section of a course, the faculty member must be qualified to teach the course for which that section is scheduled.**

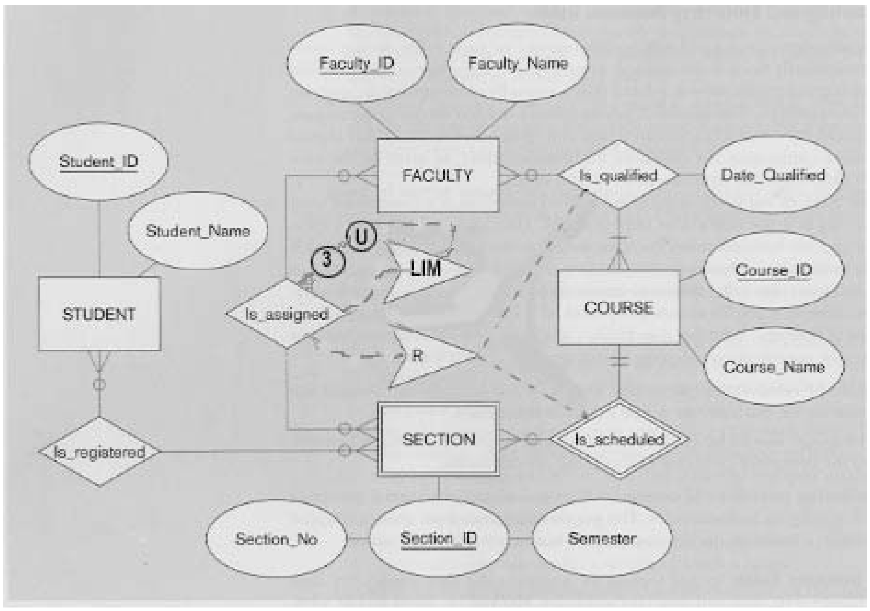
This rule refers to three entity types in figure 5.16: FACULTY, SCETION, and COURSE. The question is whether a faculty member can be assigned to teach a section. Thus the anchor object is the relation Is-assigned. We are not constraining the faculty member, nor are we constraining the section. Rather, we are constraining to assignment of the faculty member to the section. Figure 5.17 shows the dashed line from Is-assigned to the action assertion symbol.

What are the corresponding objects in this rule? For a faculty member to be assigned to teach a section of a course, two conditions are necessary.

1. The faculty member must be qualified to teach the course, and
2. The section must be scheduled for the course.

Thus in this case there are two corresponding objects. This facts is represented by the dashed lines from the action assertion symbol to each of the two relationships.

Figure 5.18 Business Rule 2: For a faculty member to be assigned to teach a section of a course, the faculty member must not be assigned to teach a total of more than three course sections.



Business Rule 2 For a faculty member to be assigned to teach a section of a course, the faculty member must not be assigned to teach a total of more than three course sections.

This rule impose a limitation on the total number of sections a faculty member may teach at a given time. Since the rule involves a total, it requires a modification of the previous notation.

As shown in figure 5.18, the anchor object is again the relationship Is-assigned. However, in this case, the corresponding object is also the relationship Is_assigned. In particular, it is a count of the total sections assigned to the faculty member. The letters “LIM” in the action assertion symbol stand for “limit.” The arrow leaving this symbol then points to a circle with the letter “U”, which stands for “upper.” The second circle then contains the number 3, which is the upper limit. Thus the constraint is read as follows: “The corresponding object is a count of the number of sections assigned to the faculty member, which has an upper limit of three.” If a faculty member is already assigned three sections, any transaction that attempts to add another section will be rejected.

You can implement business rules such as those above using the SQL language, and store the rules as part of the database definitions. One way to implement a rule with SQL is to use the CREATE ASSERTION statement, which is included in the most recent version of this language (SQL2). For example, consider Business Rule 2. The following statement creates an assertion named “Overload_Protect:”

```
CRREATE ASSERTION Overload_Protect

CHECK (SELECT COUNT(*)

FROM ASSIGNED

WHERE Faculty_ID=’12355’)<=3;
```

This statement checks whether the total number of course sections assigned to a particular faculty member is less than or equal to the limit. The database management system is responsible for ensuring that this constraint is not violated.

Many other business rules, beside the few examples of this section, are possible.

5.4.5 Identifying and Testing Business Rules

We have observed a wide variety of business rules and how they can be represented in structured grammar and EER diagrams and extensions. It is the job of a data analyst to document the specifications of business rules and then insure that the business rules are enforced as much as possible through database technologies. But, before business rules can be specified, they must be identified. Then, once specified, they should be tested before being implemented in technology.

5.5 Summary

E-R diagrams can become large and complex, including hundreds of entities. Many users and managers do not need to see all the entities, relationships, and attributes to understand the part of the database with which they are most interested. Entity clustering is a way to turn a part of an entity-relationship data model into a more macro-level view of the same data. An entity cluster is a set of one or more entity types and associated relationships grouped into a single abstract entity type. Several entity clusters and associated relationships can be further grouped into even a higher entity cluster, so entity clustering is a hierarchical decomposition technique. By grouping entities and relationships, we can layout an E-R diagram to allow you to give attention to the details of the model that matter most in a given data-modeling task.

A business rule is a statement that defines are constraints some aspect of the business. It is intended to assert business structure or control or influence the behavior of the business. Rules can be divided into three major categories: derivations, structural assertions, and action assertions. Structural assertions define the static structure of the organization. While action assertions are rules that constraint the dynamic operations of the organization. A structural assertion is a term or a fact. A term is definition of a concept about the business, and a fact is a statement involving in the association of two or more terms. Facts can be base, are fundamental, facts are derivations from mathematical or logical manipulations of other facts; they can deal with an attribute, relationship, or supertype/subtype association between facts.

Action assertions state that the action on some business rule, called the anchor object, are constrained by other corresponding objects. Constraints can be is.. the.. else Condition, integrity rules, or authorization privileges. An action assertion can permit the existence of the corresponding object, enable or disable an action, or cause an action to be taken. Other types of action assertions are simply guidelines that trigger notification of special condition.

5.6 Technical Terms

Entity Cluster: A set of one or more entity types and associated relationships grouped into a single abstract entity type.

Anchor object: A business rule (a fact) on which actions are limited.

Action: An operation, such as create, delete, update, or read, which may be performed on data objects.

Corresponding object: A business rule (a fact) that influences the ability to perform an action on another business rule.

5.7 Self Assessment Questions

1. Define the following terms.

    (a) Entity cluster (b) structural assertion (c) anchor object

2. Constraints the following terms.

    (a) structural assertion (b) action assertion

3.Give an example of each of the following

    (a) structural assertion

    (b) action assertion

4. What types of business rules are normally captured in an EER diagram.

5.8 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-6

LOGICAL DATABASE DESIGN AND THE RELATIONAL MODEL - I

6.0 Objective

To be able to

- Define the following key terms: relation, primary key, composite key, foreign key, null, entity, entity integrity rule, referential integrity constraint, well-structured relation, anomaly, recursive foreign key, normalization, normal forms.
- List the five properties of relations
- State two properties that are essential for a candidate key.
- Transform an E-R diagram to a logically equivalent set of relations

Structure

6.1 Introduction

In this lesson we describe logical database design, with special emphasis on the relational data model. Logical database design is the process of transforming the conceptual data model in the logical data model. First, the relational data model is most commonly used in contemporary database applications. Second, some of the principles of logical database design for the relational model apply to the other logical models as well. We next describe the process of transforming an E-R model into the relational model. Many CASE tools support this transformation today; however, it is important that you understand the underlying principles and procedures.

6.2 The Relational Data Model

The relational Data Model represents data in the form of tables. The relational model is based on mathematical theory and therefore has a solid theoretical foundational. However, we need only a few simple concepts to describe the Relational Model. The relational model consists of the following three components

6.2.1 Basic Definitions

1. Data Structure : Data are organized in the form of tables with rows and columns
2. Data Manipulation : Powerful operations (using the SQL) are used to manipulated data stored in the relations.
1. Data Integrity: Facilities are included to specify business rules that maintain the integrity of data when they are manipulated.

Relational Data Structure:

Relation A relation is a named, two-dimensional table of data. Table is made up of rows (records), and columns (attribute or field). Not all tables qualify as relations.

The Requirements to a table become as a Relation

1. Every relation has a unique name.
2. Every attribute value is atomic (not multivalued, not composite).
3. Every row is unique (can’t have two rows with exactly the same values for all their fields).
4. Attributes (columns) in tables have unique names.
5. The order of the columns is irrelevant.
6. The order of the rows is irrelevant.

Correspondence with ER Model

A) Relations (tables) correspond with entity types.

B) Rows correspond with entity instances.

C) Columns correspond with attributes.

E-R MODEL	**RELATIONS**
Entity type	Table
Entity instance	Record
Attribute	Column

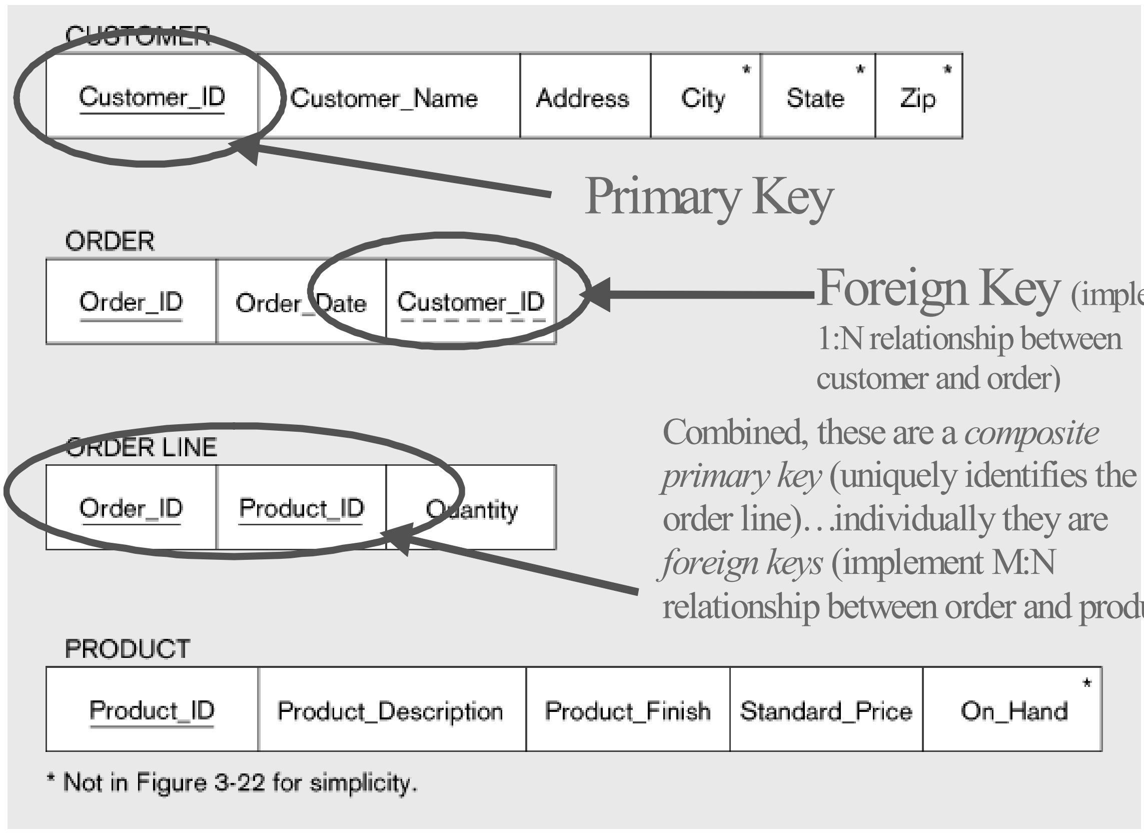
Keys Keys are special fields that serve two main purposes:

Primary key *Primary keys* are unique identifiers of the relation in question. Examples include employee numbers, social security numbers, etc. *This is how we can guarantee that all rows are unique.*

Foreign Key *Foreign keys* are identifiers that enable a dependent relation (on the many side of a relationship) to refer to its parent relation (on the one side of the relationship).

Keys can be ***simple*** (a single field) or ***composite*** (more than one field).

Figure Schema for four relations


CUSTOMER
Customer_ID
Customer_Name
Address
City *
State *
Zip *
Primary Key
ORDER
Order_ID
Order_Date
Customer_ID
Foreign Key (implements 1:N relationship between customer and order)
Combined, these are a composite primary key (uniquely identifies the order line)…individually they are foreign keys (implement M:N relationship between order and product)
ORDER LINE
Order_ID
Product_ID
Quantity
PRODUCT
Product_ID
Product_Description
Product_Finish
Standard_Price
On_Hand *
* Not in Figure 3-22 for simplicity.

6.3 Integrity Constraints

The relational data model includes several types of constraints, or business rules, whose purpose is to facilitate maintaining the accuracy and integrity of data in the database. The major types of integrity constraints are

Domain Constraints

Entity Integrity Constraints

Referential Integrity Constraints

Action assertions

6.3.1 Domain Constraints

All the values that appear in a column of a relation must be taken from the same domain. A domain is the set of values that may be assigned to an attribute. A domain definition usually consists of the following components: domain name, meaning, data type, size and allowable values.

Table 6.1 Domain Definitions for Selected Attributes

Attribute	Domain Name	Description	Domain
Customer_ID	Customer_Ids	set of all possible Customer Ids	char:size 20
Customer_Name	Customer_Names	set of all possible Customer names	char:size 20
Customer_Address	Customer_Address	set of all possible Customer addresses	char:size 30
Customer_City	Cities	set of all possible Cities	char:size 20
Customer_state	States	set of all possible States	char:size 2

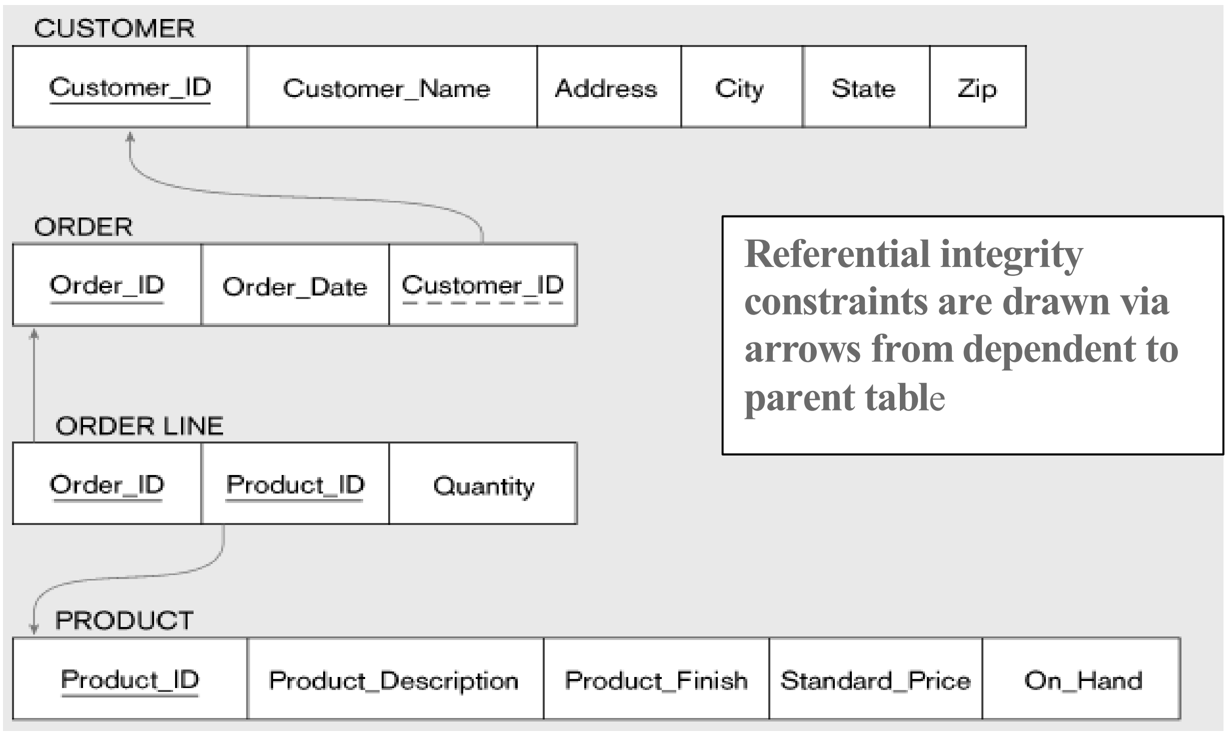
6.3.2 Entity Integrity

The Entity Integrity rule is designed to assure that every relation has a primary key, and that the data values for that primary key are all valid. In particular, every primary key attribute is non-null. Thus the entity integrity rule states “No primary key attribute may be null”.

6.3.3 Referential Integrity Constraints

In the relational data model, associations between tables are defined through the use of foreign keys. A rule that states that either each foreign key value must match a primary key value in another relation or the foreign key value must be null.

Figure. Referential integrity Constraints



For example: Delete Rules

A referential integrity constraint is a rule that maintains consistency among the rows of two relations. The rule states that if there is a foreign key in one relation, either each foreign key value must match a primary key value in another relation or foreign key value must be null.

6.3.4 Action Assertions

An **action assertion** is a statement of constraints or control on the action of the organization. It deals with the dynamic aspect of the organization. Examples of action assertion would include: ‘’A course must have a course name’’; ‘’A student must have a GPA > 2.2 to pass the course’’, or ‘’A professor can’t teach a class unless he is qualified to do so’’.

6.3.5 Creating Relational Tables

The table definitions are created using CREATE TABLE statements from the SQL data definition language. The Primary Key of each table using the PRIMARY KEY clause at the end of each table definition. The NOT NULL constraint can be used with non-primary key attributes. The FOREIGN KEY REFERENCES can also be defined with the table.

6.3.6 Well-Structured Relation

A relation that contains minimal redundancy and allows user to insert, modify, and delete the rows in a table without errors or inconsistencies.

The redundancies in a table may result in errors or consistencies (called anomalies) when a user attempts to update the data in a table. The three types of anomalies are possible: insertion, deletion and modification.

6.4 Transforming EER Diagrams into Relations

During Logical design you transform the E-R (and EER) diagrams that were developed during conceptual design into relational database schemas. Transforming (or mapping) E-R diagrams to relations is a relatively straightforward process with a well-defined set of rules.

6.4.1 Step 1: Map Regular Entities

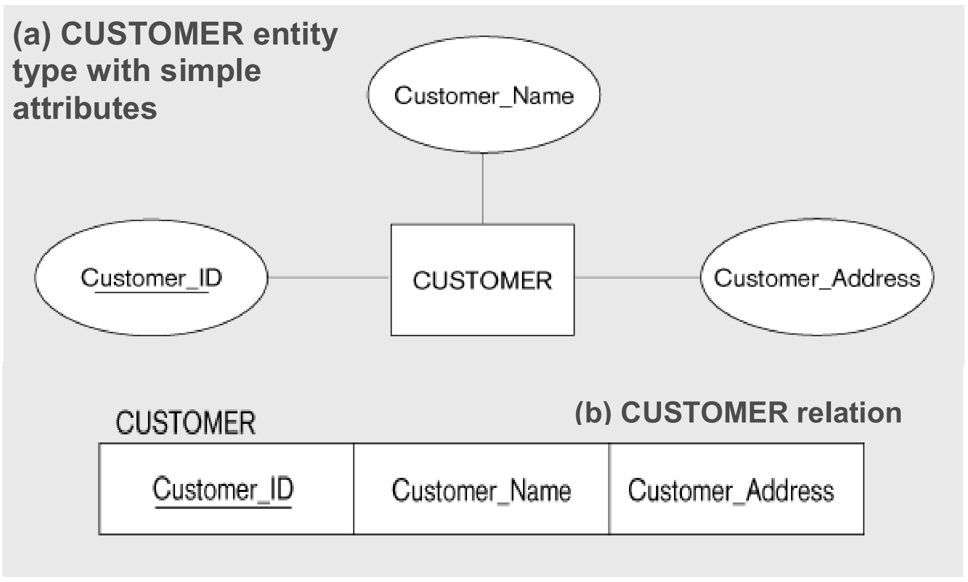
Each regular Entity type in a ER diagram is transformed into a relation. The name given to the relation is generally the same as the entity type. There are three types of regular entities.

Simple Attributes

Composite Attributes

Mulitivalued Attributes

Simple Attributes: Each regular entity type in an ER diagram is transformed into a relation. The name given to the relation is generally the same as the entity type. Each simple attribute of the entity type becomes an attribute of the relation. The identifier of the entity type becomes the primary key of the corresponding relation.



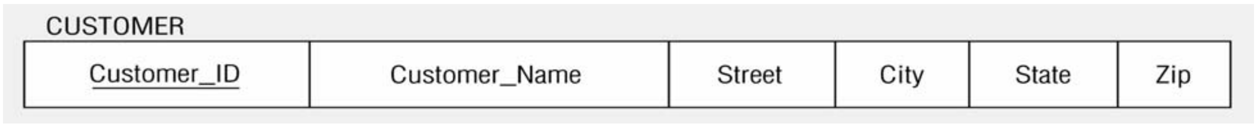
Customer_ID	Customer_Name	Customer_Address

Composite Attributes

A regular entity type has a composite attribute, only the simple component attributes of the composite attribute are included in the new relation.

Figure Mapping a composite attribute a) CUSTOMER entity type with composite attributes

b) CUSTOMER relation with address detail



CUSTOMER

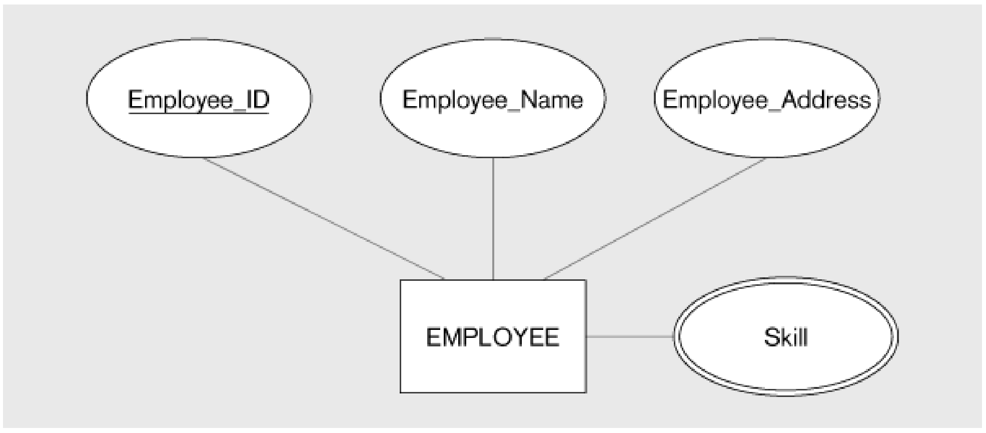
Customer_ID	Customer_Name	Street	City	State	Zip

Multivalued Attributes

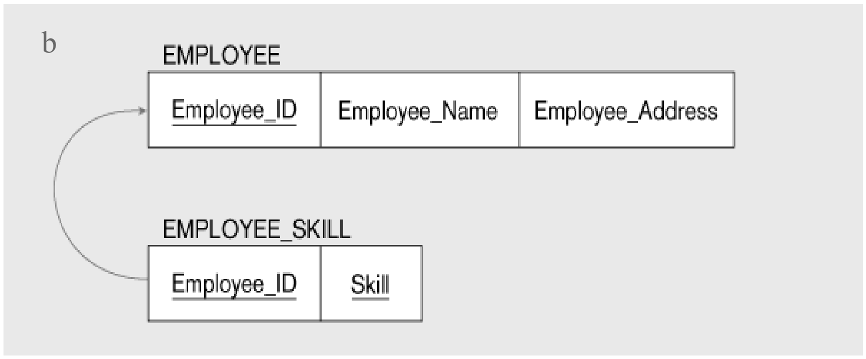
When a regular entity type contains a multivalued attribute, two new relations are created. The first relation contains all of the attributes of the entity type except the multivalued attribute.

The second relation contains two attributes together form the primary key of that relation. The first of these attributes is the primary key from the first relation, which becomes foreign key in this relation. The second is the multivalued attribute.

Figure. a) Employee entity type with Multivalued Attribute



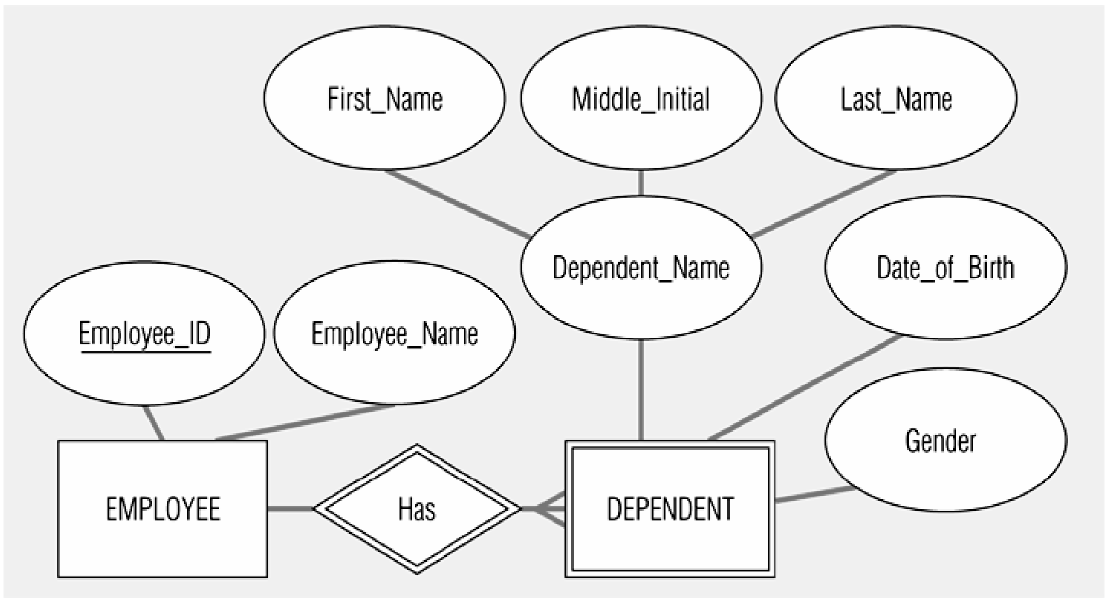
b) Mapping a multivalued attribute



6.4.2 Step 2: Mapping Weak Entities

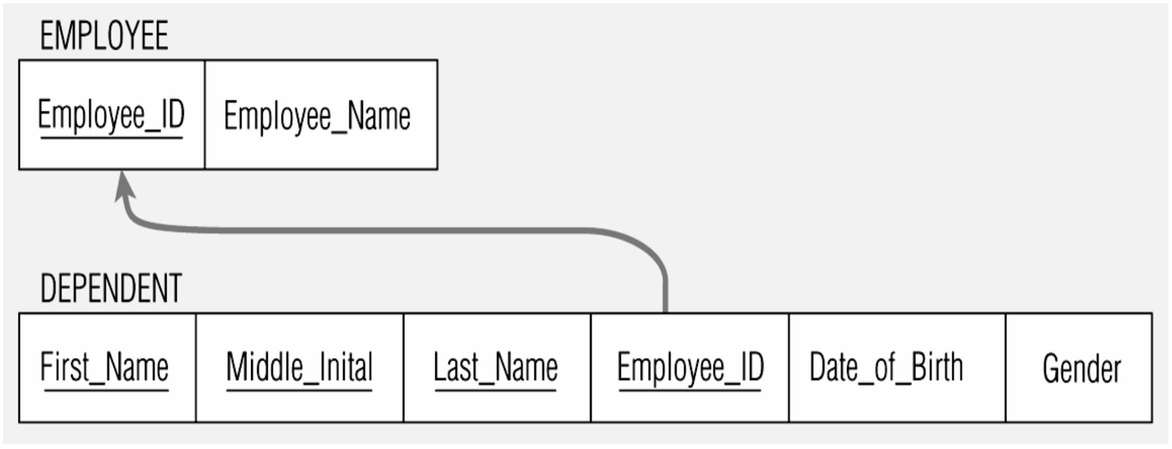
A weak entity type does not have a complete identifier, but must have an attribute called a partial identifier. First create a relation for strong entity with its attributes. Then For each weak entity type create a new relation and include all of the simple attributes as attributes of this relation. Then include *primary key* of the identifying relation as a *foreign key* attribute in this new relation. The primary key of the new relation is the combination of this *primary key* of the identifying relation and the partial identifier of the weak entity type.

Figure Example of mapping a weak entity
(a) weak entity DEPENDENT



The above figure (a) shows the weak entity type DEPENDENT and its identifying entity type EMPLOYEE.DEPENDENT_NAME is the partial identifier and also it is the composite attribute.

b) Relations resulting from weak entity



The primary key of DEPENDENT consists of four attributes. Namely Employee_Id, First_Nmae, Middle_Name, Last_Name. Date_of_Birth and Gender are non-key attributes. The arrow in the figure indicates the foreign key relationship with its primary key.

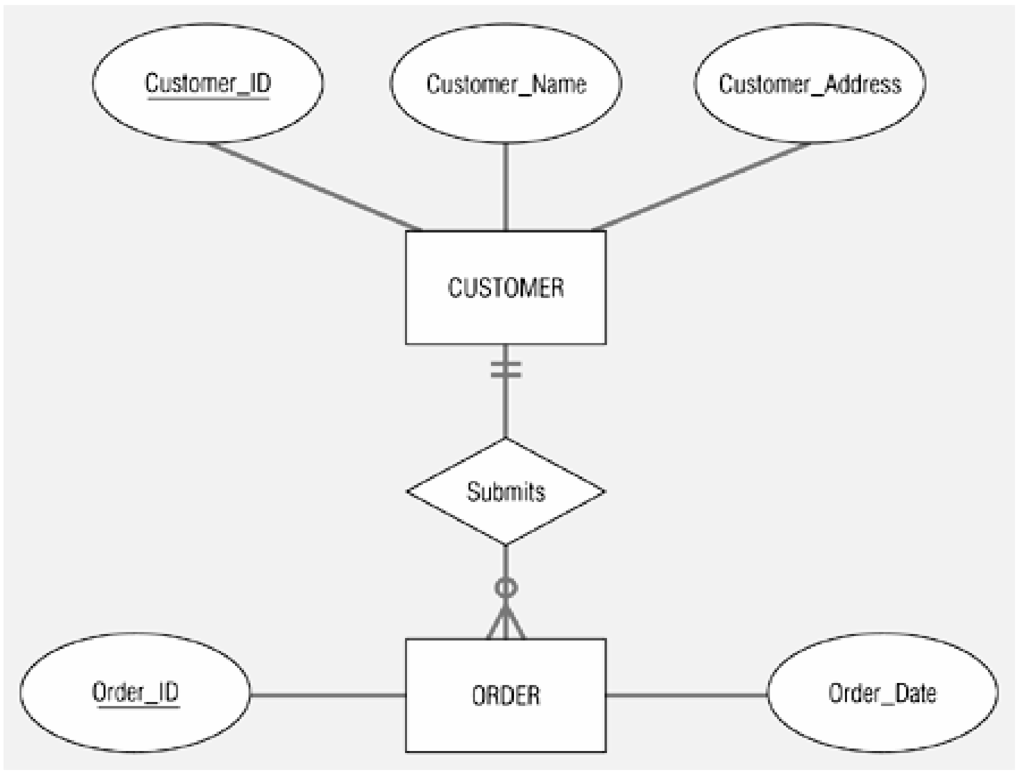
6.4.3 Step 3: Mapping Binary Relationships

The procedure for representing relationships depends on both the degree of the relationships (unary, Binary, Ternary) and the cardinalities of the relationship. There are three types of relationships.

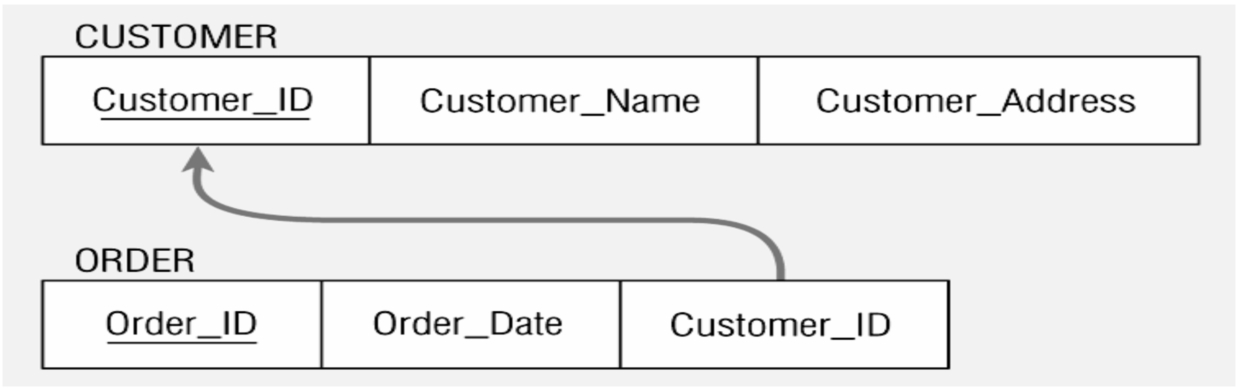
- Binary One-to-Many Relationships
- Binary Many-to-Many Relationships
- Binary One-to-One Relationships

Binary One-to-Many Relationships: For each binary 1:M relationship, first create a relation for each of the two entity types participating in the relationship. Include the primary key attribute of the entity one on one-side of the relationship as a foreign key in the relation that is on the many-side of the relationship.

Figure. Example of mapping 1: M relationship
Relationship between customers and orders



b) Mapping the relationship

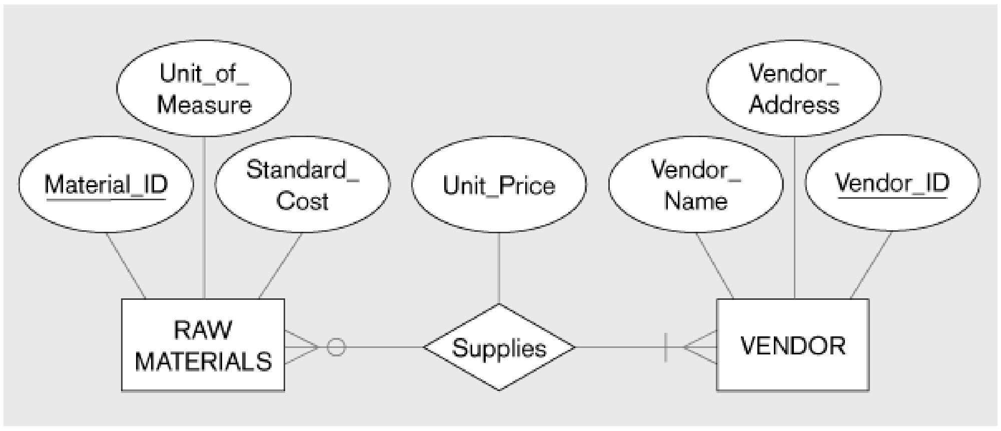


In above example, the primary key Customer_Id of CUSTOMER (the one side) is included as a foreign key in ORDER (the many-side). The foreign key relationship is indicated with an arrow.

Binary Many-to-Many Relationships

suppose that there is a binary many-to-many relationship between two entity types **A** and **B**. For such a relationship we create a new relation **C**. Include as foreign key attributes in **C** the primary key for each of the two participating entity types. These attributes become the primary key of **C**. Any non-key attributes that are associated with the M: N relationship is included with the relation **C**.

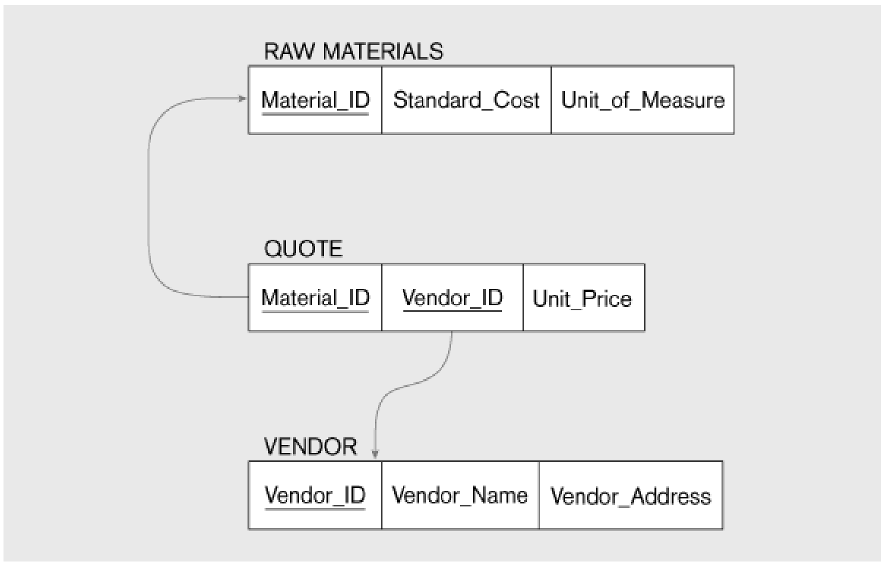
Example of Mapping Binary Many-to-Many Relationship
(a) Mapping Binary Many-to-Many Relationship



The above example represents supplies relationship between the

entity types VENDOR and REWMATERIALS. There are three relations that are formed from the entity types and the supplies relationship.

b) Three resulting relations



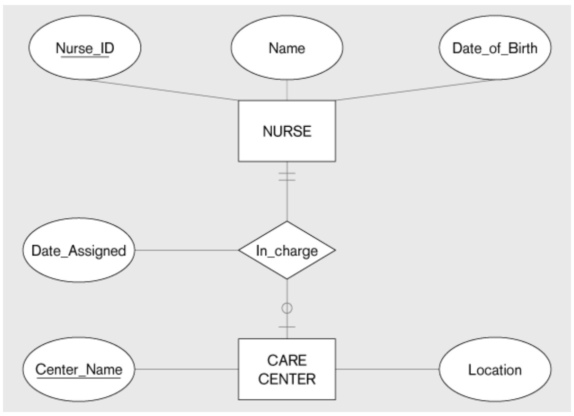
First a relation is created for each of the two regular entity types VENDOR and RAWMATERIAL. Then a relation QUOTE is created for the supplies relationship. The primary key of QUOTE is the combination of vendor_id and material_id, which are primary keys of VENDOR and RAWMATERIAL. The non-key attribute unit_price also included in QUOTE.

Binary One-to-One Relationships: Binary One-to-One relationships can be viewed as a special case of one-to-many relationships. The process of mapping such a relationship to relations requires two steps.

1. Create a relation for the mandatory side entity type.
2. Create a relation for the optional side entity type with it's own Attributes. Then add *Primary key* on the mandatory side becomes a *Foreign key* on the optional side.

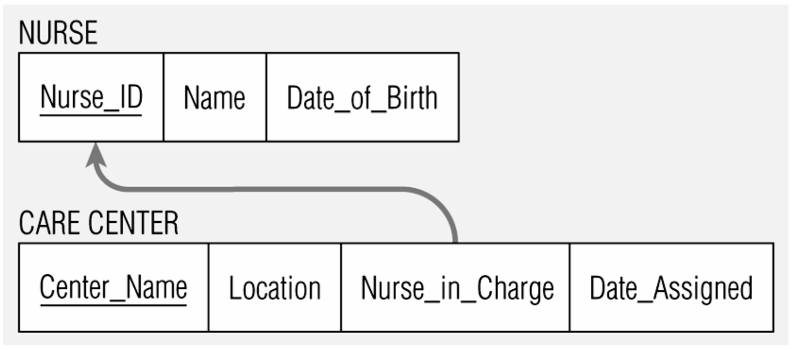
Example Mapping binary 1 : 1 Relationship

(a) Binary 1 : 1 Relationship



The above example represents a binary one-to-one relationship between the entity types NURSE and CARECENTER. Each CARECENER must have NURSE who is in charge of that center. Thus the association from CARECENTER to NURSE is a mandatory one, while the association from NURSE to CARECENTER is an optional one. The attribute Date_Assigned is attached to the in-charge relationship.

(b) Resulting relations



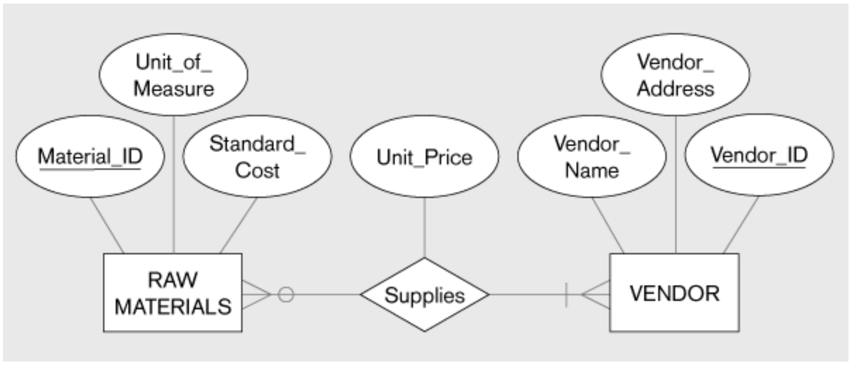
The two relations NURSE and CARECENTER are created from the two entity types. Since CARECENTER is the optional participant, the foreign key is placed in this relation. In this case the foreign key is called Nurse_in_Charge. It has the same domain as Nurse_id. The attribute date_assigned is also located in CARECENTER and would not be allowed to be null.

6.4.4 Step 4: Mapping Associate Entities

The first step is to create three relations one for each of the two participating entity types and the third for the associative entity. The relation formed from the associative entity is referred as the associative relation. The second step then depends on whether on the E-R diagram an identifier was assigned to the associative entity.

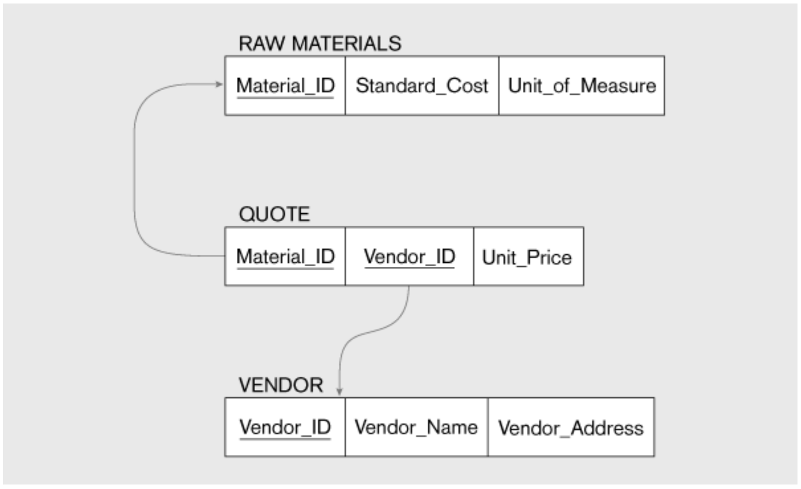
Identifier not Assigned: If an identifier was not assigned, the default primary key for the associative relation consists of the two primary key attributes from the other two relations.

Figure Mapping An associative entity: (a) An Asoociative entity



The above example represents supplies relationship between the entity types VENDOR and REWMATERIALS. There are three relations that are formed from the entity types and the supplies relationship.

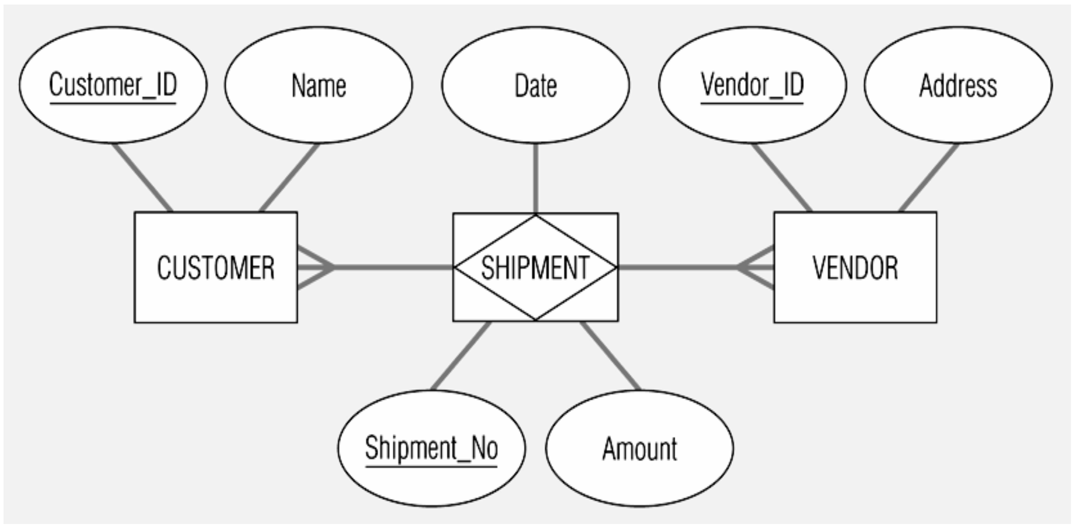
b) Three resulting relations



Identifier Assigned: create two relations one for each of the participating entity type.

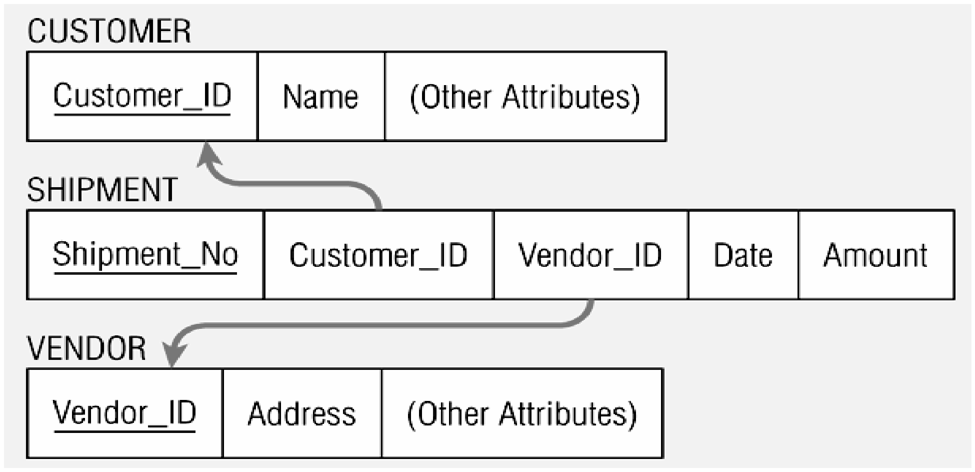
Create a new relation for the associative entity. The primary key for this relation is the identifier assigned on the E-R Diagram .The primary keys for the two participating entity types are then included as foreign keys in the associative entity.

Figure : Mapping an associative entity: (a) Associative entity



The above ER diagram shows the associative entity type SHIPMENT that links the CUSTOMER and VENDOR entity types.

(b) Three resulting relations



The result of mapping relations is shown in the above figure. The new associative relation is named SHIPMENT. The primary key is Shipment_No. Customer_Id and Vendor_Id are included as foreign keys in this relation. Date and amount are non-key attributes.

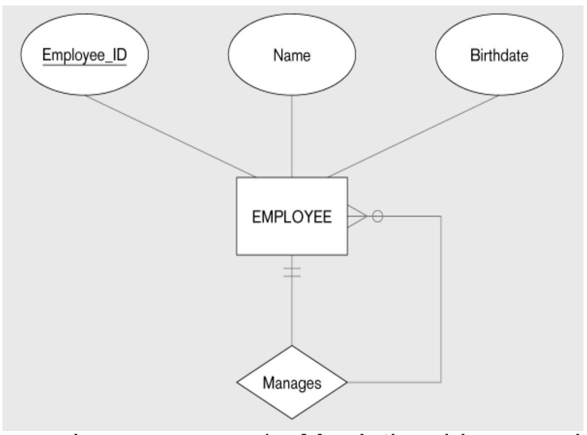
6.4.5 Step 5:Mapping Unary Relationships

Unary relationship is a relationship between the instances of a single entity type. The two most important cases of unary relationships are one_to_many and many_to_many.

Unary One-to-Many Relationships: create a relation for entity type. Then a foreign key attribute is added within the same relation that references the primary key values of the same table.

Recursive foreign key: A foreign key in a relation that references the primary key Values of that same relation.

Figure : Mapping a Unary 1 : N relationship
(a) EMPLOYEE entity with manages relationship



The above ER diagram shows a unary 1 : M relationship named manages that associate each employee of an organization with another employee who is his manager. Each employee has exactly one manager.

(b) EMPLOYEE relation with Recursive Foreign Key

The EMPLOYEE relation that results from mapping this entity relationship is shown in the above figure. The (recursive) foreign key in the relation is named Manager_ID. This attribute has the same domain as the primary key Employee_ID.

Unary Many-to-Many Relationships: with this type of relationship, two relations are Created. One to represent the entity type in the relationship and the other an associative.

Relation to represent the M : N relationship itself. The primary key of the associative relation consists of two attributes. These two attributes take their values from the primary key of the other relation. Any non-key attribute of the relationship is included in the associative relation.

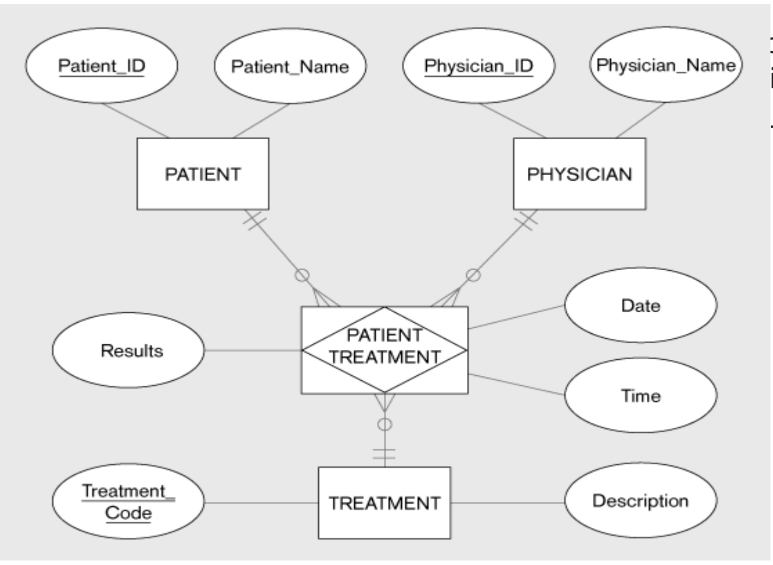
Figure : Mapping a Unary M:N Relationship(a) Bill-of-materials relationships (M:N)

The above figure shows a bill-of-materials relationship among items that are assembled from other items or components. The relationship is M:N since a given item can contain numerous component items, and an item can be used as a component in numerous other items.

(b) ITEM and COMPONENT Relations

The relations that result from mapping this entity and its relationship are shown in the above figure. The ITEM relation is mapped directly from the same entity type. COMPONENT is an associative relation whose primary key consists of two attributes that are arbitrarily names Item_No and Component_No. the attribute Quantity is a non-key attribute of this relation that for a given item. Notice that both Item_No and Component_No reference the primary key(Item_No) of the ITEM relation

6.4.6 Step 6: Mapping Ternary (and n-Ary) Relationships

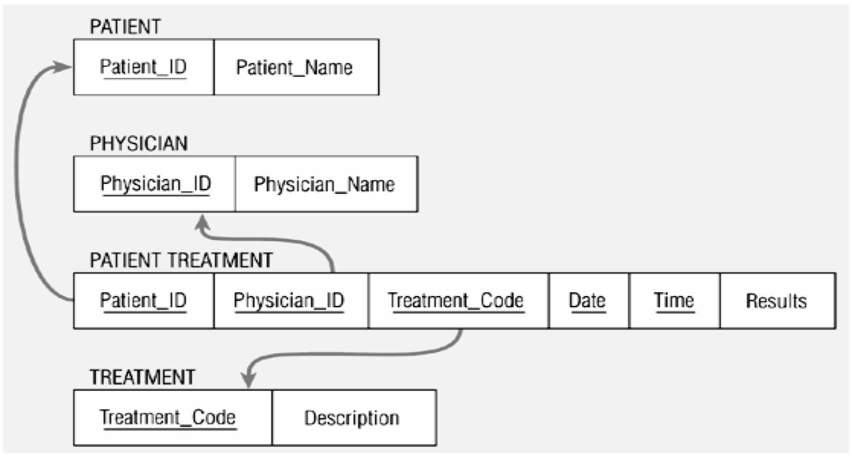


r the associative entity. The Associative entity
ip

Ternary relationship with associative entity

The above ER diagram that represents a patient receiving a treatment from a physician. The associative entity type PATIENT TREATMENT has the attributes date, time, and results. These values are recorded for these attributes for each instance of PATIENT TREATMENT

(b) Mapping the ternary relationship

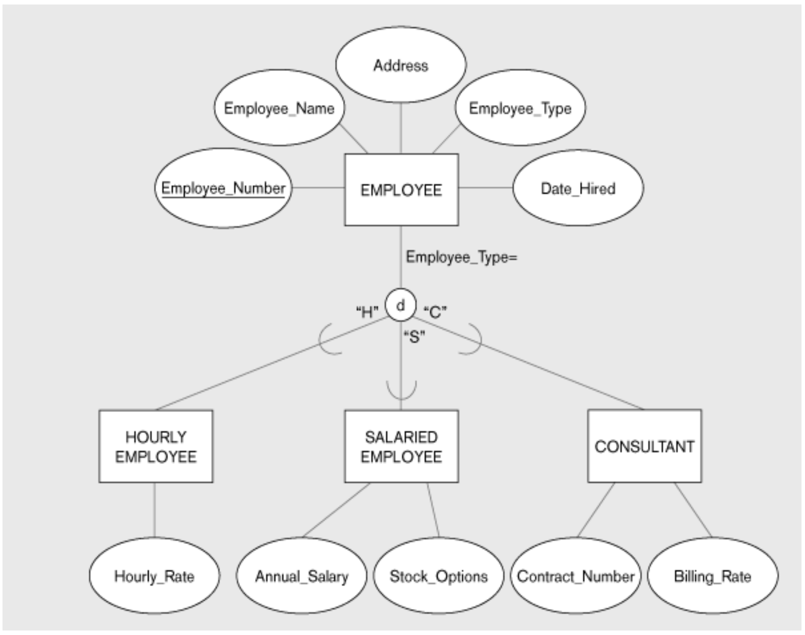


In the above diagram the primary key attributes patient_id, physician_id and treatment _code become foreign key in PATIENT TREATMENT. These attributes are components of the primary key of PATIENT TREATMENT.

6.4.7 Step 7: Mapping Supertype/Subtype Relationships

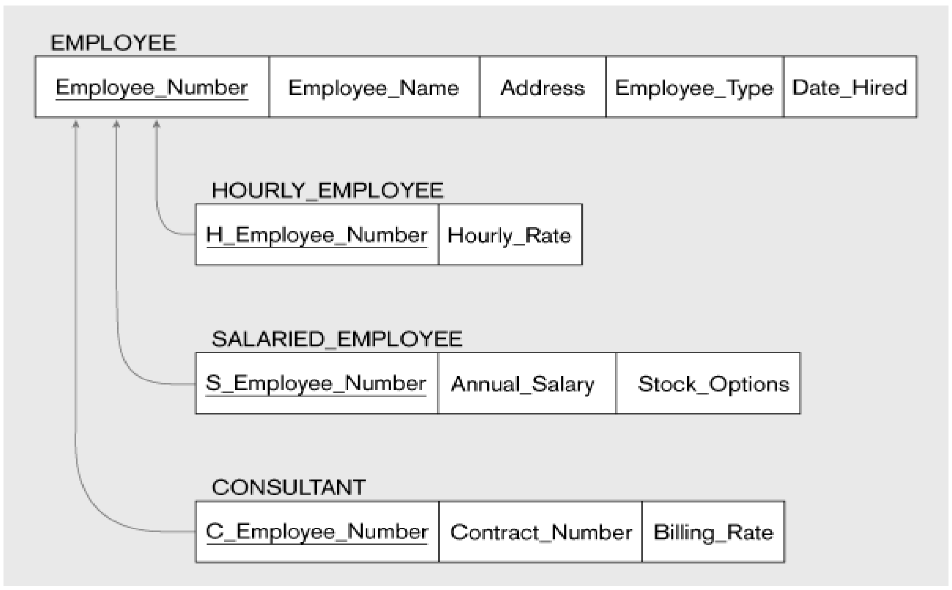
Create one relation for supertype and for each subtypes. Supertype attributes (including identifier and subtype discriminator) go into supertype relation. Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation. The 1:1 relationship established between supertype and each subtype, with supertype as primary table.

Figure : Mapping a ternary relationship a) Ternary relationship with associative entity



The above diagram shows the supertype EMPLOYEE with subtypes HOURLY EMPLOYEE , SALARIED EMPLOYEE , and CONSULTANT. The primary key of EMPLOYEE is Employee_Number, and the attribute Employee_Type is the subtype discriminator.

(b) Mapping Supertype/subtype relationships to relations



In the above diagram one relation for the supertype(EMPLOYEE) and one for each of three subtypes. The primary key for each of the four relations is Employee_Number. For example S_Employee_Number is the name for the primary key of the relation SALARIED_EMPLOYEE. Each of these attributes is a foreign key that references the supertype primary key, as indicated by the arrow in the diagram. Each subtype relation contains only those attributes peculiar to the sub-type.

6.8 Summary

Logical database design is the process of transforming the conceptual data model into a logical data model. The emphasis in this lesson has been on the relational data model, because of its importance in contemporary database systems. The relational data model represents data in the form of tables called relations. A relation is a named, two-dimensional table of data. A key property of relations is that they cannot contain multivalued attributes.

In this lesson, we described the major steps in the logical database process. This process is based on transforming E-R diagrams to normalized relations. The three steps in this process are the following: Transform E-R (and EER) diagrams to relations.

Each entity type in the E-R diagram is transformed to a relation that has the same primary key as the entity type. A one-to-one relationship is represented by adding a foreign key to the relation that represented the entity on the many-side of the relationship. A many-to-many relationship is represented by creating a separate relation.

The relational model does not directly support supertype/subtype relationships, but we can model these relationships by creating a separate table for the supertype and for each subtype.

6.9 Technical Terms

Composite Key: A primary key that consists of more than one attributes

Null: A value that may be assigned to an attribute when no other value applies or when the applicable value is unknown.

Entity integrity rule: No primary key attribute can be null.

Referential integrity Constraint: A rule that states that either each foreign key value must match a primary key value in another relation or the foreign key value must be null.

Anomaly: An error or inconsistency that may result when a user attempts to update a table that contains redundant data. The three types of anomalies are insertion, deletion, and modification.

6.10 Self Assessment Question

1. Define the following terms

   Determinant, transitive dependency, composite keys, relation,

   normalization

2. Contrast the following terms

   Candidate key; primary key

   Composite key; recursive foreign key

3. Summarize six important properties of relations.
4. Describe two properties that must be satisfied by candidate keys
5. What is the wee-structures relation? Why are well-structures relation important logical database design?
6. Describe how the following components of E-R Diagram are transformed to relations:

   Regular entity type

   Relationship(1:M)

   Multivalued attributes

   Weak entity

6.11 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-7

LOGICAL DATABASE DESIGN AND THE RELATIONAL MODEL - II

7.0 Objective

To be able to

- ☞ State two properties that are essential for a candidate key.
- ☞ Give a concise definition of each of the following: first normal form, second normal form, and third normal form.
- ☞ Briefly describe four problems that may arise when merging relations.

Structure

7.1 Introduction To Normalization

Whenever we design databases we are faced with a number of problems relating to things like data integrity, security, and efficiency. We are also faced with problems relating to the structure of the data we are planning to use.

Normalization is a design technique that is widely used as a guide in designing relational databases. Normalization is essentially a two-step process that puts data into tabular form by removing repeating groups and then removes duplicated data from the relational tables. Normalization theory is based on the concepts of **normal forms**. A relational table is said to be a particular normal form if it satisfied a certain set of constraints. There are currently five normal forms that have been defined. In this section, we will cover the normal forms that were defined by E. F. Codd.

Normalization: **The process of decomposing relations with anomalies to produce smaller, well-structured relations.**

7.2.1 Steps in Normalization

Normal Form: A state of a relation that results from applying simple rules regarding functional dependencies to that relation. Normalization is based on the concept of functional dependency.

1NF:

- Any multivalued attributes have been removed
- Every attribute value is atomic

2NF

Any partial functional dependencies have been removed.

3 NF:

Any transitive dependencies have been removed.

BCNF:

Any remaining anomalies that result from functional dependencies have been removed.

4 NF:

Any multivalued dependencies have been removed.

5 NF:

Any remaining anomalies have been removed.

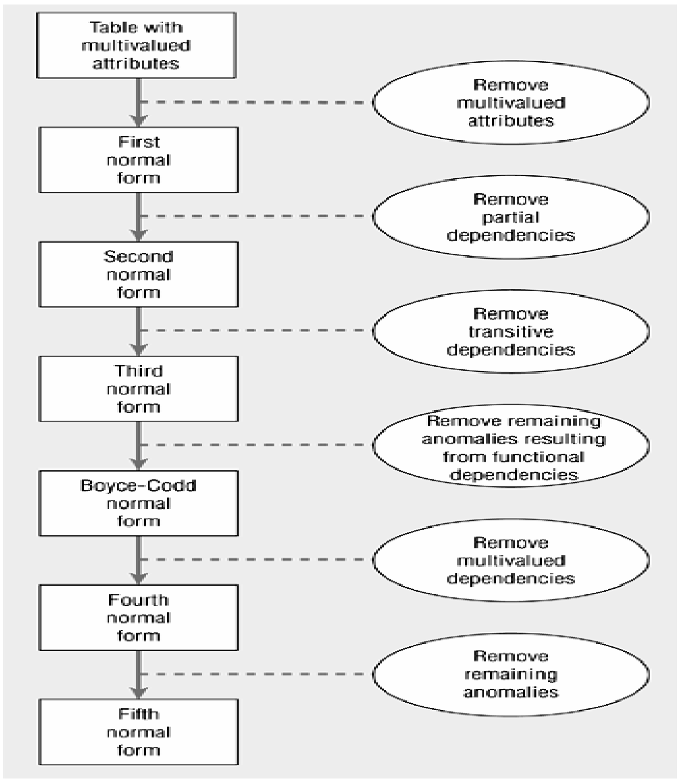


Table with multivalued attributes
Remove multivalued attributes
First normal form
Remove partial dependencies
Second normal form
Remove transitive dependencies
Third normal form
Remove remaining anomalies resulting from functional dependencies
Boyce-Codd normal form
Remove multivalued dependencies
Fourth normal form
Remove remaining anomalies
Fifth normal form

Functional dependency - is a dependency (constraint) between 2 attributes or 2 sets of attributes.

It is denoted as: A —> B means that B is functionally dependent on A. In other words, a value of A uniquely determines the value of B. The attribute on the left is called Determinant.

Example 1:

SSN — > Name, Address, Birthday.

Namely, a person's name, address and birthday are functionally dependent on SSN

Example 2:

Student_ID, Course_name —> Date_completed.

Student_ID, Course_name —>Grade.

Namely, the date the course is completed (or of the grade), is uniquely determined by the identity of the Student (employee) and the title of the course.

Determinant The attribute on the left-hand of the arrow in a functional dependency.

Candidate Key: is an attribute, or combination of attributes that uniquely identify a row in a relation.

In the EMPLOYEE1 relation, Emp_ID **—>** Name, Dept_Name, Salary. Thus, the Emp_ID uniquely define an employee (and since there are no other candidate keys) it is the primary key of this table.

The functional dependency is shown graphically as lines pointing from the primary key to the non-key attributes that are functionally dependent on the primary key., we show the dependency: Emp_ID **—>** Name, Dept_Name, Salary

7.3 The Basic Normal Forms

Now that we have examined functional dependencies and keys, we are ready to describe and illustrate first through third normal forms. We also describe the normalization of summary data that appear in information bases.

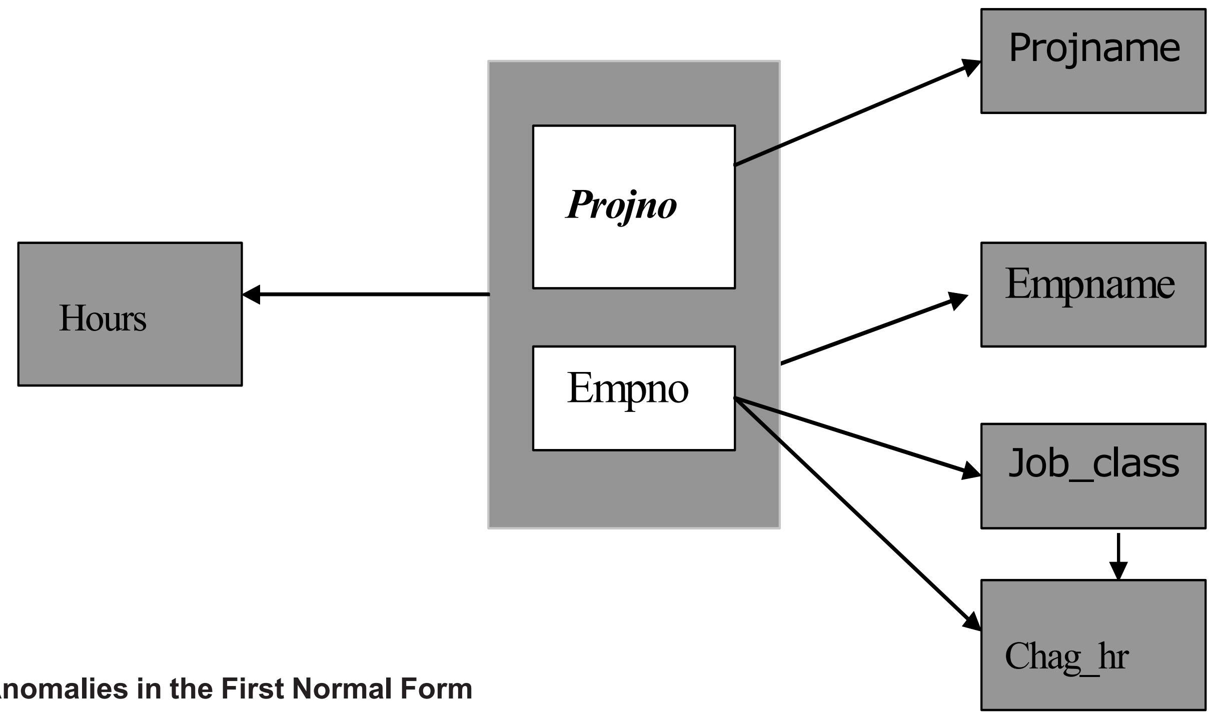
7.3.1 First Normal Form

The first of the normal forms that we study, **first normal**, imposes a very basic requirement on relations; unlike the other normal forms, it does not require additional information such as functional dependencies.

First Normal Form A relation that contains to no multivalued attributes.

Now, the above relation is in first normal form, because it has no repeating groups.

The below figure shows the FD diagram



Anomalies in the First Normal Form

Insertion: We can't enter about a project to which employees are not assigned.

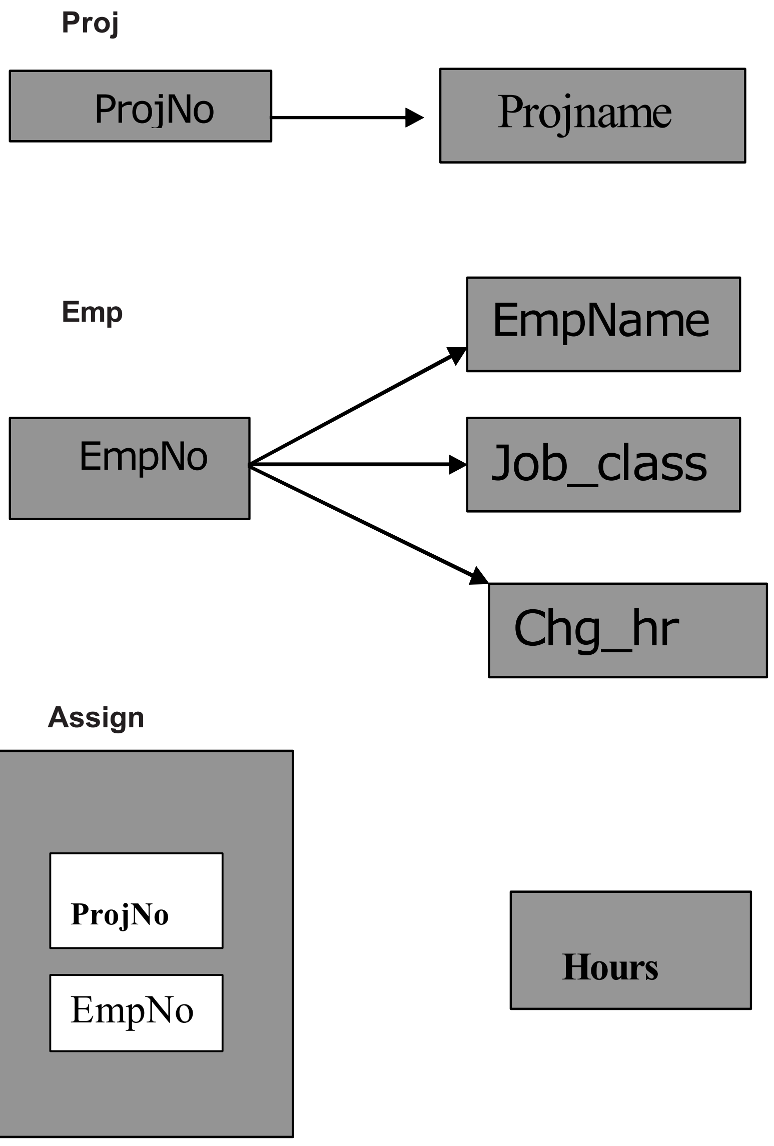
Deletion If we delete a row about a project of team strength 1 and if that employee isn't involved in any other project then we lose both project info and emp information.

Updation To change project name from x to y we face with the problem of changing wherever it occurred or will be left with inconsistent information So overcome anomalies convert the relation PROJECT in to second normal form.

7.3.2 Second Normal Form

A Relation is said to be in second normal form it is already in first normal from and if all **non key attribute** is fully functionally dependent on the Primary key.

(A **non key attributes** or **non prime attributes** is any attribute that does not participate in the primary key of the relation concerned.**)**



Clearly PROJECT is not in 2NF

So we **decompose** relation PROJECT into the following relations: **Decomposition** means without losing of data or information dividing the relation in to multiple relations**).** Clearly the relations

Proj (ProjNo,ProjName)

Assign (ProjNo, EmpNo, Hours)

Emp (Empno,EmpName,Job_Class,Chg_hr)

are in Second normal form (2NF) .

Emp :

Empno	Empname	job_class	Chg_hr
1103	Krishna	DB Designer	500
1109	Venki	Programmer	200
1102	Tom Cruise	Analyst	350
1101	Nag	Appln Designer	400
1104	Hari	Analyst	350
1105	Siri	Programmer	200
1111	Praveen	Programmer	200

Proj **Assign**

But still there exist few anomalies.

Insertion we can't enter the fact that a particular employee charges Rs. 999 until his job class is specified.

Deletion If we delete a row about an employee we lose both emp info, job class info.

Updation To change charge per hour of an employee we face with the problem of changing wherever it occurred or will be left with inconsistent information.

So to overcome these anomalies we convert Emp relation into 3NF

7.3 3 Third Normal Form

A Relation is said to be in third normal form if it is already in second normal form and it has no **transitive dependency.**

A Relation is in third normal form if and only if it is in second normal form and every non-key attribute is non-transitively dependent on the primary key.

(In a relation there may be dependency among non key fields. such dependency is called a s **transitive dependency** (a->b, b->c and a->c)**)**

The FDs in emp relation are :

Empno	→	Empname
Empno	→	job_class
Empno	→	chag_hr
job_class	→	chag_hr

in the emp relation there exists a transitive, in the following way .

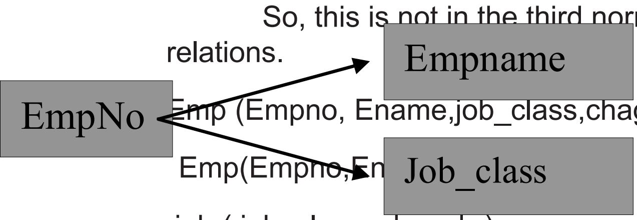
Empno→job_class	(a→b)
job_classàchag_hr	(b→c)
Empnoàchag_hr	(a→c)

So, this is not in the third normal form (3NF), so we decompose the emp relation in to two relations.

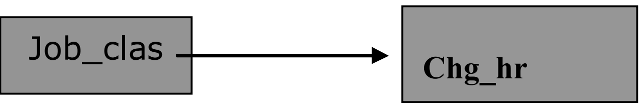
Emp (Empno, Ename,job_class,chag_hr) is decomposed in to the following two relations.

Emp(Empno,En

job (job_class, chag_hr)



Emp



Job

7.3.4 Normalizing Summary Data

Databases to support managerial decision making in an organization often contain subsets and summaries of data from operational databases. These " data warehouse" used to support higher levels of management are often normalized to avoid the same anomalies that arise in operational databases.

7.4 Merging Relations

This material pertains to situations where parts of a database have been separately designed and now need to be merged. This can happen in at least three ways:

☞ Working in teams on large projects

☞ Integrating existing databases with new information requirements and hence different views

☞ New data requirements may emerge during the life-cycle of a DB system

As a result of this activity, we may sometimes end up with tables representing the same entities. If we are going to attempt to correct this problem, we need to have a good understanding of the meaning of our data and the representation issues related to our table. Sometimes, duplication and/or overlap than can occur in merging DB are not obvious. There are at least four pretty obvious situations in which this can happen.

7.4.1 An Example

Suppose that modeling a user view results in the following 3NF relation:

EMPLOYEE1(Employee_ID, Name, Address, Phone)

Modeling a second user view might result in the following relation:

EMPLOYEE2(Employee_ID, Name, Address, Jobcode, No_Years)

Since these two relations have the same primary key (Employee_ID), they likely describe the same entity and may be merged into one relation. The result of merging the relations is the following relation.

EMPLOYEE(Employee_ID, Name, Address, Phone, Jobcode, No_Years).

Notice That an attributes that appears in both relations appears only in the merged relation.

7.4.2 View Integration Problems

When integrating relations as in the preceding example, the database analyst must understand the meaning of the data and must be prepared to resolve any problems that may arise in that process. In this we describe and briefly illustrate four problems that arise in view integration: Synonyms, homonyms, transitive dependencies, and supertype/subtype relationships.

Synonyms Two or more attributes with different names but the same meaning (in other words, they describe the same attributes of a particular entity. (Example: Student_ID, Registration_ID - both related to a student SSNO)

Homonyms The same attribute used in defining more than one entity (Example: “Account_ID” in a banking DB, where the account could refer to a Checking Account ID or a Savings Account ID, or a Money Market ID, etc. You need to be very careful with homonyms when merging relations. They may not at all refer to the same attribute.

Transitive Dependencies

When two 3NF relations are merged to form a single relation, transitive dependencies may result. For example

STUDENT A [Student_ID, Major]
STUDENT B [Student_ID, Advisor]

It appears as though these can be merged to form

STUDENT [Student_ID, Major, Advisor]

However: If all majors have the same advisor (as is the case in CIS) then Advisor is dependent on Major.

Here we have an entity, STUDENT that is in 2nd Normal Form but not 3rd NF. To fix this, we create two entities:

STUDENT [Student_ID, Major]
MAJOR_ADVISOR [Major, Advisor]

Supertype/subtype Relationships

These relationships may be hidden in user views or relations. Suppose that we have the following two hospital relations:

Patient1(Patient_ID, Name, Address)

Patient2(Patient_ID, Room_No)

Initially it appears that these two relations can be merged into a single PATIENT relation. However, the analyst correctly suspects that there are two different types of patients: resident patients and outpatients. PATIENT1 actually contains attributes common to all patients. PATIENT2 contains an attribute(Room-No) that is a characteristics only of resident patient. In this situation, the analyst should create supertype/subtype relationships for these entities:

Patient1(Patient_ID, Name, Address)

RESIDENT Patient(Patient_ID, Room_No)

OUTPATIENT(Patient_ID, Date_Treated)

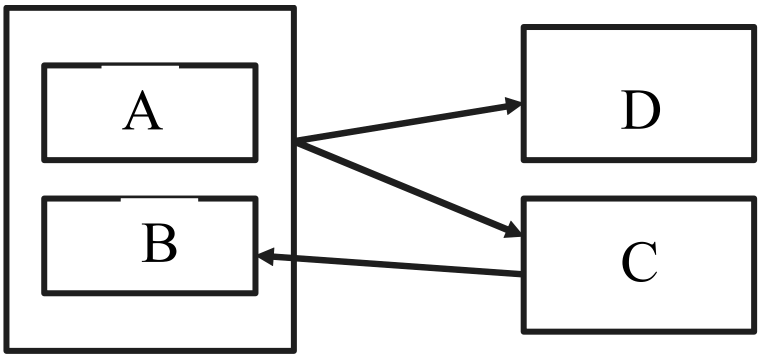
For an extended I discussion

7.5 Boyce-Codd Normal Form(BCNF)

A Relation is said to be in BCNF if and only if it is already in third normal form and every **determinant** is a candidate key.

(A relation is in BCNF if and only if the only **determinant**s are candidate keys).

(a **determinant** is any field (field or composite key)on which some other field is fully functional dependent**)**



Consider the relation SPT

Student	**Paper**	**Teacher**
Siddhu	Maths	Prof. MSR
Kishore	AI	Prof. KK
Ramki	Maths	Prof . MSR
Jones	Physics	Prof. GSR
Sirisha	Physics	Prof. Surya

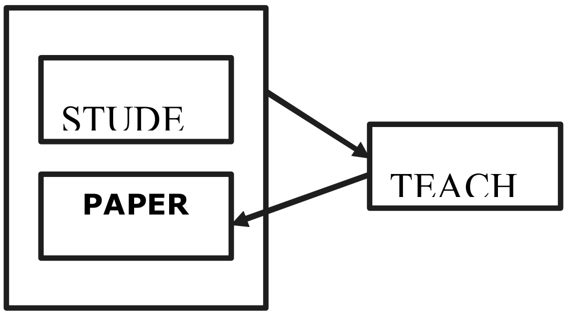
Consider the relation about student, subject and teacher.

Several teachers can teach a subject.

Example : T1 ,T3 can teach subject S1. A teacher may teach only one subject. Only one teacher will teach a particular subject to a student.

Prof. MSR teaches only Maths, Prof. Surya , Prof GSR teaches Physics , but for different students.

The FDD is



The relation SPT is in 3NF but not in BCNF .

Anomalies

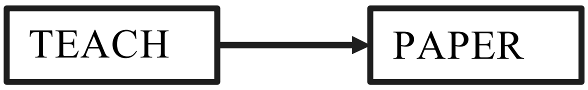
Delete : If we delete info that Sirisha is studying Physics ,we lose info that Prof. Surya teaches Physics. There is difference between Determinant and Candidate Key .(Teacher is Determinant but not Candidate Key)

Similarly Insert ,Update Anomalies

So we decompose SPT relation into

(S,T) S,T

(T, P) T,P

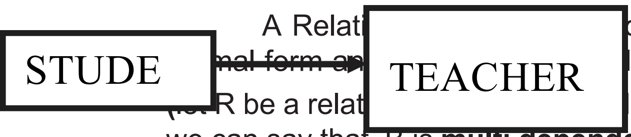


NOW ST TP are in BCNF

Student	Teacher
Siddhu	MSR
Kishore	KK
Ramki	MSR
Jones	GSR
Sirisha	Surya

Teacher	Paper
MSR	Maths
KK	AI
GSR	Physics
SURYA	Physics

7.6 Fourth Normal Form (4NF)



A Relati ourth normal form if and only if it is already in Boyce –Codd mal form an **lued dependencies**.

R be a relat C are the arbitrary subsets of the set of attributes of R, then we can say that B is **multi dependent** on A. It is denoted by AààB.)

(consider three fields X, Y and Z in a Relation . if for each value of X, there is well-defined set of values of Y and a well defined set of values of Z and the set of values of Y is independent of the set of values of Z , then **multivalued dependency** exists**)**

Multivalued Dependencies

The type of dependency that exists when there are at least three attributes in a relation, with a well-defined set of B and C values for each A value, but those B and C values independent of each other.

Fifth Normal Form (5NF)

A Relation is said to be in Fifth Normal Form if and only if it is already in fourth normal form and it has no **join dependencies.**

A Relation R is in Fifth normal form or Projection Normal Form if and only if every join dependency in R implied by the candidate key of R.

(let R be a Relation, and let A, B……..Z be arbitrary subsets of the set of attributes of R. then we say that R satisfies the **JD (Join Dependency)**

*(A, B, C…Z)

if and only if R is equal to the join of its projections on A,B,C,…………Z**)**

(A Relation who has a join Dependency cannot be decomposed by projection into other relations without any difficulty and undesirable results. The occurrence of this type of dependency is very large)

7.7 Higher Normal Forms

At least two higher-level normal forms have been defined: fifth normal form (%NF) and domain-key normal form(DKNF). Fifth normal form deals with a property called " lossless joins. these cases occur very rarely and are difficult to detect in practice. For this reason, we do not describe %NF in this text.

Domain key normal form(DKNF) is an attempt to define an " ultimate normal form" that takes into account all possible types of dependencies and constraints. Although the definition of DKNF is quite simple, according to these authors " its practical utility limited". For this reason we do not describe DKNF in this text

7.8 Summary

The relational model does not directly support supertype/subtype relationship, but we can model these relationships by creating a separate table. The primary key of each subtype is the same as for the subtype. The supertype must have an attribute called the subtype discriminator that indicates to which subtype each instance of the supertype belongs. The purpose of normalization is to device well-structured relation that are free of anomalies that would otherwise result when the relationships are updated or modified. Normalization is based on the analysis of functional dependencies, which are constraints between two attributes. It may be accomplished in several stages.

First normal form: A table is in the first normal form if it contains no repeating columns.

Second normal form: A table is in the second normal form if it is in the first normal form and contains only columns that are dependent on the whole (primary) key.

Third normal form: A table is in the third normal form if it is in the second normal form and contains only columns that are no transitively dependent on the primary key.

When you follow these rules, the tables of the model are in the third normal form, according to E. F. Codd, the inventor of relational databases. When tables are not in the third normal form, either redundant data exists in the model, or problems exist when you attempt to update the tables.

7.9 Technical Terms

Normal form : A state of a relation that results from applying simple rules regarding functional dependencies to that relation.

Transitive dependency: Let R be a relation and let a, b and c are the attributes of R then we say that R satisfies transitive dependency if there exists aàb and bàc and consequently aàc.

Functional Dependency: many-to-one relationship shared by columns of values in database tables. A functional dependency from column X to column Y is a constraint that requires two rows to have the same value for the Y column if they have the same value for the X column.

Transitive dependency: A functional dependency between two nonkey attributes.

Alias: An alternative name used for an attribute.

Non-Loss Decomposition: without losing of data, dividing the relation into multiple number of relations called Non loss Decomposition.

7.10 Self Assessment Questions

1. What does Data Normalization mean? What are the rules for Normalization?
2. Explain the First normal Form [1NF] with an example?
3. Explain the concept of functional Dependency?
4. What is Decomposition? Write about Loss-less Decomposition?
5. How do you represent a 1:M unary relationship in a relation data model ?
6. How do you an represent a M:N ternary relationship in a relation data model ?

7.11 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-8

PHYSICAL DATABASE DESIGN AND PERFORMANCE-I

8.0 Objective

To be able to

- ☞ Describe the physical database design process, it objectives, and deliverables.
- ☞ Know field, data type, physical record, page, blocking factor
- ☞ Coding and compression techniques
- ☞ Controlling data integrity, denormalization
- ☞ Choose storage formats for attributes from a logical data model
- ☞ Select an appropriate file organization by balancing various important design factors.
- ☞ Describe three important types of file organization

Structure

8.1 Introduction

The purpose of physical database design is to translate the logical description of data into technical specifications for storing and retrieving data. Here the goal is to create a design for storing data that will provide adequate performance and ensure database integrity, security and recoverability.

Physical File: A named portion of secondary memory allocated for the purpose of storing physical records. Some computer operating systems allow a physical file to be split into separate pieces called extents.

Table Space: A table space is a named set of disk storage elements in which data from one or more database tables may be stored. Oracle includes several table spaces one for system data (data dictionary), one for temporary workspace, one for database recovery and several to hold user data.

All files organized by using two basic constructs called sequential storage and pointers. With sequential storage one field or record is stored right after field or record. A pointer is a field of data that can be used to locate a related field or record of data.

8.2. Physical Database Design Process

Many Database design decisions are implicit when you choose database management technology. But for the few decisions that are not implicit, you must deal.

The information or input needed for physical database design includes,

1. Normalized relations with volume estimates
2. Definition of each attribute
3. Description of where and when data are used
4. Expectations for response time, security Backup and recovery and integrity
5. Description of technologies used

The critical decisions you explicitly take, will effect the integrity and performance of the application. They are

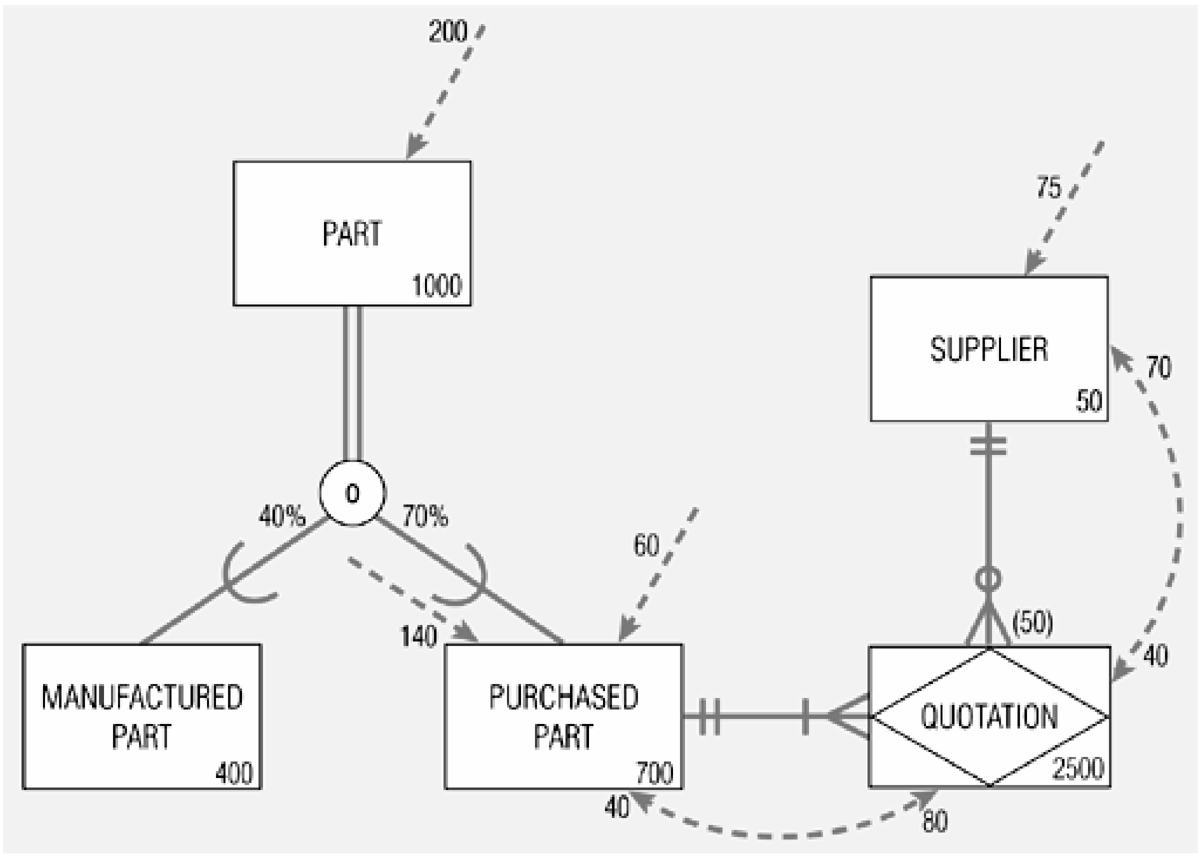
1. Choosing the storage format (or) Data type for each attribute to minimize storage space and to maximize data integrity.

2. Grouping attributes the logical data model into physical records.
3. Arranging similarly structured records in secondary memory so that individual and group of records can be stored retrieved and update rapidly.
4. Selecting structures (i.e. indexed and database architectures) for storing to make retrieving related data more efficient.
5. Preparing strategies for handling quires against the database.

8.2.1 Data volume and Usage Analysis

Either the final step you need to take in logical database design or the first step you need to take in physical database design is to estimate the size and usage patterns of the database called Data volume and usage. An easy way to show these statistics about data volume and usage is by adding notations to the EER diagrams.

Figure 8.1 Composite Usage map Furniture



8.3. Designing Fields

A field is the smallest unit of application data recognized by the system software. The basic decision you must make in specifying each field is

1. Type of the data used to represent values of that field
2. Data integrity that controls missing values.

8.3.1 Choosing Data Type

A data type is a detail-coding scheme recognized by the system software like DBMS. The DBMS you use provides several data types. The following are the data types provided by Oracle 8i,

CHAR – fixed-length character

VARCHAR2 – variable-length character (memo)

LONG – large number

NUMBER – positive/negative number

DATE – actual date

BLOB – binary large object (good for graphics, sound clips, etc.)

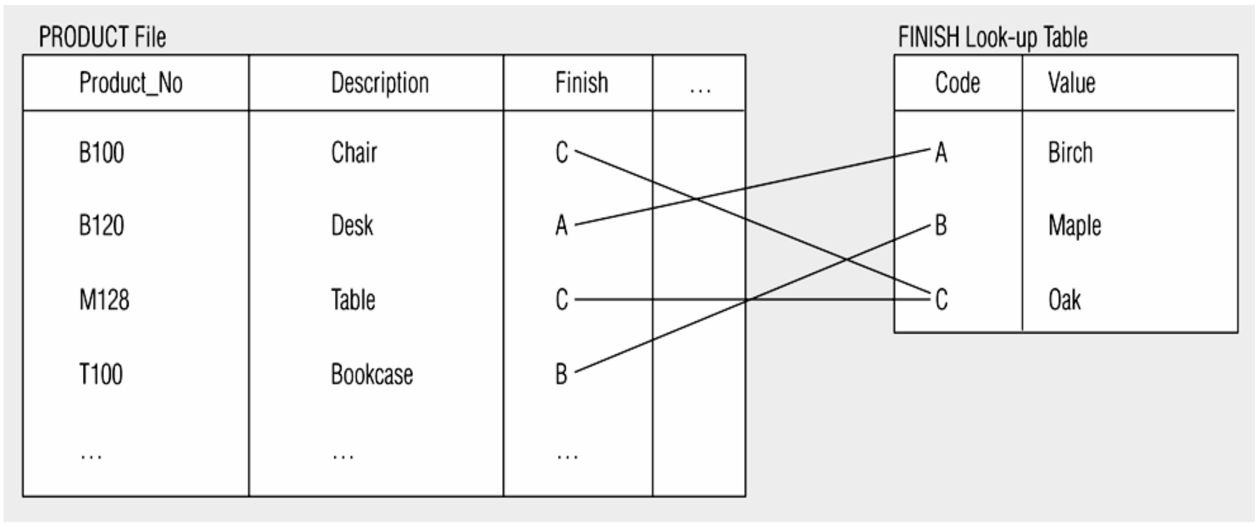
Selecting a data type involved objects,

1. Minimize storage space
2. Represent all possible values
3. Improve data integrity
4. Support all data manipulations

Coding and Compression Techniques

A field with a limited number of possible values can be translated into a code that requires less space. But this is beneficial if the finish field is infrequently used or the number of distinct values very large, otherwise it increases the cost will be overhead.

8.2 Example code-look-up table



PRODUCT File

Product_No	Description	Finish	…
B100	Chair	C	
B120	Desk	A	
M128	Table	C	
T100	Bookcase	B	
…	…	…	

FINISH Look-up Table

Code	Value
A	Birch
B	Maple
C	Oak

8.3.2 Controlling Data Integrity

The integrity controls that a DBMS may support are,

1. **Default value**: A default value is the value a field will assume when a user do not enters an explicit value for an instance of that field.
2. **Range Control**: It limits the set of possible values a field may assume. It controls entry of the wrong data by the operators.
3. **Null value control**: A null value is an empty value. Some times a field value may not be available at the time of data entry. In that case the field value can be null and can be entered later when data is available.
4. **Referential Integrity**: Referential integrity on a field is a form of range control in which the value of that field must exist as the value in some field in another row of the same or different table.

Handling Missing Data

When a field may be null simply entering no value may be sufficient. But how should these field involved in calculations? There are two options for handling missing data are,

a. Using default value

b. Not permitting missing values.

Other possible methods for handling missing data are,

a. Substitute an estimate of missing value.

b. Track missing data and create special report

c. Perform sensitivity testing, so that missing data are ignored unless knowing a value might significantly change result.

8.4 Designing Physical Records and Demoralization

Physical Record

A group of fields in adjacent memory locations retrieved and written together as a unit by DBMS is a Physical Record. The design of physical record is achieve two goals,

1. Efficient use of secondary storage
2. Data processing speed

The efficient use of secondary storage is influenced by both the size of physical record and structure of secondary storage. The Computer Operating system read data from hard disk in units called pages.

Page

A Page is the amount of data read or written by an operating system in one Secondary memory input or output operation. Depending on computer system, a physical record may or may not span in two or more pages. Thus, if page length is not an integer multiple of physical record length, wasted space may occur at the end of a page.

Blocking Factor

The no. of physical records per page is called Blocking factor.

Generally Normalized relations solve data maintenance anomalies and minimize redundancies but may not yield efficient data processing. For example, a query may result joining many normalized tables is complex and time taking process; one solution for this is Demoralization.

8.4.1 Demoralization

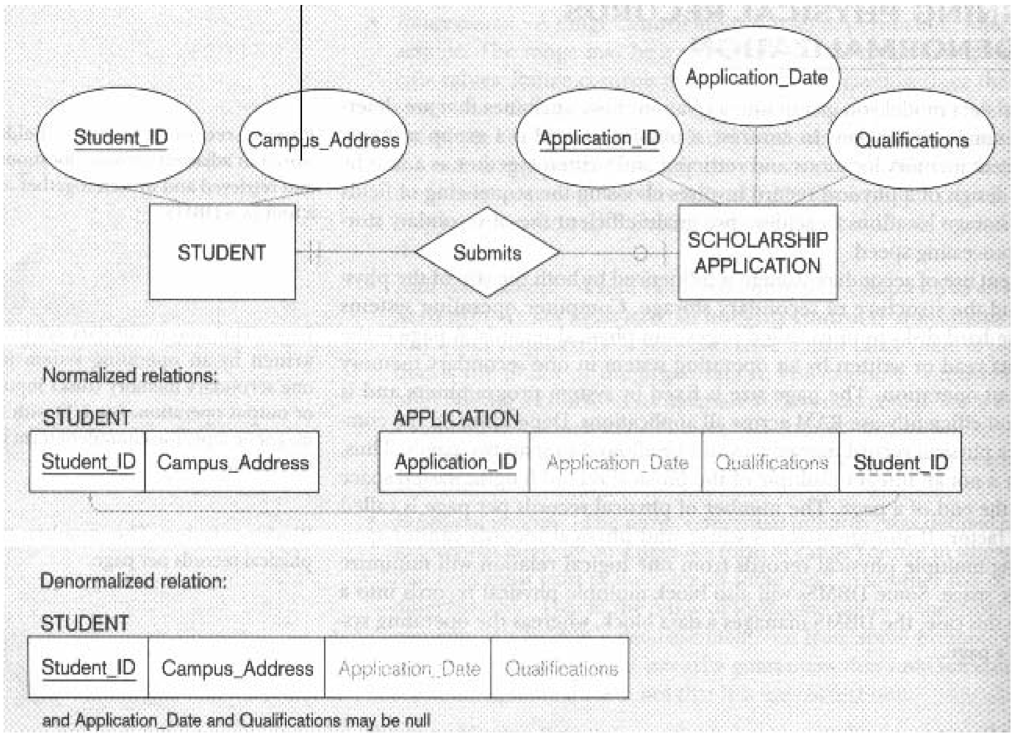
The process of transforming normalized relations into un-normalized physical record specifications. The process may yield data processing speed but can increase the chance of errors and inconsistencies.

There are several common demoralization opportunities,

1. Two entities with a one-to-one relationship
2. A many-to-many relationship (associative entity) with nonkey attributes
3. Reference data

Two entities with a **One-to-one** relationship and if any one is optional can be combined into one relation.

Figure 8.3 A possible demoralization situation: two entities with one-to-one relationship



Normalized relations:

STUDENT

Student_ID	Campus_Address

APPLICATION

Application_ID	Application_Date	Qualifications	Student_ID

Denormalized relation:

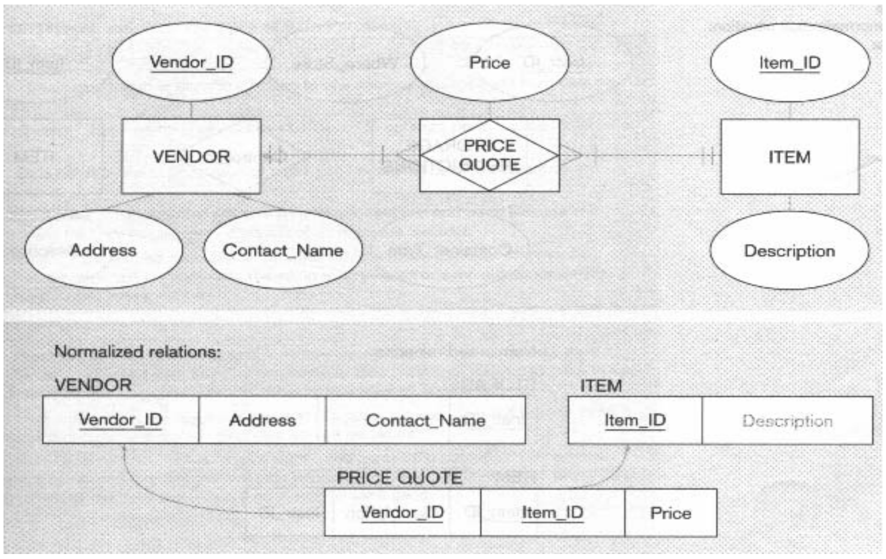
STUDENT

Student_ID	Campus_Address	Application_Date	Qualifications

and Application_Date and Qualifications may be null

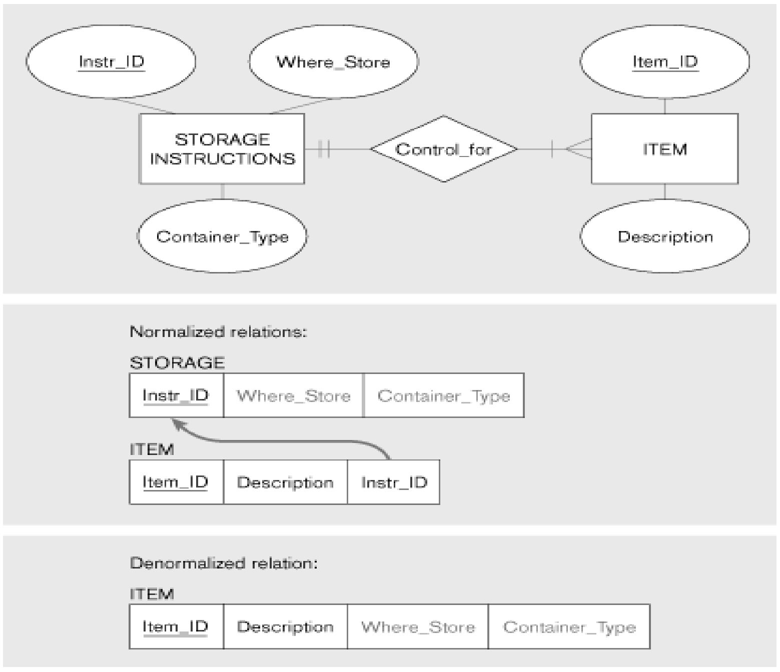
A **many-to-many** relationship with attributes is generally called associative entity. If that associative entity have no key attributes, then those non-key attributes can be combined with an entity on either side of relation.

Figure 8.3 A possible demoralization situation: a many-to-many Relationship with nonkey attributes



If **reference data** exist in an entity on the one-side of a one-to-many relationship and this entity does participates in no other database relationships, then those two entities can be combined.

Figure 8.5 A possible Demoralization situation : reference data



Partitions

An alternative to Demoralization for increasing data processing speed is partitions. The process of decomposing table into multiple tables called partitioning. Three types of partitions are,

1. **Horizontal partitioning:** Distributing the rows of a table into several separate files For example a customer relation could be broken into four regional customer files based on the value of a field region. The Oracle 8i DBMS has three horizontal partitioning methods,
   a) Key Range Portioning: Each partition is defined by a range of values for one or more columns of normalized table.
   b) Hash Partitioning: Use Hash algorithm make the Data are spread across partitions independent of any partition key value.
   c) Compost Partitioning: Combination of aspects of both key range and hash partitioning.
2. **Vertical Partitioning:** Distributions columns of a relation into separate physical records, repeating the primary key in each of the partitions.
3. **Combination of horizontal and vertical partitioning:** Each data file by horizontal partition can again partitioned into several vertical partitioned data files.

Advantages of Partitioning

1. Efficiency: Data used together are stored close to one another and separate from data not used together.
2. Local Optimization: Each partition of data can be stored to optimize performance for its own use.
3. Security: Data not relevant to one group of users can be segregated from data they are allowed to use.
4. Recovery and uptime: Smaller files will bake less time to recover, and other files are still accessible if one file is damaged, so the effects of damage are isolated.
5. Load Balancing: Files will take less time to recover, and other files are still accessible if on file is damaged, so the effects of damage are isolated.

Disadvantages of Partitioning:

1. Inconsistent Access speed: Different partitions may yield different access speeds, thus confusing user.
2. Complexity: Partitioning is usually not transparent to programmers, who will have to write more complex programs when combining data across partitions.
3. Extra space and update time : Data may be duplicated across the partitions, taking extra storage compared to storing all the data in normalized file. Updates which affect data in multiple partitions can take more time than if one file were used.

8.5 Designing physical Files

Physical File: A named portion of secondary memory allocated for the purpose of storing physical records.

Tablespace: A named set of disk storage elements in which physical files for database tables may be stored

Extent : A contiguous of disk storage space.

8.5.1 Pointer

A field of data that can be used to locate a related field or record of data

8.5.2 File Organizations

A file organization is a technique for physically arranging records of a file on secondary storage devices. In choosing a file organization for a particular file in a database, you should consider important factors,

1. Fast data retrieval
2. High throughput for processing data
3. Efficient use of storage space
4. Protection from failures or data loss
5. Minimizing need for reorganization
6. Accommodation growth
7. Security from unauthorized use

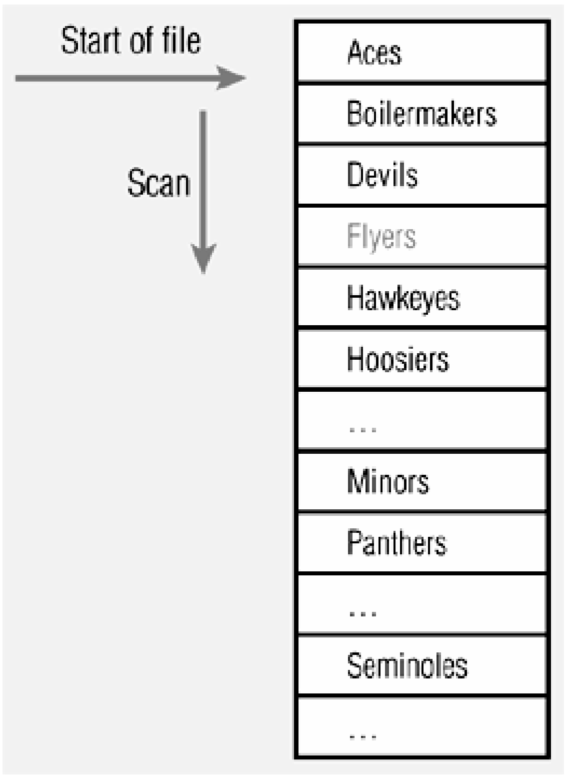
The following are the basic file organizations

1. Sequential File Organization
2. Indexed File Organization
3. Hashed File Organization

Sequential File Organization

In sequential file organization the records in the file are stored in sequence according to a primary key value. To locate a particular record a program must normally scan the file from the beginning until the desired must record is located. Because of their inflexibility, sequential files are not used. But may be used for files that backup data.

Figure 8.7 Comparison of file organizations a) Sequential



Records of the file are stored in sequence by the primary key field values.

Indexed File Organization

In indexed File Organizations, the records are stored either sequentially or non-sequentially with an index allow DBMS to locate individual records.

Index: Index is a table that is used to determine the location of rows in a file that satisfy some condition. Each index entry matches a key value with one or more records, if one record called as primary index, if more than one is recorded called as secondary.

There are three types of Indexed organizations:

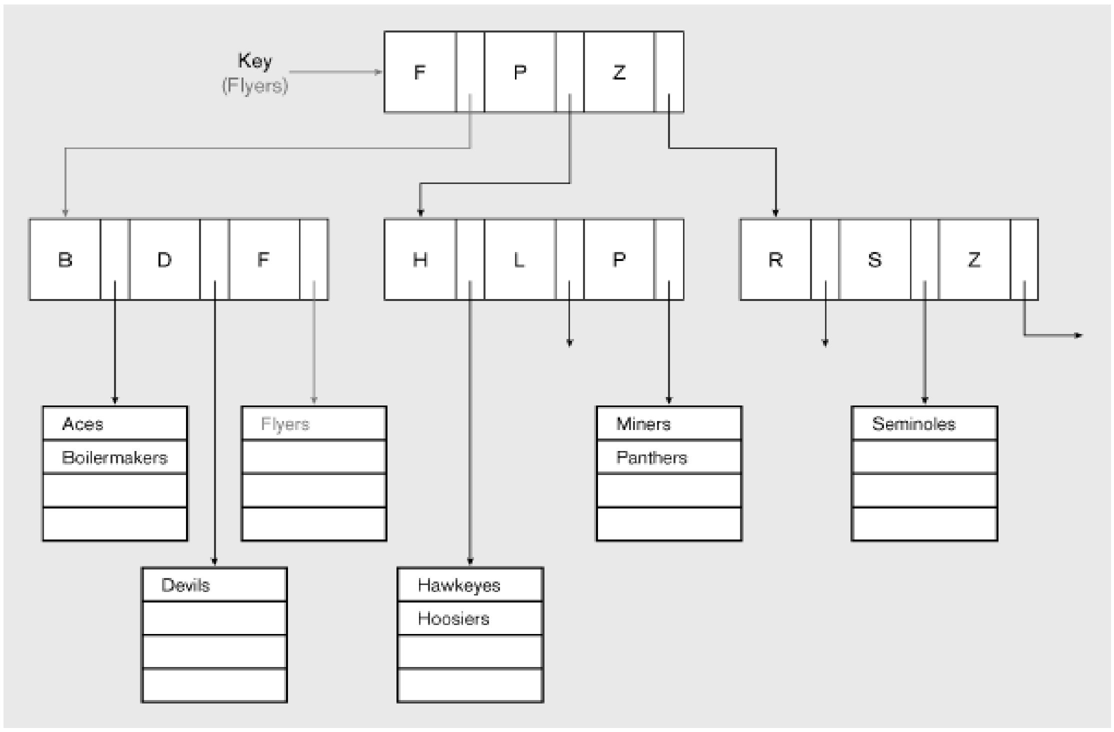
☞ B-Tree

☞ Bitmap

☞ Join Index

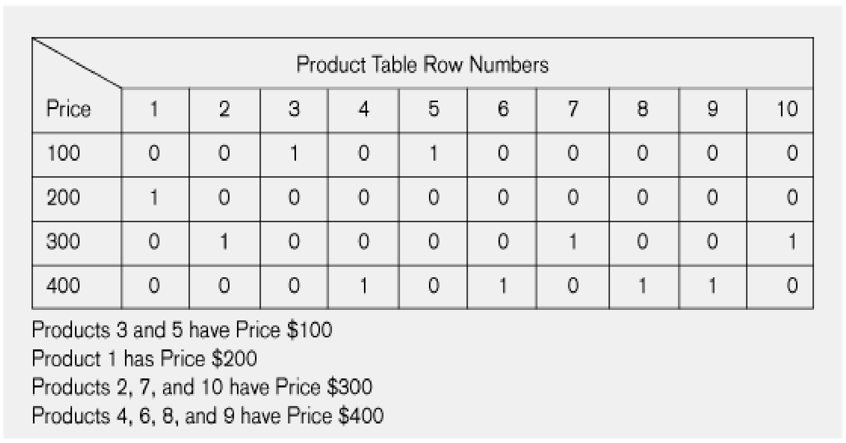
B-Tree Index: The example illustrates that indexed can be built on top of indexes, i.e. creating a hierarchical set of indexes. This may be desirable since an index itself is a file and if it is very large. Each index has a key value and a pointer to another index or to a data record.

(b) Indexed



Bitmap-index: A table of bits in which each row represents the distinct values of a key and each column is a bit which indicates that the record for that bit column position has the associated field value. A bitmap is ideal for attributes that have even few possible values. A bitmap also often requires less storage space than a tree index.

Figure 8.8 Bitmap index on Product Price attribute

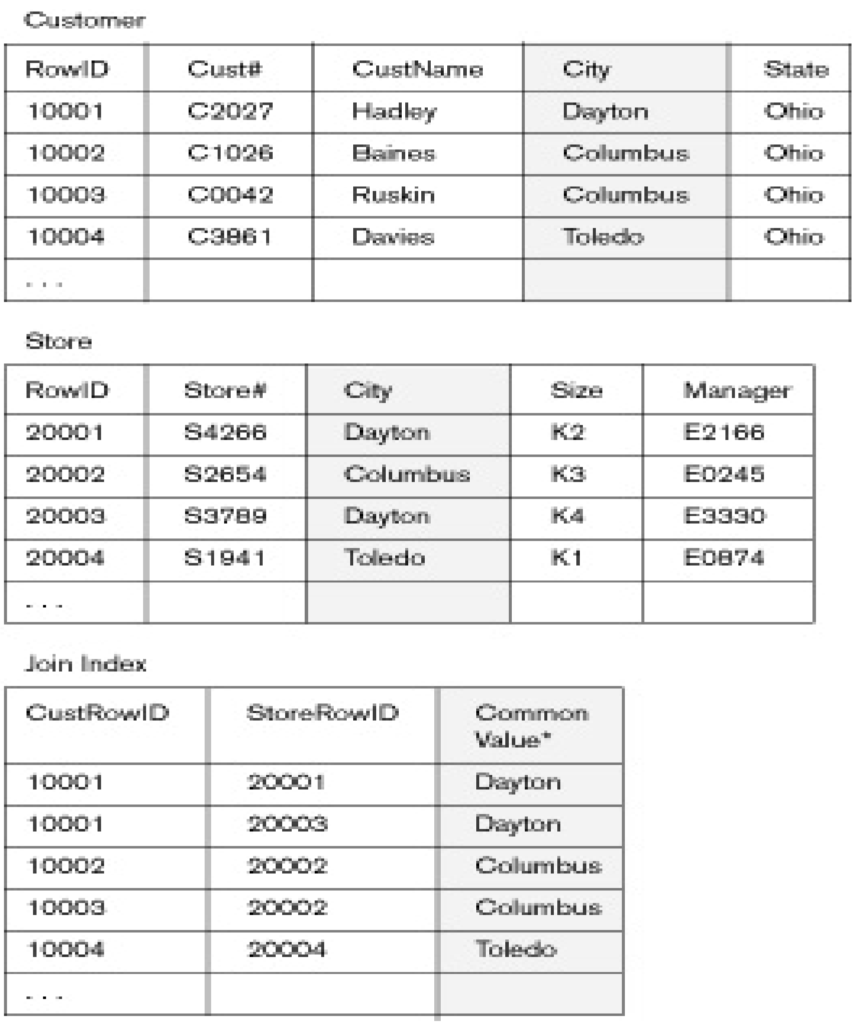


	Product Table Row Numbers									
Price	1	2	3	4	5	6	7	8	9	10
100	0	0	1	0	1	0	0	0	0	0
200	1	0	0	0	0	0	0	0	0	0
300	0	1	0	0	0	0	1	0	0	1
400	0	0	0	1	0	1	0	1	1	0

Products 3 and 5 have Price $100
Product 1 has Price $200
Products 2, 7, and 10 have Price $300
Products 4, 6, 8, and 9 have Price $400

Join index: An index on columns from two or more tables that come from the same domain values.

Figure 8.9 Join indexes (a) Join index for common nonkey columns



Customer

RowID	Cust#	CustName	City	State
10001	C2027	Hadley	Dayton	Ohio
10002	C1026	Baines	Columbus	Ohio
10003	C0042	Ruskin	Columbus	Ohio
10004	C3861	Davies	Toledo	Ohio
. . .				

Store

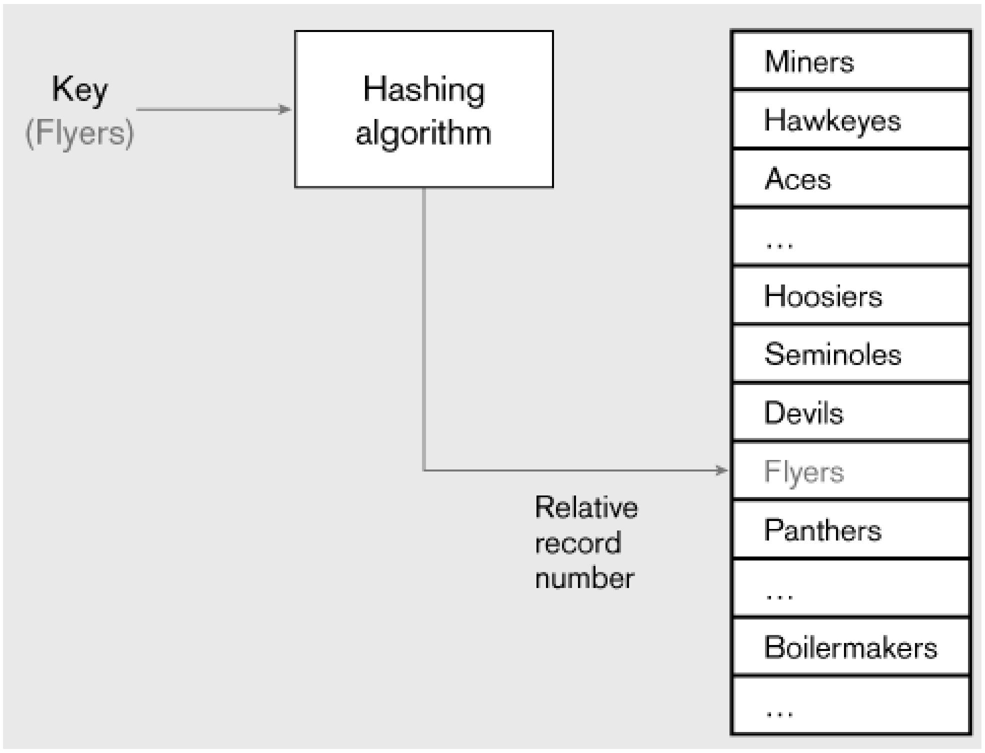
RowID	Store#	City	Size	Manager
20001	S4266	Dayton	K2	E2166
20002	S2654	Columbus	K3	E0245
20003	S3789	Dayton	K4	E3330
20004	S1941	Toledo	K1	E0874
. . .				

Join Index

CustRowID	StoreRowID	Common Value*
10001	20001	Dayton
10001	20003	Dayton
10002	20002	Columbus
10003	20002	Columbus
10004	20004	Toledo
. . .		

Hashed File Organization: In a hashed file organization the address of each record is determined using a hash algorithm. A hashing algorithm is a routine that converts a primary key value into a record address. Hashing and indexing can be combined into what is called Hash Index table i.e. a file organization that uses hashing to map a key into a location in an index where there is a pointer to the actual data record.

Figure 8.7 (continued) C) Hashed



8.5.3 Summary of File Organizations

The three families of file organizations cover most of the file organizations you will have at your disposal as you design physical files and databases. You are unlikely to be able to use these with a database management system.

You should review this table and study Figure 8.7 to see why each organization feature is true.

8.5.4 Clustering Files

Some database management systems allow adjacent secondary memory space to contain rows from several tables i.e. A physical file does not contain records with identical structures.

A cluster is defined by the tables and the column or columns by which the tables are joined. A cluster is a group of tables that share the same data blocks because they share common columns and after used together.

Ex: Create cluster dept_emp (clusterkey Number (8));

Create table Emp (empno, Number (8), ename varchar2 (20), doj date, deptno Number (4))

```
      Cluster dept_emp (deptno);
  Create table dept (deptno Number (4), dname varchar2 (15), desc varchar2 (30))
        Cluster dept_emp (deptno);
```

8.5.5 Designing Controls for Files

One additional aspect of a database file about which you may have design options are the types of controls you can use to protect the file from destruction or contamination or to reconstruct the file if it is damaged. Because a database file is stored in a proprietary format by the DBMS, there is a basic level of access control. You may require additional security controls on fields, files, or databases.

Briefly, files will be damaged, so the key is the ability to rapidly restore a damaged file. Backup procedures provide a copy of a file and of the transactions that have changed the file. When a file is damaged, the file copy or current file along with the log of transactions are used to recover the file to an uncontaminated state. In terms of security; the most effective method is to encrypt the contents of the file so that only programs with access to the decryption routine will be able to see the file contents

8.6 Summary

During physical database design, you the designer translate the logical description of data into the technical specifications for storing and retrieving data. The goal is to create a design for strong data that will provide adequate performance and insure database integrity, security, and recoverability. In physical database design you consider normalized relations and data volume estimates, data definitions, data processing requirements and their frequencies, user expectations, and database technology characteristics to establish field specifications, record design, file organizations, and a database architecture.

A field is the smallest unit of application data, corresponding to an attribute in the logical data model. You must determine the data type, integrity controls, and how to handle missing values for each filed, among other factors. A data type is a detailed coding scheme for representing organizational data. Data may be coded or compressed to reduce storage space.

Field integrity control includes specifying a default value, range of permissible values, null value permission, and referential integrity.

Demoralization is done to place is one physical record those attributes frequently needed together in an i/O operation. Demoralization includes horizontal partitioning, which breaks a relation into multiple record specifications by placing different rows into different records based upon common column values. Demoralization also includes vertical partitioning, which distributes the columns of a relation into separate files, repeating the primary key in each of the files.

A physical file is a named portion of secondary memory allocated for the purpose of storing physical records. Data within a physical file are organized through a combination of sequential storage and pointers. A pointers is a filed of data that can be used to locate a related field or record of data.

A file organization arranges the records of a file on a secondary storage device. The three categories of file organizations are. (1)Sequential, which stores records in sequence according to a primary key value (2) indexed, in which records are stored sequentially or non sequentially and an index is used to keep track of where the records are stored;(3) hashed, in which the address of each records is determined using an algorithm that converts a primary key value into a record address. Physical records of several types can be clustered together into one physical file in order to place records frequently used together close to one another in secondary memory.

8.7 Technical Terms

Field: The smallest unit of named application data recognized by system software.

Data Type: A detailed coding scheme recognized by system software, such as a DMBS, for representing organizational data.

Horizontal Partitioning: Distributing the rows of a table into several separate files

Vertical Partitioning: Distributing the columns of a table into several separate physical records.

File Organization: A technique for physically arranging the records of a file on secondary storage devices.

Secondary Key: One filed or a combination of fields for which more than one record may have the same combination of values. Also called a nonunique key.

8.8 Self Assessment Questions

1. Define each of the following terms
   (a) File Organization
   (b) Sequential file Organization
   (c) Indexed file organization
   (d) Hashing file organization
   (e) Demoralization
2. Contrast the following terms
   (a) Horizontal partitioning; vertical partitioning
   (b) Physical file; table space
3. What are the major inputs of physical database design?
4. What are the key decisions in physical database design?
5. What information is shown on a composite usage map?
6. Describe three ways to handle missing field values.
7. List three common situations that suggest demoralizing relations to form physical records.
8. What are the advantages and disadvantages of horizontal and vertical partitioning?

8.9 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- **G.Venugopal Rao.**, M.Sc, M.Phil.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-9

PHYSICAL DATABASE DESIGN AND PERFORMANCE-II

9.0 Objective

To be able to

☞ Describe the purpose of indexes and the important considerations in selecting attributes to be indexed.
☞ Creating and observing Primary key index
☞ Creating a Secondary key indexes
☞ Different types of databases, different database architectures
☞ Query performance, data blocks, I/O controllers

Structure

9.1 Introduction

We will learn about different uses of indexes, which are important in speeding up the retrieval of data, and will learn the major differences between different architectures of databases.

We must carefully perform physical database design, since the decisions made during this stage have a major impact on data accessibility, response times, security, user friendliness, and similarly important information system factors. Database administration plays a major role in physical database design, so we will return to some advanced design issues in this lesson.

9.2 Using and Selecting Indexes

Most database manipulations require locating a row that satisfy some condition. For example, we may want to retrieve all customers in a given zip code or all students with a particular major. Scanning every row in a table looking for the desired rows may be unacceptably slow, particularly when tables are large, can greatly speed up this process, and defining indexes is an important part of physical database design.

As described in the section on indexes above, indexes on a file can be created for either a primary or a secondary key or both. It is typical that an index would be created for the primary key of each table. The index is itself a table with two columns: the key and the address of the record or records that contain that key value. For a primary key, there will be only one entry in the index for each key value.

9.2.1Creating a Unique Key Index

The customer table defined in the section on clustering has a primary key of Customer_ID. A unique key index would be created on this field using the following SQL command.

```
CREATE UNIQUE INDEX CUSTONDEX ON CUSTOMER(CUSTOMER_ID);
```

In this command, CUSTINDEX is the name of the index file created to store the index entries. The ON clause specifies which table is being indexed and the column that form the index key. When this commands is executed, any existing records in the Customer table would be indexed. If there are duplicate values of Customer_ID, the CREATE INDEX command will fail. Once the index is created, the DBMS will reject any insertion or update of data in the CUSTOMER table that would violate the uniqueness constraint on Customer_IDS.

When a composite unique key exits, you simply list all the elements of the unique key in the On clasue. For example, a table of line items on a customer order might have a composite unique key of ORDER_ID and Product_ID.The SQL command to create this index for the Orde_Line table would be as follows.

```
CREATE UNIQUE INDEX LINEINDEX ON ORDER_LINE(ORDER_ID,PRODUCT_ID);
```

9.2.2 Creating Secondary key Index

Database users often want to retrieve rows of a relation based on values for various attributes other than the primary key. For Example , in a product table, users might want to retrieve records that satisfy any combination of the following conditions.

All table products(Description='Table')

All oak furniture(Finish='Oak')

All dining room furniture(Room='DR')

All furniture priced below $500(Price<500)

To speed up such retrievals, we can define an index on each attributes that we use to qualify a retrieval, for example, we could create a nonunique index on the Description field of the Product table with the following SQL commands.

```
CREATE INDEX DESCINDX ON PRODUCT( DESCRITPION);
```

Notice that the term UNIQUE should not be used with secondary key attributes, since each value of the attribute may be repeated. As with unique keys, a secondary key index can be created on a combination of attributes.

To create a bitmap index, you follow a similar command structure. If we wanted a bitmap index for the Description field , the command would be:

```
CREATE BITMAP INDEX DESCBITINDX PRODUCT(DESCRIPTION);
```

9.2.3. When to Use Indexes

During physical database design, you must choose which to sue to create indexes. There is a trade-off between improved performance for retrievals through the use of indexes, and degraded performance for inserting, deleting, and updating the records in a file. Thus, indexes should be used generously for databases intended primarily to support data retrievals, such as for decision support and data warehouse applications. Indexes should be used judiciously for database that support transaction processing and other applications with heavy updating requirements, since the indexes impose additional overhead.

Following are some rules of thumps for choosing indexes for relational databases.

1. Indexes are most useful on large tables.
2. Specify a unique index for the primary key of each table.
3. Indexes are most useful for columns that frequency appear in WHERE clauses if SQL commands either to qualify the rows to select or for linking table. In the latter case, the index is on a foreign key in the Order-Line table that is used in joining tables.
4. Use an index for attributes referenced in ORDER BY and GROUP BY clauses. You do have to be careful, through, about these clauses. Be sure that the DBMS will, in fact, use indexes on attributes listed in these clauses.
5. Use an index when there is significant variety in the value of an attribute. Oracle suggests that an index is not useful when there are fewer 30 different values for an attribute, and an index is clearly useful when there are 100 or more different values for an attributes. Similarly, an index will be helpful only if the results of a query which uses that index do not exceed roughly 20 percent of the total number of records in the file.
6. Check your DBMS for the limit, if any, on the number of indexes allowable per table. Many systems permit no more than 19 indexes, and may limit the size of an index key value. If there is such a limit in your system, you will have to choose secondary keys that will most likely lead to improved performance.
7. Be careful indexing attributes that have null values. For many DBMSs, rows with a null value will not be referenced in the index. Such a search will have to be done by scanning the file.

Selecting indexes is arguably the most important physical database design decision, but it is not only way you can improve the performance of a database. Other ways address such issues as reducing the costs to relocate records, optimizing the use of extra or so-called free space in files, and optimizing query processing algorithms.

9.3 Designing Databases

Most modern information systems utilize database technologies, either database management systems or data warehouse systems, for data storage and retrieval. Each type of database technology allows different types of access paths. So, the process of choosing the appropriate type of DBMS or data warehousing technology is one of matching the needed access paths with the capabilities of the database technology.

9.3 .1 Choosing Database Architectures

There are different styles of database management and data warehousing systems, each characterized by the way data are defined and structured, called database architectures. Deciding which database architecture best suits your database is fundamental to database design. A particular database management or warehousing system supports one of five different architectures.

1. **Hierarchical Database Model**: In this model, files are arranged in a top-down structure that resembles a tree or chart. Data stored in this model is in one-to-many relationship. The top file is called root, the bottom files are called leaves and intermediate files have one parent file and one or several children files.

2. **Network database model**: In this model, each file may be associated with an arbitrary number of files. Typically this model support only one-to-many relationships, but some support many-to-many relationships. This model supports a wider variety of processing requirements than Hierarchical data model, but it is little complex.

3. **Relation database model:** This defines simple tables for each relation and many-to-many relationships. Reference keys link the tables together, represents the relationship between entities. This is famous and has advantages than above models.

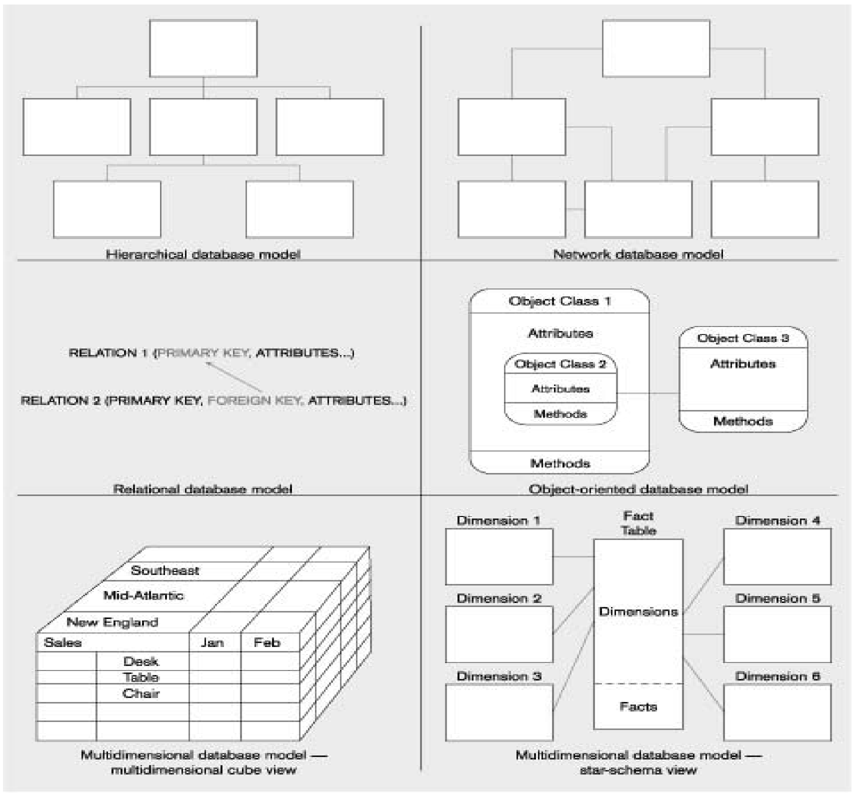
4. **Object-oriented database model**: In this model, attributes and methods that operate on those attributes are encapsulated in structures called object classes. New object classes are defined from more general object classes. A major advantage of this model is for complex data types like graphics, video and sound.

5. **Multidimensional database model**: This data model is used in data warehousing applications. There are two views for this model called multidimensional cube and star scheme. In multidimensional cube, each cell contains data relevant to the intersection of all of its dimension values.

   In **star schema**, a fact table is in center contains fact data surrounded by dimension tables contains description of each fact that is stored in fact table.

   Examples shown in the following picture,

Figure 9.12 Database Architecture


Hierarchical database model
Network database model
Object Class 1
Attributes
Object Class 2
Attributes
Methods
Methods
Object Class 3
Attributes
Methods
RELATION 1 (PRIMARY KEY, ATTRIBUTES...)
RELATION 2 (PRIMARY KEY, FOREIGN KEY, ATTRIBUTES...)
Relational database model
Object-oriented database model
Southeast
Mid-Atlantic
New England
Sales
Jan
Feb
Desk
Table
Chair
Multidimensional database model — multidimensional cube view
Dimension 1
Fact Table
Dimension 4
Dimension 2
Dimensions
Dimension 5
Dimension 3
Facts
Dimension 6
Multidimensional database model — star-schema view

9.4 Optimizing For Query Performance

The primary purpose today for physical database design is to optimize the performance of database processing. Database processing includes adding, deleting and modifying a database, as well as a variety of data retrieval activities. For database that have greater retrieval traffic than maintenance traffic need query optimization. Varieties of query optimizations are,

9.4.1 Parallel Query Processing

The most common approach is to replicate the query so that each copy works against a portion of the database (usually horizontal partition). The same query is run against each partition in parallel on separate processor and the intermediate results from each processor are combined to create the final query result.

9.4.2 Overriding Automatic Query optimization

With most relational databases you can learn the optimizer's plan for processing query before running query. The query optimizer chooses the best plan based on statistics about each table such as average row length and number of rows. Query writer can force the DBMS to do steps differently than the optimizer thinks is the best plan.

9.4.3 Picking data Block size

Data are transferred between RAM and disk memory in blocks or pages. The size of the page can affect the performance of queries. Too small page size may result in many I/O operations per table. Too large a page size may result in extra data being transferred with wasted time.

9.4.4 Balancing I/O across Disk Controllers

A Disk controller manages the I/O operations for the disk drives attached to that controller. More controllers are better than a few, but each controller increase cost. The benefits of multiple controller came when they can all are kept busy.

9.4.5 Guidelines for Better Query Design

1. Create and use reasonable member of indexes.
2. Use compatible data types for fields and literals in queries.
3. Write simple queries.
4. Break complex queries into multiple, simple parts.
5. Reduce use of nested query inside another query.
6. Combine a table with itself.
7. Create temporary tables for group of queries.
8. Combine update operations.
9. Retrieve only the data you need.
10. Don't have DBMS sort without an index.
11. Learn and understand query optimization.

9.5 Summary

The indexed file organization is one of the most popular in use today. An index may be based on a unique key or a secondary (non-unique) key, which allows more than one record to be associated with the same key value. The new form of an index, a bitmap index, creates a table of bits in which an on bit means that the related record has the related key value. A join index indicated rows from tow or more tables that have common values for related fields. A hash index table makes the placement of data independent of the hashing algorithm and permits the same data to be accessed via several hashing functions on different fields. Indexes are important in speeding up data retrieval, especially when multiple conditions are used for selecting, sorting, or relating data. Indexes are useful in a wide variety of situations, including for large tables, for columns that are frequently used to qualify the data to be retrieved, when a field has a large number of distinct values, and when data processing is dominated by data retrieval rather than data maintenance.

Database architectures in use today are hierarchical, network, relational, object-oriented, and multidimensional. Hierarchical and network architectures primarily appear in legacy applications, whereas relational, object-oriented and multi-dimensional architectures are used for new systems development.

The introduction of multiprocessor database serves has made possible new capabilities in database management systems. One major new feature is the ability to break a query apart and to process the query in parallel against segments of a table. Such parallel query processing can greatly improve the speed of query processing. Also database programmers can improve database-processing performance by providing the DBMS with hints about the sequence in which to perform table operations. These hints override the cost-based optimizer of the DBMS. Both the DBMS and programmers can look at statistics about the database to determine how to process a query.

Having developed complete physical data specifications, you are now ready to begin implementing the database with database technology. Implementation means defining the database and programming client and server routines to handle the queries, reports, and transactions against the database.

9.6 Self Assessment Questions

1. Under what circumstances is a bitmap index desirable ?
2. Want are the benefits of a hash index table?
3. What is the purpose of clustering of data in a file?
4. State seven rules of thumb for choosing indexes.
5. Contrast the two ways of viewing multidimensional databases.
6. How can use of the Explain command help in writing a more efficient query?
7. Explain four options for optimizing query performance.
8. Which level of RAID of best? Why?

9.7 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- **G.Venugopal Rao,** M.Sc, M.Phil.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-10

SQL - 1

10.0 Objective

To be able to

- ☞ Define the following terms: relational DBMS, catalog, schema, data definition language, data manipulation language, data control language, base table, dynamic view, materialized view, referential integrity.
- ☞ Interpret the history and role of SQL in database development.
- ☞ Define a database using the SQL data definition language.
- ☞ Write single table queries using SQL commands.
- ☞ Establish referential integrity using SQL.

Structure of the Lesson

10.1 Introduction

Oracle is a Relational Database Management System (RDBMS). Oracle being RDBMS, stores data in tables called relations. These relations are two-dimensional representation of data, where rows called tuples represent records and columns called attributes represent pieces of information contained in the record.

Oracle provides a rich set of tools to allow design and maintenance of the database. Major tools are,

RDBMS Kernel	: Database Engine
SQL	: Structured Query Language
SQL * PLUS	: Addition to SQL
PL / SQL	: Procedural Language SQL, allows Procedural processing of SQL statements
SQL * DBA	: Database Administrator's tool set
DEVELOPER 2000	: ORACLE'S GUI tool for Forms.

10.2 The SQL Environment

An SQL environment, It includes an instance of a relational DBMS along with the actual databases which are accessible by the DBMS. The environment supports multiple databases, such as a production (live) and a development (for debugging new programs) database. Each database is contained in a **catalog** which describes all the objects in the database (for all schemas). Each database has a named schema(s) associated with a catalog.

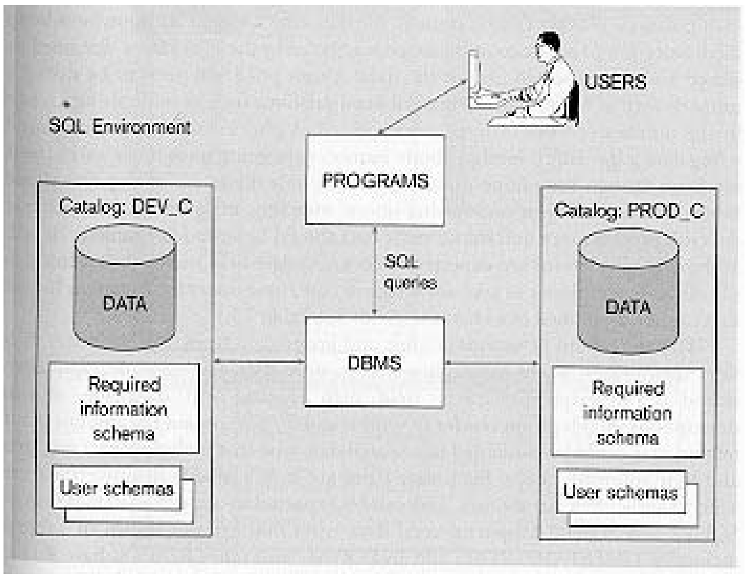
10.3 Defining a Database in SQL

The **schema** is a collection of related objects created by the user, such as tables, views, domains, constraints, triggers, etc. In other words, it is everything you create as it relates to a particular database.

The diagram below shows SQL environments for two catalogs — one used for development purposes, and the other the actual production level DBMS (live version with real data and all monitoring and security protection)

Note that information in the DBMS is created and maintained using SQL commands. The use of simple SQL commands will be used to carry out complex operations on the database. We will study how this is done over the next two Lecture Sets.

Figure : A simplified schematic of a typical SQL environment, as described by the SQL-92 standard



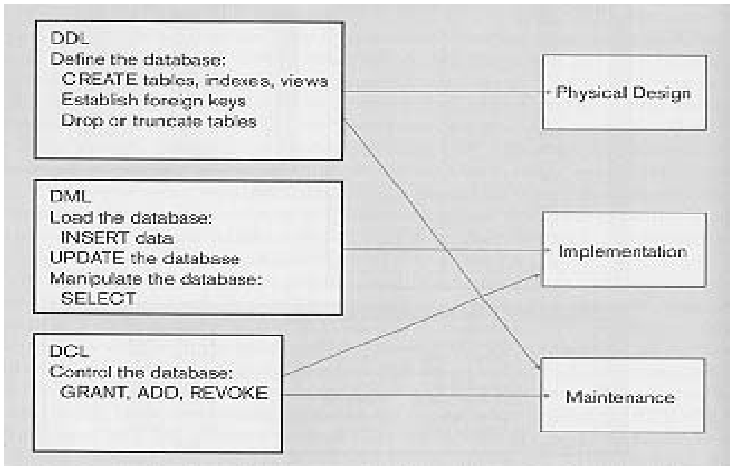
10.3.1 Generating SQL Database Definitions

SQL commands are classified into three types:

Data definition language (DDL) commands: Are used to create, alter and drop tables and views.

Data manipulation language (DML) commands**:** Are used for inserting, updating, modifying and querying data in the database. DML commands may be issued interactively or imbedded within programs written in 3GL languages such as C or COBOL.

Data control language (DCL) commands: Are used by the DBA (database administrator) to control the database such as grant or revoke privileges for accessing the database.



Generating SQL Database Definitions

Three SQL DDL CREATE commands are included in Entry-Level SQL-92.

CREATE SCHEMA used to define that portion of the database that a particular user owns. Schemas are dependent on a catalog, and contain schema objects including tables, views, domains and constraints etc

CREATE TABLE Defines a new table and its columns. Tables are dependent on the schema. Table can be a base table or a derived table. Tables are dependent on schema and are created by executing an SQL query that creastes a TABLE (rather than a query)

CREATE VIEW Defines a logical table from one or more tables or views.

DROP TABLE will destroy a table including its name, definitions and content. The DROP command pretty much acts as the inverse of opposite of CREATE.

DROP SCHEMA and DROP VIEW - Used to destroy schemas and views.

ALTER TABLE may be used to change the definition of an existing table by adding, dropping, or changing a column or dropping a constraint.

There are 5 other create commands in the base SQL-92 language standard. Your SQL could meet the Entry and Intermediate level SQL-92 standard without them.

CREATE CHARACTER SET - This is especially useful for languages other than English

CREATE COLLATION - Specifies the order that a character set assumes. Does not have to be ASCII. Each character in a character set is assigned a numeric value that determines its collating order.

CREATE TRANSLATION - Maps one character set to another in case of translations — for example, English to French, or ASCII to EBCDIC, etc..

CREATE ASSERTION - A schema object that can be used to create new CHECK constraints.

CREATE DOMAIN - A schema object that establishes a domain or set of valid values for an attribute. The domain can consist of a data type, default value, collation, or other constraint.

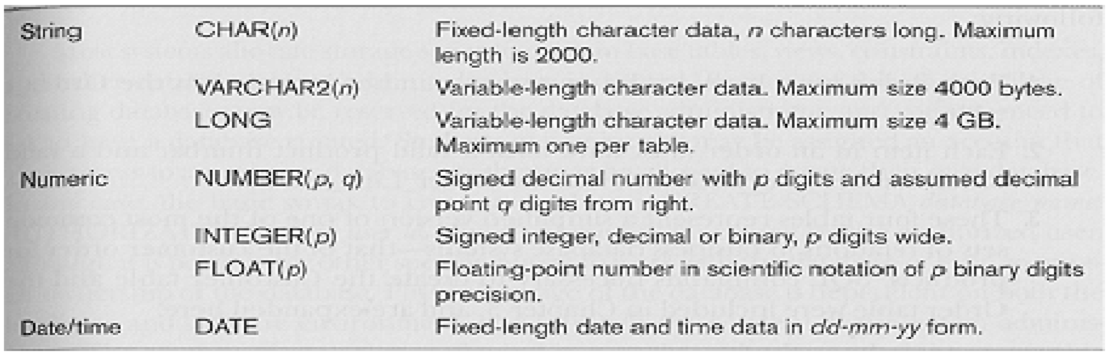
10.3.2 Creating Tables

Once the data model is designed and normalized, the columns needed for each table can defined using the SQL CREATE TABLE command.

Steps for creating a table:

- Be sure you have a normalized table to begin with.
- Provide a complete datatype specification for each attribute, especially length and precision (if needed)

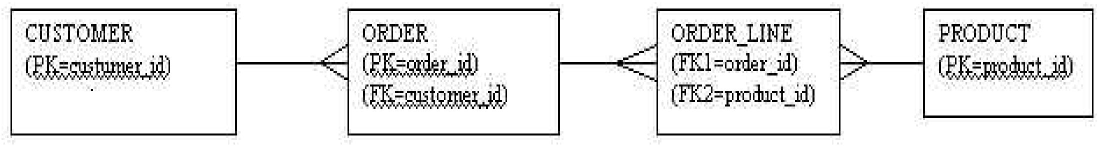
Data Types



String	CHAR(*n*)	Fixed-length character data, *n* characters long. Maximum length is 2000.
	VARCHAR2(*n*)	Variable-length character data. Maximum size 4000 bytes.
	LONG	Variable-length character data. Maximum size 4 GB. Maximum one per table.
Numeric	NUMBER(*p*, *q*)	Signed decimal number with *p* digits and assumed decimal point *q* digits from right.
	INTEGER(*p*)	Signed integer, decimal or binary, *p* digits wide.
	FLOAT(*p*)	Floating-point number in scientific notation of *p* binary digits precision.
Date/time	DATE	Fixed-length date and time data in *dd-mm-yy* form.

- Be sure you know which columns can be **NULL** and which cannot. Note that once it is established it is enforced for all data.
- Identify those columns that have to be **UNIQUE** — that is, no two table instances can have the same data value for such a column. Each such column is a candidate key (but only one can be chosen as PRIMARY KEY). UNIQUE and PRIMARY KEY are called **column constraints**.
- Identify all **foreign key**-primary key pairs (either as table is created or later, by adding a constraint. The “parent table should be created first so that the referencing or child table can reference an existing table when it is created. REFERENCES can be used to enforce **referential integrity**.
- Determine default values — those values to be used in a particular column when no value is entered for an instance of an entity.
- Identify any columns for which domain constraints need to be tightened or stronger than given. You can use CHECK to place a tighter column constraint. Note that “Spruce” or “White Maple” are not legal PRODUCT_FINISH values as shown in the table example a little further down. All legal values are included in the specified set of values.
- Create the table and any desired indexes using the CREATE TABLE and CREATE INDEX statements.

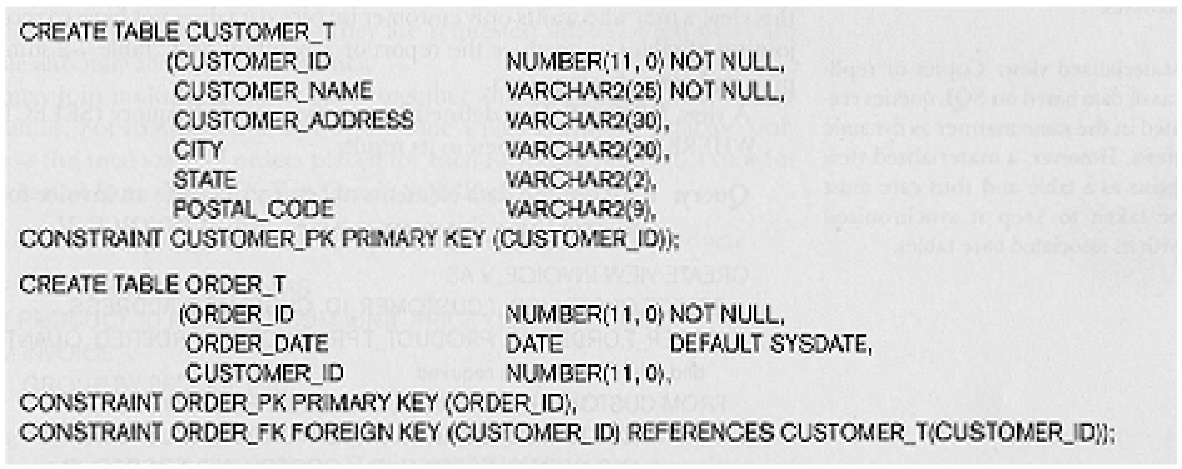
Example :



Let’s create tables for the ERD defining the relations between CUSTOMER, ORDER, ORDER_LINE and PRODUCT.

PK = Primary Key, FK = Foreign Key.

Figure : SQL database definition commands for pine Valley Furniture Company



```
CREATE TABLE CUSTOMER_T
            (CUSTOMER_ID                 NUMBER(11, 0) NOT NULL,
             CUSTOMER_NAME               VARCHAR2(25) NOT NULL,
             CUSTOMER_ADDRESS            VARCHAR2(30),
             CITY                        VARCHAR2(20),
             STATE                       VARCHAR2(2),
             POSTAL_CODE                 VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
             (ORDER_ID                   NUMBER(11, 0) NOT NULL,
              ORDER_DATE                 DATE          DEFAULT SYSDATE,
              CUSTOMER_ID                NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));
```

```
CREATE TABLE PRODUCT_T
            (PRODUCT_ID               INTEGER       NOT NULL,
             PRODUCT_DESCRIPTION      VARCHAR2(50),
             PRODUCT_FINISH           VARCHAR2(20)
                          CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                              'Red Oak', 'Natural Oak', 'Walnut')),
             STANDARD_PRICE           DECIMAL(6,2),
             PRODUCT_LINE_ID          INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
            (ORDER_ID                 NUMBER(11,0) NOT NULL,
             PRODUCT_ID               NUMBER(11,0) NOT NULL,
             ORDERED_QUANTITY         NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Note that names such as those with extensions _FK1 and _PK are useful aliases to give to column names. If you ever have occasion to read a constraint table, these names will help make the table more readable.

10.3.3 Using and Defining Views

A view is like a window through which data on tables can be viewed or changed. A view is derived from another table or view, which is referred as the base table. A view is stored as a SELECT statement only but has no data of its own. It manipulates data in the underlying base table.

Syntax: CREATE [OR REPLACE] VIEW view_name[col1,col2, . . .]

AS SELECT statement;

Ex 1: create view v_dept30 as select * from emp where deptno=30;

Ex 2: create view dept_summary(name,min_sal,max_sal,avg_sal)

as select dname,min(sal),max(sal),avg(sal) from emp e,dept d

where e.deptno=d.deptno group by dname;

View text can be viewed using a data dictionary view USER_VIEWS.

Views can be deleted using DROP command.

Syntax, DROP VIEW view_name;

10.3.4 Creating Data Integrity Controls

Constraints: Constraint is a rule that can be applied on a table or a column of a table. Constraints can also be used to prevent mistakes in data entry. Constraints can be categorized into following,

1. Entity Integrity Constraints.

Primary key b. Unique

Domain Integrity Constraints

Not null b. Check

Referential Constraint

Constraints can be attached to a table or column in two ways as follows,

1. Table Constraints: Defining separately from definitions of columns in the table for one or more columns (using ALTER command).

2. Column Constraints: Defining within the definition of columns for single column (within CREATE command).

Primary Key Constraint: A key field, which can be used to identify records (entity values) uniquely. A key field can be a composite. A table may contain more than one key fields, but one of the key fields can be treated as primary key field.

Syntax:

Table Constraint:

[CONSTRAINT constraint_name] PRIMARY KEY(column, column, . . .)

Ex: alter … add primary key(deptno) …

Column Constraint:

[CONSTRAINT constraint_name] PRIMARY KEY

Ex: create … deptno number(3) primary key, . . .

Unique Constraint: Column with UNIQUE constraint specifies that all entities having different (unique) values for that column.

Syntax :

Table Constraint :

[CONSTRAINT constraint_name] UNIQUE(column,column,..)

Ex: alter … add unique(ssno) …

Column Constraint:

[CONSTRAINT constraint_name] UNIQUE

Ex: Create … Ssno varchar2(8) unique, . . .

Null / Not Null Constraint: This constraint can be defined only as Column constraint. If a column has NOT NULL constraint, null values are not permitted (i.e. It would not allow you to omit value for that column). Columns without NOT NULL, is referred as NULL (i.e. it would allow you to omit value for that column if necessary is treated as null value).

EX: create . . . eno number(3) not null, apt_number varchar2(10), . . .

Check Constraint: This constraint is for defining a condition that each row must satisfy. Same Syntax for both table constraint and column constraint.

Syntax:

[CONSTRAINT constraint_name] CHECK(condition)

Ex 1: create … eno number(3) check(eno>0 and eno <=500), …

Ex 2: alter table emp add constraint ck_emp check(sal + comm <10000);

Foreign Key Constraint: It provides referential integrity rules either within a table or between table. This can be used in relationship with either a primary or unique key elsewhere. Usually foreign key of one table (called child table ex. emp) refers primary key of another table (called parent table ex. dept).

Syntax:

Table Constraint:

[CONSTRAINT constraint_name] FOREIGN KEY (column, column, . . .) REFERENCES table_name(column,column, .. .) [ON DELETE CASCADE]

Ex: alter … add constraint fk_dno foreign key(dno) references dept(dno);

Column Constraint:

[CONSTRAINT constraint_name] REFERENCES table_name(column,column, . . .)

Ex: create table dept(dno number(3) primary key, dname varchar2(15));

Ex: create table emp(eno number(3) primary key, . . . dno number(3) references dept(dno));

As the result of FOREIGN KEY constraint, a record in parent (ex.dept) could not be deleted if rows exists in child (ex.emp) with the same value of reference field. The ON DELETE CASCADE is the option can be appended to FOREIGN KEY syntax (only for table constraint), and can be used to delete child automatically if parent is deleted.

Ex: alter … add constraint fk_dno foreign key (dno) references dept (dno) on delete cascade;

10.3.5 Creating Table Definitions

Table definitions may be changed by ALTERing column specifications.
ALTER TABLE command may be used to add new columns to an existing table.

We can use three commands to alter tables, ADD, DROP, and ALTER. For example, to an an attribute (column) such as a cell phone number to a table, we can specify

```
ALTER TABLE CUSTOMER_T
   ADD (cell_phone VARCHAR (10));
```

←· a new attribute (cell phone) is added

Note that this add has no impact on any of the code (queries and reports) written except those in which this attribute is to be printed. All others remain unchanged. This would not likely be the case on old file-based systems.

This command is invaluable for the always expected modifications that have to be made to databases after they have been created. (As invaluable as it is, it is easier to add an attribute from the design view of a table.)

Note that the attribute is added to all instances of the table. Normally, when adding an attribute, we want to allow the values to be NULL at least until we have entered the values for all table instances.

10.3.6 Removing Table

```
DROP TABLE customer_t;
```

(Note, the Security privileges (GRANT/REVOKE) are not implemented in Access. The limited security of Access is created using the Tools/Security menu.

The DROP command can only be executed by a person that was granted the DROP ANY TABLE system privilege. Furthermore, to maintain data integrity the DROP may be qualified by the RESTRICT or CASCADE. If restrict is specified, the command fails if there are any dependent objects such as views or constraints that reference the table. If CASCADE is specified, all dependent objects are removed.

10.4 Inserting, Updating & Deleting Data

Once tables and views have been created, it is necessary to populate them with data and maintain those data before queries can be written. The INSERT command which is used add the records into table.

```
INSERT INTO customer_t
        VALUES  (001, ‘Contemporary Casuals’, ‘100 Main St.’, ‘Phila’, ‘PA’, ‘19111);

                (←    all data attributes are added.)

  INSERT INTO customer_t (customer_id, customer_name)

        VALUES (100, ‘New customer’);  ← only 2  attributes are entered

  INSERT INTO ca_customer_ SELECT FROM customer_t
            WHERE state = ‘CA’    t

      ( Populating a table from another table (not  in Access)

  INSERT INTO order_t (order_id, customer_id, order_date)
          VALUES (123, 100, ’25-jan-01')    ←          entering date
```

***** Note that data values to be inserted in the same order as the columns appear in the table. If only some fields are to be defined by the insert, NULL values can be used for the others or you can specify the particular fields to be filled in (for the indicated attribute). When fields are specified, they should be in the same order as they appear in the table and the data must also be in that order

10.4.1 Batch Input

The INSERT command is used to enter one row of data at a time or to add multiple rows as the result of a query. Some versions of SQL have a special command or utility for entering multiple rows of data as a batch: the INPUT command. Oracle includes a program, SQL*Loader, which runs from the command line, and can be used to load form a file into the database. This popular program is tricky to use and is not included in the scope of this taxt.

10.4.2 Deleting Database Contents

```
UPDATE product_t
        SET unit_price = 775.00
              WHERE product_id = 7;
```

: It allows you to remove one or more rows from a table.

```
        DELETE  FROM  table_name [WHERE condition];

DELETE FROM customer_t            ← deleting rows that meet a certain criterion
        WHERE state = 'PA';
   DELETE FROM customer_t;                      ← All rows are deleted
```

10.4.3 Changing the Database Contents

UPDATE : It allows you to change values of rows in a table.

```
          UPDATE table_name SET column=expression
           [WHERE condition];

UPDATE product_t
        SET unit_price = 775.00
              WHERE product_id = 7;
```

10.5 Internal Schema Definition in RDBMSs

The internal schema of a relational database can be controlled for processing and storage efficiency. The ISO standard does not address most efficiency issues so a number of these definitions are not available in all DBMS systems.

Some of the things we can do to tune a DB are indicated below.

10.5.1 Creating Indexes

We can index primary of secondary keys to increase the speed of row selection, table joining or row ordering.

This in turn can provide more efficient sequential and random access to base table data.

```
CREATE INDEX name_idx ON CUSTOMER_T (customer_name);
```

It is important to note that creating indexes takes up more table space so you want to carefully choose the keys you decide to index. Overhead is also required every time and indexed attribute value is changed.

Dropping Indexes

Dropping an index (to increase the speed of table updates):

```
DROP INDEX name_idx
```

10.6 Summary

This Lesson has introduced the SQL language for relational database definition (DDL), manipulation (DML), and control (DCL), commonly used to define and query relational database management systems (RDBMS).

This standard has been criticized for many flaws, and in reaction to these, and to increase the power of the language, extensions are constantly under review by the ANSI X3H2 committee and ISO/IEC JTC1/SC 21/WG3 DBL. The current standard is known as SQL-99.

The establishment of SQL standards and conformance certification tests has contributed to relational systems being the current dominant form of new database development.

Benefits of the SQL standards include reduced training costs, improved productivity, application portability and longevity, reduced dependence on single vendors, and improved cross-system communication.

The SQL environment includes an instance of an SQL DBMS along with accessible databases and associated users and programs. Each database is included in a catalog and has a schema that describes the database objects. Information contained in the catalog is maintained by the DBMS itself, rather than by the users of the DBMS.

The data definition language (DDL) commands of SQL are used to define a database, including its creation and the creation of its tables, indexes, and views. Referential integrity is also established through DDL commands.

The data manipulation (DML) commands of SQL are used to load, update, and query the database through use of the SELECT command. Data control language (DCL) commands are used to establish user access to the database.

10.7 Technical Terms

SQL: SQL (Structured Query Language) is a standard interactive and programming language for getting information from and updating a database.

Schema: A description of the data represented within a database.

Attribute: Characteristic of an entity/object, eg: the *name* of a person.

Data Definition Language: **L**anguage sub-system of a data management system that is used to define the structure of the database.

Domain: A domain is the set of allowable values for one or more attributes.

Degree: The degree of a relation is the number of attributes it contains.

Nested Query: Nested query is when a SELECT statement embedded within another SELECT statement.

10.8 Self Assessment Questions

1. What is SQL? Explain Features of SQL?
2. Write about structure of SQL?
3. What are various string operations?
4. what are SQL-92 and SQL-99?Briefly describe how SQL-99 differs from SQL-92.
5. describe a relational DBMS, its underlying data model, data storage structures, and manner4 of establishing data relationships.
6. list six potential benefits of achieving an SQL standard that is widely accepted.
7. What are various Set operations?
8. Distinguish among DDC,DMC and DCC.
9. Explain various Aggregate functions with example queries?

10.9 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-11

SQL- II

11.0 Objective

The objective of this Lesson is to introduce the main concepts of data retrieval in the context of database information systems using structured Query Language

- Data retrieving with SELECT command
- Retrieving data by specifying expressions, functions
- Usage of Special operators in SQL
- Write single table queries using SQL commands.
- Establish referential integrity using SQL.

Structure

11.1 Introduction

The SQL environment includes an instance of an SQL database management system along with databases accessible by that DBMS and the users and programs that may use that DBMS to access the databases.

Each database is contained in a catalog, which describes any object that is a part of the database, regardless of which user created that object. If more than one user has created objects in the database, combing information about all users schemas will yield information for the entire database. Users can browse the catalog contents by using SQL select statements.

11.2 Processing Single Tables

Queries that affect only a single table. The SELECT command with its various clauses, that allows on e to query the data contained in the tables and ask many different questions, or a d hoc queries. The basic construction of an SQL command is fairly simple and easy to learn. However, because the basic syntax is relatively easy to learn, it is also easy to write SELECT queries that are syntactically correct but do not answer the exact question that is intended. Before running queries against a large production database, always test queries carefully on a small test set of data to be sure that they are returning the correct results. Let we begin by exploring SQL

11.2.1 Clauses of the SELECT Statement

The SELECT statement is used to retrieve information from the database objects. All relational algebra operations can be implemented in this statement.
Syntax:

SELECT clause FROM clause [WHERE clause] [ORDER BY clause];

Where SELECT clause, which lists the columns to be displayed, FROM clause, which lists tables involved. Column and table names are separated by comma. Character '*' can be used in SELECT clause to select all columns from the table(s).

Ex 1: select eno, ename from emp;

Ex 2: select eno, dname from emp, dept where emp.dno = dept.dno;

Ex 3: select * from emp;

The SELECT clause, also includes following,

Arithmetic expressions.

Columns aliases.

Concatenated columns.

Literals.

DISTINCT clause

11.2.2 Using Expressions

Arithmetic expressions: Combination of one or more values, operators(like +, -, *, and /) and functions(power, round, etc), which evaluate to a values.

Ex: select ename, salary+250*12 from emp;

Ex: select ename, (salary+250)*12 from emp;

Ex: select ename, round(salary) from emp;

Column aliases A Column alias gives a column with an alternative heading on output.

Ex: select ename, salary*12 annual_salary from emp;

Concatenation operator An operator '||' allows columns to be linked to other columns, arithmetic expression or constant value and vice versa.

Ex: select eno||ename from emp;

Literal A literal is any character, expression, number included on the SELECT statement list, which is output for each row returned.

Ex: select eno, 'working for', dname from emp, dept where emp.dno = dept.dno;

Ex: select ename ||' working for '|| dname from emp, dept where emp.dno = dept.dno;

11.2.3 Using Functions

Functions make the basic query block more powerful, and are used to manipulate data values. Functions differ in the number of row upon which they act. Functions are of two general types,

1. Single row(or scalar) functions.

Number Functions
Character Functions
Returning Character Values

Returning Number Values
Date Functions
Conversion Functions
Other Functions

2. Group (or aggregate) functions.

Single Row Functions These functions can be used for calculations on single record. Single row functions can appear in select lists and WHERE clauses(with or with out GROUP BY clause).

1. Absolute - ABS(n) : Returns absolute value of n.

Ex: SELECT ABS(-15) “ABSOLUTE” FROM DUAL;

2. Logarithm to base 10 - LOG(m,n) : Returns the logarithm base m of n. Where m is +ve and should be >1, n can be any +ve.

    Ex: select log(10,100) “log base 10 of 100” from dual;

3.. Sign - SIGN(n): Returns -1, if n<0. Returns 1, if n>0. Otherwise returns 0.

    Ex: select sign(-15) “sign” from dual;

4. Square Root - SQRT(n): Returns square root of n, Where n can not be -ve.

    Ex: select sqrt(25) “square root” from dual;

5.. Truncate - TRUNC(n,[m]): Returns n truncated to m decimal places. m can be omitted, to 0 places. m can be -ve, to truncate(make zero) m digits left of the decimal point.

    Ex: select trunc(15.79) “ truncate” from dual;

Character Functions Character functions can be classified into two types.

I. Character functions **returning character** values are

1. CHR(n): Returns the character having the binary equivalent to n in the database character set.

    Ex: select chr(75) “character” from dual;

2. CONCAT(char1,char2): Returns char1 concatenated with char2.

    Ex: select concat(eno,ename) from bujji123;

4. LOWER(char): Returns char with all letters lowercase.

    Ex: select lower(‘Jkc College’) from dual;

5. UPPER(char): Returns char with all letters uppercase

    Ex: select upper(‘Jkc College’) from dual;

II. Character Functions **returning number** Values are,

1. ASCII(char): Returns the decimal representation of char.

    Ex: select ascii(‘A’) from dual;

2. INSTR(char1,char2[,n[,m]]) : Searches char1 beginning with its nth character for the mth occurrence of char2 and returns the position of character in char1 that is the first character of this occurrence. If n is -ve Oracle counts and searches backward from the end of char1. The value m must be +ve.

Ex: select instr(‘corporate floor’,’or’,3,2) from dual;

Date Functions The date functions are,

1. SYSDATE: Returns the current date and time.

   Ex: select to_char(sysdate,’dd-mon-yyyy hh12') from dual;

2. ADD_MONTHS(d,n): Returns the date after n months to the date d.

   Ex: select add_months(sysdate,60) “date after 60 months” from dual;

Conversion Functions Convert a value from one data type to another.

1. TO_CHAR(d[,fmt]): Converts d of DATE data type to a value of VARCHAR2 data type in the format specified by the date format fmt. If you omit fmt, d is converted to a VARCHAR2 value in the default date format.

   Ex: select TO_CHAR(hiredate,’month dd, yyyy’) new_date_format from emp where ename=‘smith’;

2. TO_DATE(char[,fmt]) : Converts char of CHAR or VARCHAR2 data type to a value of DATE data type. The fmt is a date format specifying the format of char. If fmt is omit, char must be in default date format.

   Ex: select to_date(‘october 25, 2002’,’month dd, yyyy’) to_date from dual;

Other Functions

1. GREATEST(expr[,expr] . . .): Returns greatest of the list of expressions.

   Ex: select greatest(‘ramesh’,’suresh’,’kamalesh’) greatest from dual;

   Ex: select greatest(33,423,87,345,76,32) greatest from dual;

2. LEAST(expr[,expr] . . .): Returns the least of the list of expressions.

   Ex: select least(‘ramesh’,’suresh’,’kamalesh’) least from dual;

Group Functions These function can be used for calculations on groups of selected records. Group functions can appear in select lists, GROUP BY and/or HAVING clauses. They are,

Examples on Functions:

1..AVG([DISTINCT/ALL]n): Returns average value of the field n.

   Ex: select avg(sal) average from emp;

2. COUNT({*/[DISTINCT/ALL] n}): Returns the number of rows in the query .

   Ex: select count(job) from emp;

   Ex: select count(distinct job) from emp;

3. MAX([DISTINCT/ALL]n): Returns maximum value of field n.

Ex: select max(sal) from emp;

4. SUM([DISTINCT/ALL]n): Returns sum of values of field n.

Ex: select sum(sal) from emp;

11.2. 4 Using Wildcards

The use of the asterisk (*) as a wildcard in a SELECT statement has been previously shown. Wildcards may also be used in the WHERE clause where an exact match is not possible. Here, the keyword LIKE is paired with wildcard characters and usually a string containing the characters that are known to be desired matches. The wildcard character %, is used to represent may collection of characters, Thus, using LIKE '%Desk' when searching PROCDUCT_DESCRIPTION will find all different types of desks carried by Pine Valley Furniture. The underscore '-' is used as a wildcard character to represent exactly one character, rather than any collection of characters. Thus, using LIKE '_-drawer' when searching PRODUCT _ NAME will find any products with specified drawers, such as 3-drawer, 5-drawer, or 8-drawer dressers.

11.2.5 Comparison Operators

= , >, >= , <, <=

<>	not equal to
!=	not equal to
*	Wild card
%	Any collection of characters, used with LIKE operator, LIKE '%desk' ← Not in Access

11.2.6 Using Boolean Operators

AND	joins two or more conditions and returns results only when all conditions are true.
OR	joints two or more conditions and returns results when any conditions are true
NOT	negates an expression

SQL operators The **SQL operators**, those operate with all data types can be used in SELECT statement. There are four SQL operators,

1. BETWEEN . . . AND . . .: Test for values between, inclusive of low and high range.

Ex: select * from emp where salary between 2000 and 4500 order by salary;

2. IN (list): Test for values in a specified list.

Ex: select * from emp where dob in ('15-aug-1947', '26-jan-1950');

3. LIKE: Test for values that match a character pattern. Two symbols can be used to construct the search string are % (for any sequence of zero or more characters) and _ (for any single character).

Ex: select * from emp where ename like 'S%';
select * from emp where ename like '____';

(i.e list of all employee whose name exactly 4 characters in length.)

4. IS NULL: Test for values that are null.

Ex: select * from emp where dob is null;

The **SQL negation Operators** for negating values of the above operators are,

1. NOT BETWEEN
2. NOT IN
3. IS NOT NULL
4. NOT LIKE

Ex: select * from emp where salary not between 2500 and 4500;

Ex: select * from emp where ename not like 'S%';

Ex: select * from emp where dob not in('15-aug-1947','26-jan-1950');

Ex: select * from emp where dob is not null;

11.2.7 Ranges

The comparison operators < and > are used to establish a range of values. The keywords BETWEEN or NOT BETWEEN can also be used. For example, to find those products with a standard price between $200 and $300, the following query could be used.

Query: Which products in the PRODUCT view have a standard price between $200 and $300?

```
SELECT PRODUCT_DESCRIPTION, STANDARD_PRICE
   FROM PRODUCT_V
       WEHRE STANDARD_PRICE > 199 AND STANDARD_PRICE < 301;
```

Result:

```
PRODUCT_NAME              STANDARD_PRICE
Coffee Table              200
Computer Desk             250
```

The same result will be returned by this query.

Query: Which products in the PRODUCT view have a standard price between $200 and $300?

```
SELECT PRODUCT_DESCRIPTION, STANDARD_PRICE
   FROM PRODUCT_V
         WHERE STANDARD_PRICE BETWEEN 200 AND 300;
```

Result: Same as previous query.

Adding NOT before BETWEEN in this query will return all the other products in PRODUCT_V because their prices are less than $200 or more than $300.

11.2.8 Distinct

Sometimes when returning row that don't include the primary key, duplicate rows will be returned. For example, look at this query and the results it returns.

Query: What are the order numbers included in the ORDER_LINE table?

```
SELECT ORDER_ID FROM ORDER_LINE_T;
```

Eighteen rows are returned, and many of them are duplicates since many orders were for multiple items.

Result:

```
ORDER_ID

         1001
         1001
         1001
         1002
         1003
         1004
         1004
         1005
         1006
         1006
         1006
         1007
         1007
         1008
         1008
         1009
         1009
         1010
18 rows selected.
```

If, however, we add the keyword DISTINCT, then only one occurrence of each ORDER_ID will be returned, one for each of the ten orders represented in the table.

Query: What are the order numbers included in the ORDER_LINE table?

```
SELECT DISTINCT ORDER_ID FROM ORDER_LINE_V;
```

Result:

```
ORDER_ID

        1001
        1002
        1003
        1004
        1005
        1006
        1007
        1008
        1009
        1010
10 rows selected.
```

DISTINCT and its counterpart, ALL, can only be used once in a SELECT statement. It comes after SELECT and before any columns or expressions are listed. If a SELECT statement projects more than one column, only rows that are identical for every column will be eliminated. Thus, if the statement above also includes QUANTITY, 14 rows are returned because there are now only four duplicate rows rather than eight. For example, both items ordered on ORDER_ID 1004 were for two items, so the second pairing of 1004 and 2 will be eliminated.

Query: What are the unique combinations of order number and order quantity included in the ORDER_LINE table?

Result:

```
ORDER_ID        QUANTITY

        1001            1
        1001            2
        1002            5
        1003            3
        1004            2
        1005            4
        1006            1
        1006            2
        1007            2
        1007            3
        1008            3
        1009            2
        1009                3
        1010                10
```

14 rows selected.

11.2.9 IN and NOT IN Lists

To match a list of values, consider using IN.

Query: List all customers who live in warmer states.

```
SELECT CUSTOMER_NAME, CITY, STATE
  FROM CUSTOMER_V
        WHERE STATE IN ('FL', 'TX', 'CA', 'HI');
```

Results:

CUSTOMER_NAME	CITY	ST
Contemporary Casuals	Gainesville	FL
Value Furniture	Plano	TX
Impressions	Sacramento	CA
California Classics	Santa Clara	CA
M and H Casual Furniture	Clearwater	FL
Seminole Interiors	Seminole	FL
Kaneohe Homes	Kaneohe	HI

7 rows selected.

11.2.10 Sorting Results: The ORDER BY Clause

The ORDER BY Clause is used to sort the rows, which are returned by SELECT. Default ordering of data is ascending.

Ex: select * from emp order by eno;

Ex: select * from emp order by doj desc; (i.e descending order)

Ex: select eno, job, ename from emp order by dno, salary desc;

(i.e. dno in ascending, salary in descending)

11.2.11 Categorizing Results: Group By Clause

This clause can be used to divide the rows in a table into smaller groups. Group functions can be applied on each group. Rows may be pre-excluded with WHERE clause before dividing them into groups.

Ex 1: select deptno, count(*) from emp group by deptno;

Ex 2: select job,avg(sal) from emp where job!='MANAGER' group by job;

11.2.12 Qualifying Results by categories: The Having Clause

The HAVING clause restrict the groups that are grouped already by the GROUP BY clause.

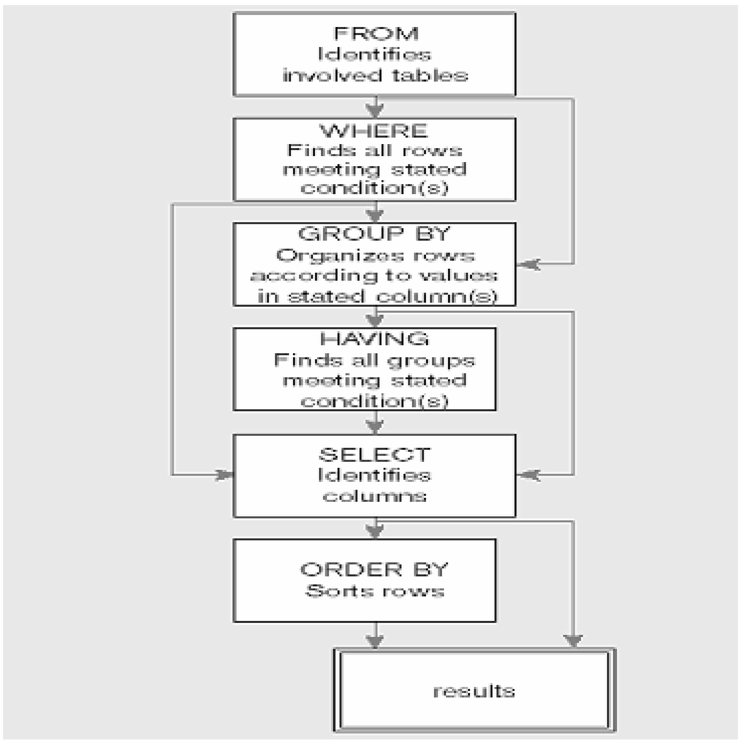
Ex: select deptno,count(*),sum(sal),avg(sal) from emp group by deptno having sum(sal)>10000;

The **Order Of all Clauses** in SELECT Command is,

SELECT column(s)
FROM table(s)
[WHERE row condition(s)]
[GROUP BY column(s)
[HAVING group of rows condition(s)]]
[ORDER BY column(s)];

The same has depicted here

Figure SQL statement processing order


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
GROUP BY
Organizes rows according to values in stated column(s)
HAVING
Finds all groups meeting stated condition(s)
SELECT
Identifies columns
ORDER BY
Sorts rows
results

11.3 Summary

SQL commands may directly affect the base tables, which contain the raw data, or they may affect a database view which has been created. Changes and updates made to views may or may not be passed on to the base tables. The basic syntax of an SQL SELECT statement contains the following keywords: SELECT, FROM, WHERE, ORDER BY, GROUP BY, and HAAVING. SELECT determines which attributes will be displayed in the query results table. FROM determines which tables or views will be used in the query. WHERE sets the criteria of the query, including any joins of multiple tables which are necessary. ORDER BY determines the order in which the results will be displayed. GROUP BY is used to categorize results and may return either scalar aggregates or vector aggregates. HAVING qualifies results by categories.

Understanding the basic SQL, syntax presented in this Lesson should enable the reader to start using SQL effectively, and to build a deeper understanding of the possibilities for more complex querying with continued practice.

11.4 Technical Terms

Scalar aggregate : A single value returned from an SQL query that includes from an SQL query that includes an aggregate function.

Vector aggregate: Multiple values returned from an SQL query that includes an aggregate function.

11.5 Self Assessment Questions

1. Explain SELECT command in detail
2. What is the difference between COUNT, COUNT DISTICT, and COUNT(*) in SQL? When will these three commands generate the same and different results?
3. What is the evaluation order for the Boolean operators (AND, OR, NOT) in an SQL command? How can one be sure that the operators will work in the desired order rather than in the prescribed order?
4. If an SQL statement includes a GROUP BY clause, the attributes that can be requested in the SELECT statement will be limited. Explain that limitation.

11.6 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-12

Advanced SQL

12.0 Objective

☞ Define the following key terms: join, equi-join, natural join, outer join, correlated subquery, user-defined datatype, persistent Stored Modules, trigger, function, procedure, embedded SQL, and dynamic SQL.

☞ Write single and multiple table queries using SQL commands.

☞ Define three types of join commands and use SQL to write these commands

☞ Write noncorrelated and correlated subqueries and know when to write each.

☞ Establish referential integrity using SQL.

☞ Understand common uses of database triggers and stored procedure.

☞ Discuss the SQL-99 standard and explain its enhancements and extensions to SQL-92.

Structure

12.1 Introduction

This lesson demonstrates multiple table queries in some detail. Different approaches to getting results from more than one table are demonstrated, including the use of subqueries, inner and outer joins, and union joins. Triggers, small modules of code that include SQL execute automatically when a particular condition, defined in the trigger, exists.

12.2 Processing of Multiple Tables

All queries we performed so far have used data from one table. The real power of the relational model is being made out of related tables, and the availability of the SQL to process data in the relational database (of many tables).

This module demonstrates multi-table queries including the use of subqueries, different joins and set relations.

To avoid ambiguity in case of identical attribute (column) names in various tables, we must precede the column name with the table name separated by a period.

table_name.attribute_name

Typically the questions we need to answer require data from two or more tables. To process data from several tables, we combine these tables to form a single table by using the Join operations.

The general form of the SELECT statement when a join operation is involved is:

SELECT column_name1 [, column_name2 ....]

FROM table_name1, table_name2

WHERE table_name1.column_name = table_name2.column_name;

The join operation pulls data from two or more tables listed in the FROM clause. The tables are joined together, and then the SQL extracts the data listed in the SELECT clause.

A **join** is a relational operations that causes two or more tables with a common attribute (say, Primary Key - 'PK' and Foreign key - 'FK') to be combined into a single table or view. It is like the UNION operator in algebra. The **joined table** contains the rows (records) which exist in both tables.

The rows appearing in the joined table are selected according to certain rules as specified according to whether we have an INNER JOIN, OUTER JOIN, LEFT INNER JOIN or RIGHT INNER

JOIN.

Note that if we omit the WHERE clause, this would result in a table of all possible combination of rows from all combined tables.

Before looking at formal descriptions of different kinds of joins, you should review the examples illustrated earlier in these notes.

12.2.1 Equi-Join

The join is based on equality between values in the common columns. The common columns appear (redundantly) in the result table.

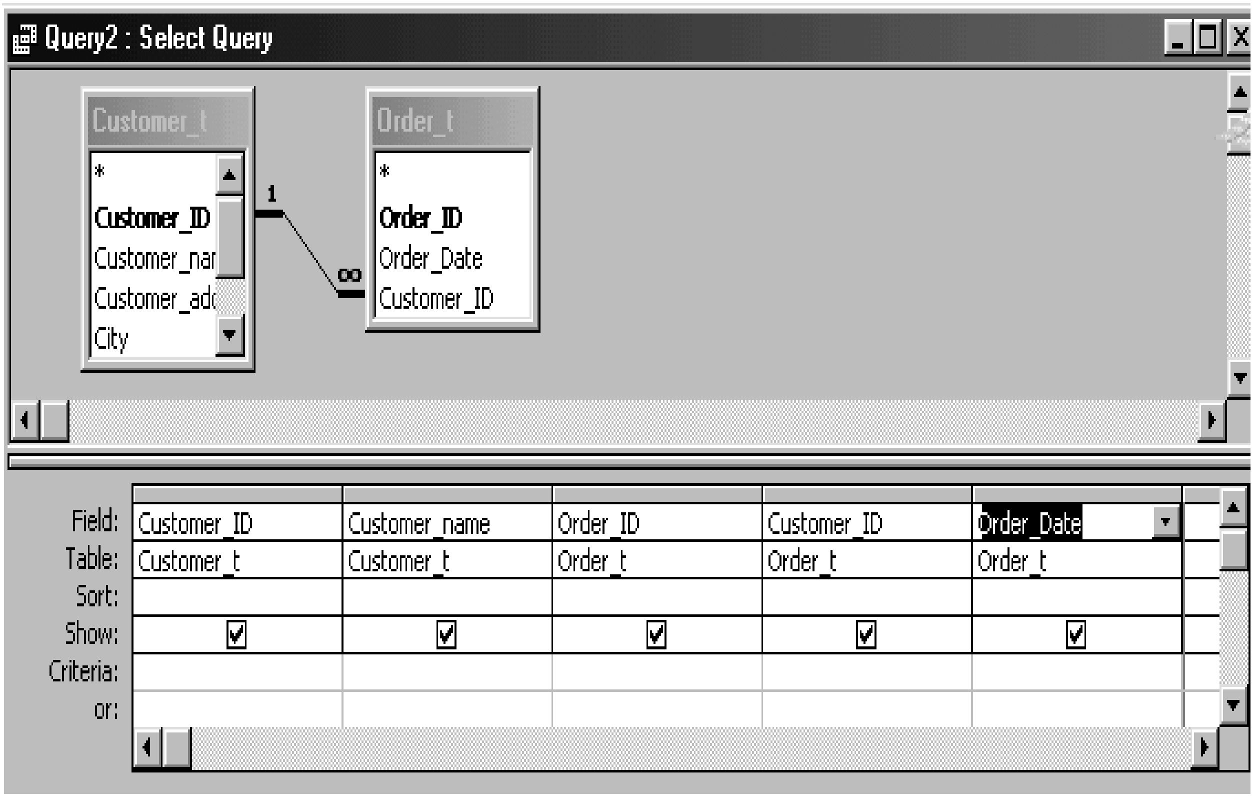
Example: display the customer ID and name and the orders for each customer. The data will come from the customer and order tables, joined together on the common key of customer_ID.

SELECT customer_t.customer_id, order_t.customer_id, customer name, order_id FROM customer, order_t WHERE customer_t.customer_id = order_t.customer_id ;

In ACCESS the Equi-join is called **INNER JOIN** and the command is:

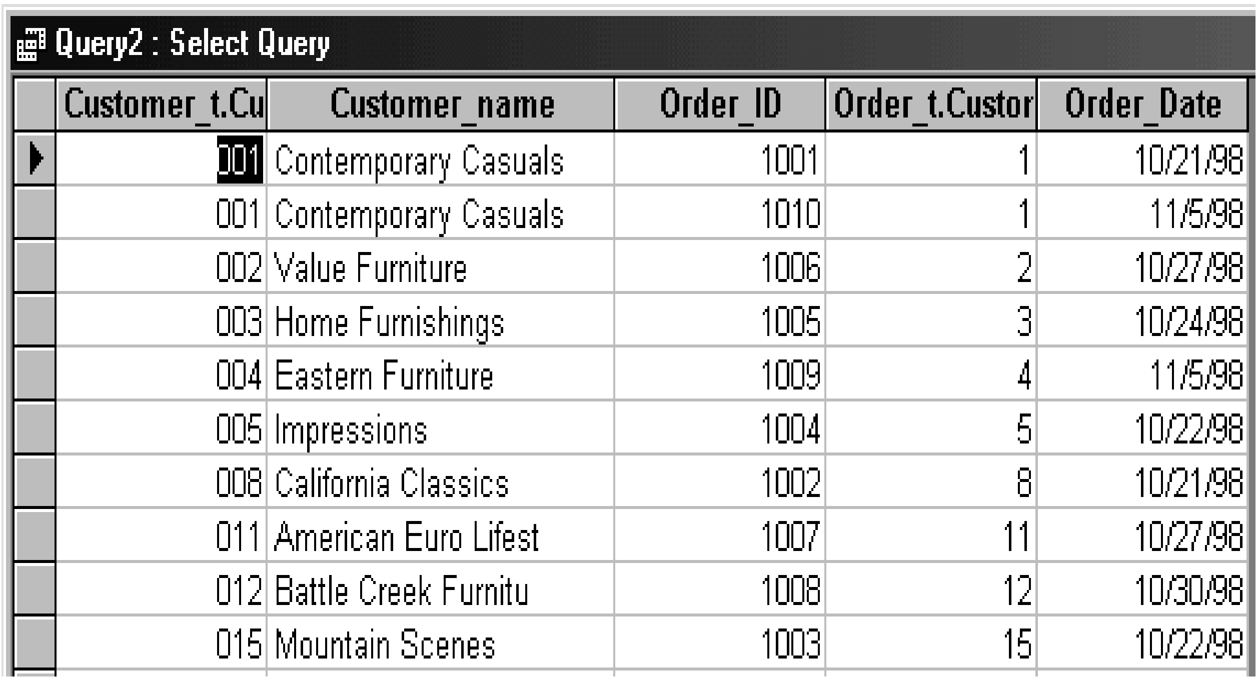
SELECT Customer_t.Customer_ID, Customer_t.Customer_name, Order_t.Order_ID, Order_t.Customer_ID,Order_t.Order_Date FROM Customer_t INNER JOIN Order_t ON Customer_t.Customer_ID = Order_t.Customer_ID;

Note that we are displaying (repeating) the customer_ID) key, (on which the tables are joined on, of both tables.)



Field:	Customer_ID	Customer_name	Order_ID	Customer_ID	Order_Date
Table:	Customer_t	Customer_t	Order_t	Order_t	Order_t
Sort:					
Show:	✓	✓	✓	✓	✓
Criteria:					
or:					

The results of the query are:


Query2 : Select Query

Customer_t.Cu	Customer_name	Order_ID	Order_t.Custor	Order_Date
001	Contemporary Casuals	1001	1	10/21/98
001	Contemporary Casuals	1010	1	11/5/98
002	Value Furniture	1006	2	10/27/98
003	Home Furnishings	1005	3	10/24/98
004	Eastern Furniture	1009	4	11/5/98
005	Impressions	1004	5	10/22/98
008	California Classics	1002	8	10/21/98
011	American Euro Lifest	1007	11	10/27/98
012	Battle Creek Furnitu	1008	12	10/30/98
015	Mountain Scenes	1003	15	10/22/98

12.2.2 Natural Join

Natural Join The natural join is used in cases where the two columns being joined have the same name. It is same as the equi-join, except that the one of the duplicate columns (key) is eliminated. In the example below, one of the customer_id columns would be eliminated from the resulting table.

```
SELECT customer_t.customer_id, customer_name, order_id

FROM customer_t, order_t

WHERE customer_t.customer_id = order_t.customer_id ;
```

Note that only customers that have orders (matched on customer_ID) are displayed. We use the WHERE clause to only join a customer with the corresponding orders, based on the customer_ID in the Customer table and customer_ID in the Order table and we do not bother to list

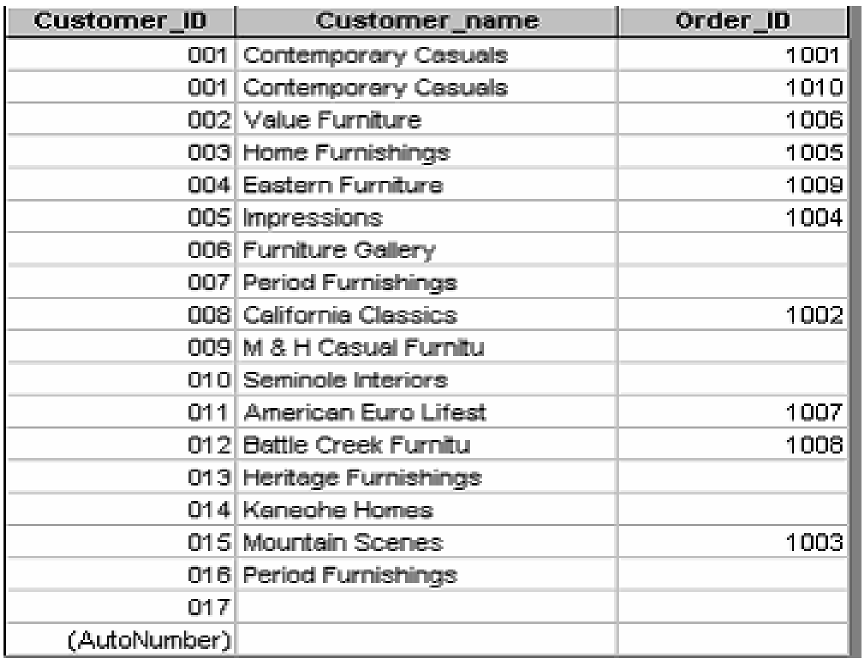
12.2.3 Outer Join

A Join in which rows that do not have matching values in common columns are nevertheless included in the result table.

In the previous join commands, the resulting join table included entries for which the specified keys matched — that is, there were one or more rows in each table with a matching key. For example, in the previous SQL example, we are displaying customers which placed orders. Customers that do not have orders are not displayed, such as customers number 6, 7, 10, 13 and 14. Suppose we want to display ALL customers whether they have orders or not. This can be done using the outer join. An **Outer Join,** is a join in which rows that do not have matching values in common columns are nevertheless included in the result table. As

Example: List customer_id and name and order_id and also include those customer on the customer table, that do not have orders.

```
SELECT customer_t.customer_id, customer_name, order_id
FROM customer_t, LEFT OUTER JOIN order_t
WHERE customer_t.customer_id = order_t.customer_id;
```



Customer_ID	Customer_name	Order_ID
001	Contemporary Casuals	1001
001	Contemporary Casuals	1010
002	Value Furniture	1006
003	Home Furnishings	1005
004	Eastern Furniture	1009
005	Impressions	1004
006	Furniture Gallery	
007	Period Furnishings	
008	California Classics	1002
009	M & H Casual Furnitu	
010	Seminole Interiors	
011	American Euro Lifest	1007
012	Battle Creek Furnitu	1008
013	Heritage Furnishings	
014	Kaneohe Homes	
015	Mountain Scenes	1003
016	Period Furnishings	
017		
(AutoNumber)		

The LEFT JOIN clause indicates that all entries in the left table are in the join table. Thus, customer_t, is to be shown in its entirety along with the orders for the customers that have orders. Note that some people refer to thisjoin as the left outer join and some call it a left inner join. Perhaps simply calling it a left join solves the problemas long as you remember what it does.

12.2.4 Union Join

The results of a UNION JOIN will be table that includes all the data from each table that is joined. The results table will contain all of the columns from each table and will contain an instance for each row of data included from each table.

12.2.5 Sample Multiple Join Involving Four Tables

The capability to relate the objects to each other by joining the tables provides critical business information and reports to employees. Here is a sample join query that involves a four-table join. The query produces a result table that includes the information needed to create an invoice for order #1006.

Ex. Query : Assemble all information necessary to create ian invoice for order number 1006.

```
SELECT CUSTOMER_T.CUSTOMER_ID, CUSTOMER_NAME, CUSTOMER_ADDRESS,
CITY, SATE, POSTAL_CODE, ORDER_T.ORDER_ID, ORDER_DATE, QUANTITY,
PRODUCT_NAME, UNIT_PRICE, (QUANTITY * UNIT_PRICE) FROM CUSTOMER_T,
```

ORDER_T, ORDER_LINE_T, PRODUCT_T WHERE CUSTOMER_T.CUSTOMER_ID = ORDER_LINE.CUSTOMER_ID AND ORDER_T.ORDER_ID = ORDER_LINE_T.ORDER_ID AND ORDER_LINE_T.PROEUCT_ID= PRODUCT_PRODUCT_ID AND ORDER_T. ORDER_ID = 1006

12.2.6 Subqueries

At times we need to retrieve rows in one table based on conditions in a related table. To do so we use sub-queries. A subquery involves embedding a query (SELECT - FROM - WHERE clauses) within the WHERE or HAVING clauses of another (outer) query.

The format of a subquery is:

```
SELECT column-name1 [,column-name2] ...
FROM table-name
WHERE column-name IN
(SELECT column-name          <— (the subqueryis enclosed in parenthesis)
FROM table-name
WHERE (search condition) );
```

The inner query, is executed first and provides values for the search condition of the outer query. The subquery is enclosed in parenthesis and follows the rules of a regular query.

Example :

```
SELECT customer_ID, customer_name
FROM customer_t
WHERE customer_id IN
(SELECT DISTINCT customer_id
FROM order_t) ;
```

Note, the subquery is evaluated first and returns the customer IDs:

1,2,3,4,5,8,11,12 and 15.

The outer query selects the names of customers from the same set — IN(1,2,3,4,5,8,11,12,15).

The qualifiers, **NOT**, **ANY** and **ALL** may be used in front of the IN, or we could use logical operators such as =, <, >.

Subqueries – EXISTS When using subqueries, we can use the conditions:

EXISTS and NOT EXISTS instead of the IN operator.

If the sub-query returns an intermediate result table which contains one or more rows, then the EXISTS will take a value of true, and false if no rows are returned.

12.2.7 Correlated Subqueries

The result of the inner query was used to limit the processing of the outer query. The Correlated Subqueries use the result of the outer query to determine the processing of the inner query. The inner query is somewhat different for each row referenced in the outer query. The inner query must be computed for each outer row, whereas in the earlier examples, the inner query was computed only once for all rows processed in the outer query.

Ex. Query : Show all orders that include furniture finished in natural ash

```
SELECT DISTINCT ORDER_ID FROM ORDER_LINE_T
  WHERE  EXISTS
    (SELECT * FROM PRODUCT_T
            WHERE PRODUCT_ID = ORDER_LINE_T.PRODUCT_ID
            AND PRODUCT_FINISH = ‘Natural ash’);
```

12.2.8 Using Derived Tables

Subqueries are not limited to inclusion in the WHERE clause. They may also be used in the FROM clause, creating a temporary derived table that is used in the query. Creating a derived table that has an aggregate value in it, such as MAX, AVG, or MIN allows the aggregate to be used in the WHERE clause.

Ex. Query : Which products have a standard price htat is higher than the average standard price

```
SELECT PRODUCT_DESCRIPTION, STANDARD_PRICE, AVGPRICE
        FROM
                (SELECT AVG(STANDARD_RPICE) AVGPRICE
                FROM PRODUCT_T), PRODUCT_T
                        WHERE STANDARD_PRICE>AVGPRICE;
```

12.2 9 Combining Queries

The **UNION** operator is used to merge the results of two or more queries into a single result. The general form of the Union operator is placed between the select statements.

SQL executes each query separately and the results from each query are merged with the results of the next query. In the process of merging the queries duplicate rows are removed from the answer.

In order for the queries to be merged, they must be union-compatible; i.e., the data types and size of the corresponding items in each SELECT clause must be identical. The merged results can be ordered using the ORDER BY clause with a numeric sort key referring to the column number.

The general form of the union operator is:

SELECT statement

UNION

SELECT statement

| UNION |

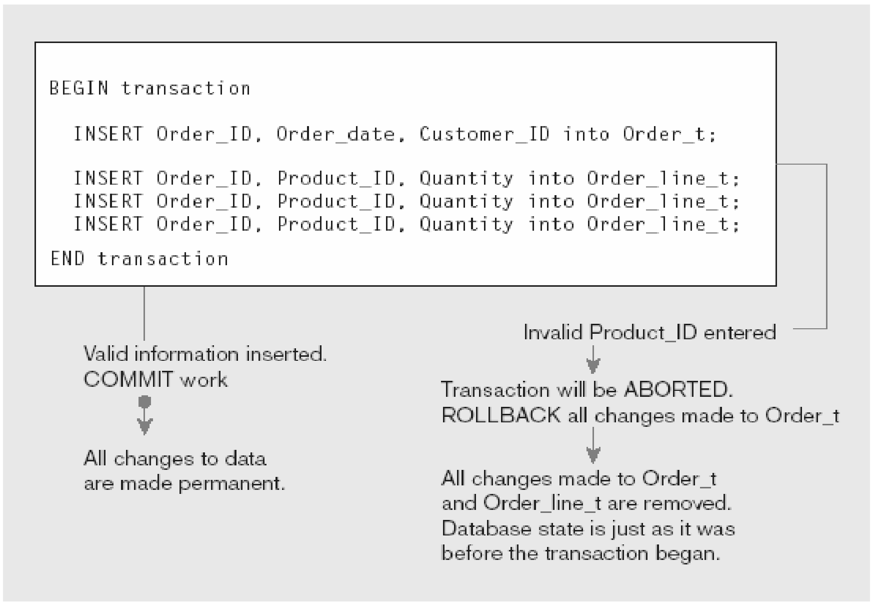
| SELECT statement |

| ORDER BY integer [DESC] |

The SQL executes each query separately and the results from each query are merged with the results of the other query. In the process of merging the queries, duplicate rows are removed from the answer. For the queries to be merged, they must be union-compatible — i.e., the data types and sizes of the corresponding attributes in each SELECT clause must be identical. The merged results can be ordered using the **ORDER BY** clause with a numeric key referring to the column in the SELECT clause.

12.3 Ensuring Transaction Integrity

One of the functions of a DBMS is to ensure transactions integrity. For example, when an order is entered, the order header and all the line items must be entered, or none should be entered in case the operation is aborted. Say, in the process of entering an order with 3 line items, 2 line items were entered but the product_id of the 3rd line was missing. The data entry person pressed the 'Esc' key aborting the transaction. Let's see what happened. The data entry program includes the following SQL code. It identifies the beginning and end of the transaction.



```
BEGIN transaction

  INSERT Order_ID, Order_date, Customer_ID into Order_t;

  INSERT Order_ID, Product_ID, Quantity into Order_line_t;
  INSERT Order_ID, Product_ID, Quantity into Order_line_t;
  INSERT Order_ID, Product_ID, Quantity into Order_line_t;

END transaction
```

Valid information inserted.
COMMIT work
All changes to data are made permanent.
Invalid Product_ID entered
Transaction will be ABORTED.
ROLLBACK all changes made to Order_t
All changes made to Order_t and Order_line_t are removed. Database state is just as it was before the transaction began.

The first INSERT line enters the order followed by three INSERTs for the line items. If valid information inserted then a COMMIT clause is executed for storing the data. If invalid product_id is entered then the transaction will be aborted and ROLLBACK is performed. All changes to the order_t and order_line_t are removed just as it was before the transaction began.

When a single SQL command constitutes the transaction, the RDBMS will automatically COMMIT or ROLLBACK after the command is run. When a transaction involves multiple command lines, it is up to the programmer to identify the beginning and end of the transaction.

12.4 Data Dictionary Facilities

RDBMS store database definition information in system-created tables; we can consider these system tables as a data dictionary. Since the information is stored in tables, it can be accessed by using SQL SELECT statements that can generate reports about system usage, user privileges, constraints, etc. Further, a user who understands the systems table structure can extend existing tales or build other tables to enhance the built in feature.

Some of the Features

- ☞ System tables that store metadata
- ☞ Users usually can view some of these tables
- ☞ Users are restricted from updating them

Some examples in Oracle

- ☞ DBA_TABLES–descriptions of tables
- ☞ DBA_CONSTRAINTS–description of constraints
- ☞ DBA_USERS–information about the users of the system

12.5 SQL-99 Enhancements and Extensions to SQL

User-defined datatype (UDT) SQL-99 allows users to define their own datatype by making it a subclass of a standard type or creating type that behaves as an object. UDTs may also have defined functions and methods.

Programming Extensions

Persistent Stored Modules (SQL/PSM): Extensions defined in SQL-99 that include the capability to create and drop modules of code stored in the database schema across user sessions.

CASE A statement that executes different sets of SQL sequences, according to a comparison of values or the value of a WHEN clause, using either search conditions or value expressions. The logic is similar to an SQL CASE expression, but ends with END CASE rather than END and has no equivalent to the ELSE NULL clause.

IF If a predicate is TRUE, and SQL statement will be executed. The statement ends with an ENDIF, and contains ELSE and ELSEIF statements to manage flow control for different conditions.

LOOP Causes a statement to be executed repeatedly until a condition exists that results in an exit.

LEAVE Statement used to set a condition that results in exiting from a loop.

FOR Statement that will execute once for each row of a results set.

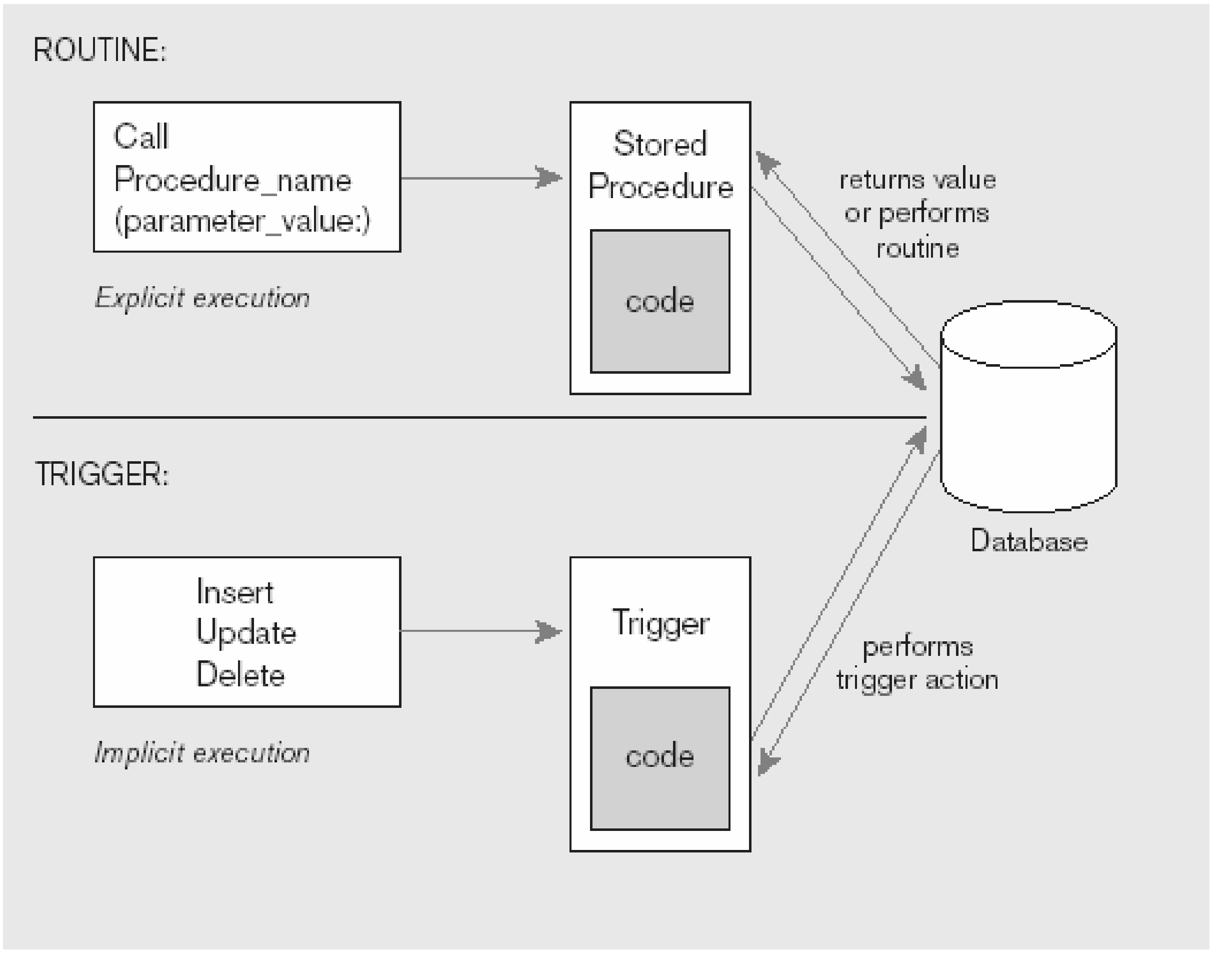
WHILE A statement that will be executed as long as a particular condition exists. Incorporates logic that functions as a LEAVE statement.

REPEAT Similar to the WHILE statement, but the condition is tested after execution of the SQL statement.

ITERATE Statement used to restart a loop.

12.6 Triggers and Routines

Triggers and Routines are stored and executed in the database objects because they are stored in the database and controlled by the DBMS. The code required to create them is stored in only one location and is administered centrally. This promotes stronger data integrity and consistency of use within the database. Since they are stored once, code maintenance is also simplified. Both triggers and routines consist of blocks of procedural code.



12.6.1 TRIGGERS

A named set of SQL statements that are considered when a data modification (INSERT, UPDATE, DELETE) occurs. If a condition stated within the triggers met, then a prescribed action is taken.

Constraints can be thought of as a special case of triggers. They also are applied automatically as a result of data modification commands. Triggers have three parts, the event, the condition, and the action, and these parts are reflected in the coding structure for triggers.

- **Functions**–routines that return values and take input parameters
- **Procedures**–routines that do not return values and can take input or output parameters

Simplified Trigger syntax

```
CREATE TRIGGER trigger_name
        {BEFORE | AFTER | INSTEAD OF} {INSERT | DELETE | UPDATE} ON
        table_name
        [FOR EACH {ROW | STATEMENT}] [WHEN (search condition)]
        <triggered SQL statement here>;
```

12.6.2 ROUTINES

Routines can be either procedures or functions. The term procedure and function are used in the same manner as they are in the other programming languages.

- **Functions**–routines that return values and take input parameters
- **Procedures**–routines that do not return values and can take input or output parameters

Syntax

```
{CREATE PROCEDURE | CREATE FUNCTION} routine_name
([parameter [{,parameter} . . .]])
[RETURNS data_type result_cast]    /* for functions only */
[LANGUAGE {ADA|C|COBOL|FORTRAN|MUMPS|PASCAL|PLI|SQL}]
[PARAMETER STYLE {SQL|GENERAL}]
[SPECIFIC specific_name]
[DETERMINISTIC|NOT DETERMINISTIC]
[NO SQL|CONTAINS SQL|READS SQL DATA|MODIFIES SQL DATA]
[RETURNS NULL ON NULL INPUT|CALLED ON NULL INPUT]
[DYNAMIC RESULT SETS unsigned_integer]      /* for procedures only */
[STATIC DISPATCH]                           /* for functions only */
[NEW SAVEPOINT LEVEL | OLD SAVEPOINT LEVEL]
routine_body
```

12.7 Embedded SQL and Dynamic SQL

Embedded SQL

Including hard-coded SQL statements in a program written in another language such as C or Java

Dynamic SQL

The process of making an application program capable of generating specific SQL code on the fly, as the application is processing.

12.8 Summary

Nested subqueries, where multiple SELECT statements are nested within a single query, are useful for more complex query situations. A special form of the subquery, a correlated, requires that a value be known from the outer query before the inner query can be processed. Other subquires process the inner query return a result to the next outer query, and then that outer query is proposed.

Other advanced SQL topics include the use of embedded SQL and the use of triggers and routines. SQL can be included within the context of many third-generation language including COBOL, C, Fortran and Ada. The use of embedded SQL allows for the development of more flexible interfaces, improved performances, and improved database security. User-defined functions that run automatically when records are inserted, updated, or deleted are called triggers. Procedures are user-defined code modules, which can be called on order to execute.

An amendment to SQL-99 that adds a set of analytical functions has been proposed. Extensions already included now make SQL computationally complete, and include flow control capabilities in a set of SQL specifications known as Persistent Stored Modules. SQL/PSM can be used to create applications, or to incorporate procedures and functions, using SQL datatype directly. SQL-invoked routines, including trigger, functions, and procedures, are also included in SQL-99. Users must realize that these capabilities have been included as vendors-specific extensions previously, and will continue to exist for exist for some time

12.9 Technical Terms

Correlated subquery: In SQL: A subquery in which processing the inner query depends on data from the outer query.

Persistent Stored Modules (SQL/PSM): Extensions defined in SQL-99 that include the capability to create and drop modules of code stored in the database schema across user sessions.

12.10 Self Assessment Questions

1. Define each of the following terms:

   Dynamic SQL, correlated subquery, join, natural Join, outer Join, function, SQL/PSM

2. When is an outer join used instead of a natural join ?

3. Explain the processing order of a correlated subquery.
4. Care must be exercised when writing triggers for a database, what are some of the problems that could be encountered?
5. Explain the structure of a module of code that defines a trigger?
6. Explain the purpose of SQL/PSM
7. When would you consider using embedded SQL and when dynamic SQL?

12.11 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-13

THE CLIENT / SERVER DATABASE ENVIRON-MENT

13.0 Objective

Define the following key terms: Client server systems, file server, fat client, database server, stored procedure, three-tier architecture, thin client, application partitioning, Symmetric multiprocessing (SMP), massively parallel processing (MPP)/shared nothing architecture, Middleware.

- ☞ List several major advantages of the client/server architecture, compared to other comput ing approaches.
- ☞ Explain the three components of application logic : data presentation services, processing services and storage services.
- ☞ Suggest the range of possibilities for partitioning these services in various client/server architectures.
- ☞ Distinguish among a file server, a database server, a three-tiered, and an n-tiered architec ture.
- ☞ Describe middleware and explain how middleware facilities client/server architecture.

Structure

13.1. Introduction

Client/Server System: A networked computing model that distributes processes between clients and servers, which supply the requested services. In a database system, the database generally resides on a server that processes the DBMS. The clients may process the application systems or request services from another server that holds the application program.

Application partitioning: The process of assigning portions of application code to client or server partitions after it is written, in order to achieve better performance and interoperability (ability of a component to function on different platforms).

Symmetric Multiprocessing (SMP): A parallel processing architecture where the processors share a common shared memory.

13.2 Client/Server Architecture

This concept involves separating the processing of an application between two systems. One performs all activities related to database (server) and other performs activities related to application (client). A client or front-end database application also interacts with the database by requesting and receiving information from the database server. A database server or back end is used to Manage the database objects among multiple clients who concurrently request the server for the same data.

Application Logic Components

In client/server architecture that have evolved can be distinguished by the distribution of application logic components across clients and servers. The components are Presentation logic, Processing logic and Storage logic.

Figure 13.1 Application logic components

Presentation Logic
Input
Output
Processing Logic
I/O Processing
Business rules
Data management
Storage Logic
Data Storage and Retrieval

Presentation (or) input/output logic component: This component is responsible for formatting and presenting data on the user's screen or other output devices, and for managing user input from a keyboard or other input device.

Processing logic component: This processing logic handles

(i) data processing logic

(ii) Business rules

(iii) Data management logic

(i) Data processing logic: this logic includes such activities as data validation and identification of processing errors.

(ii) Business rules: Business rules that have not been coded at the DBMS level may be coded in the processing component.

(iii) Data management: this logic identifies the data necessary for processing the transaction or query.

Storage logic component: This component is responsible for data storage and retrieval from the physical storage device associated with the application.

Different types of client/server architectures are

File server architecture

Database server architecture

Three-Tier architecture

13.2.1 File Server Architecture

***File server*:** File server is a devise that manages file operations and is shared by each of the CLIENT PCs attached to the LAN.

In file server architecture all data manipulations occurs at the workstations (client) where data are requested. The client handles the presentation logic, processing logic and much of storage logic. In this architecture one or more file servers are attached to the LAN. Each of these file servers acts as an additional hard disk for each of the client PCs. With a file server each client PC may be called a fat client.

***Fat client*:** A client PC that is responsible for processing presentation logic, processing logic and much of storage logic.

ADVANTAGES:

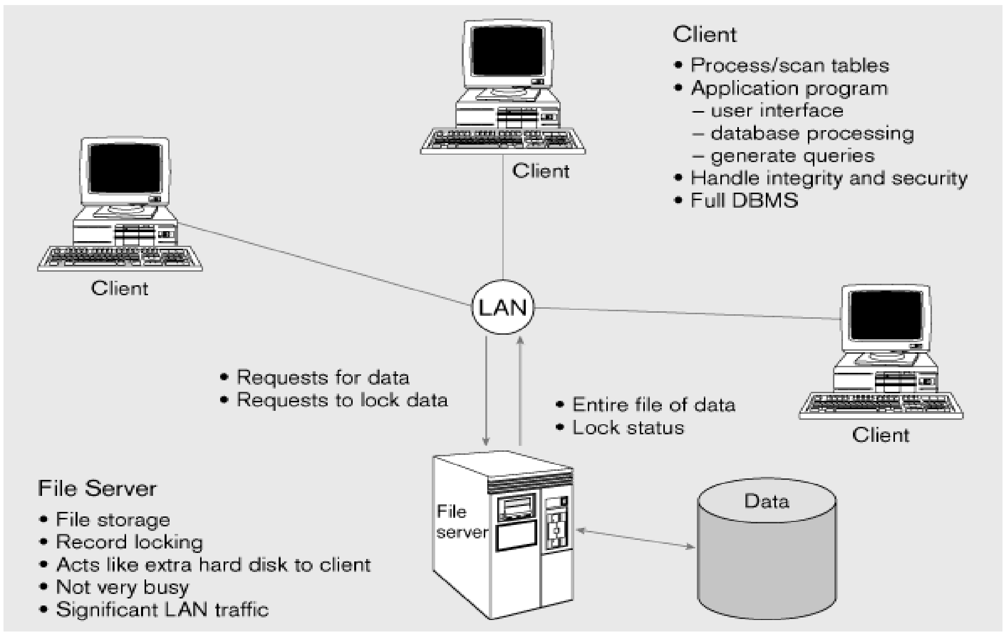
1. Each client PC is authorized to use the DBMS when a database application program runs on that PC.

2. Primary characteristic of file server architecture is that all data manipulation is performed at the client PCs, not at the file server.

3. The file server acts simply as a shared data storage.

4. Entire files are transferred from the server to the client for processing.

Figure 13.2 File Server model



13.2.2 Limitations of File Servers

There are three limitations when using file servers on LANs.

- Considerable data movement is generated across the network.
- The whole file is transferred to the client PC and then scanned at the client to find the few desired records, thus the server does very little work, the client is busy with extensive data manipulation and the network is transferring large blocks of data.
- Each client workstation must devote memory to a full version of the DBMS. This means there is a less memory for application program on the client PC.
- Most important is the DBMS copy in each workstation must manage the shared database integrity. At the same time it is capable of recognize locks and take to initiate the proper locks.

13.2.3 Database Server Architecture

Database server: A device that is responsible for database storage, access and processing in a client/server environment. This is also called a TWO-TIER client/server environment.

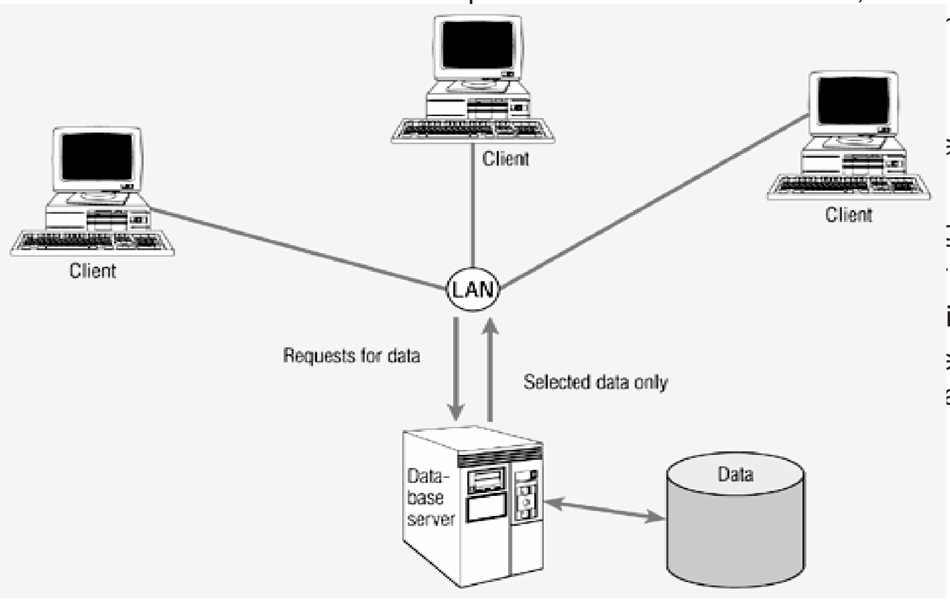
In this system the client workstation is responsible for presentation logic and data processing logic. The database server is responsible for database storage, access and processing. i.e. DBMS is placed on server.

Thin client: A PC configured for handling user interfaces and some application processing, Usually with no or limited local data storage.

Figure 13.3 Database sever architecture (two-tier architecture)

Advantages

☞ With the DBMS placed on the database server, LAN traffic is reduced. Because only those



nitted to the client station, rather than

ally.

g VB or PB to create an application.

portability.

ine users increases.

en in a language such as PL/SQL or

and is stored on the server, where it

3. Improved security
4. Improved data integrity
5. Thinner clients

13.3 Three-Tier Architecture

In general, a **three-tier architecture** includes another server layer in addition to the client and database server layers. Such configurations are also referred to as n-tier, multitier, or enhanced client/server architectures. The additional server in three-tier architecture may be used for different purposes. Often application programs reside on the additional server, in which case it is referred to as an application server. The additional server may hold a local database while another server holds the enterprise database. Each of these configurations is likely to be referred to as a three-tier architecture, but the functionality of each differs, and each I s appropriate for a different situation.

A client/server configuration that includes three layers: a client layer and two server layers. While the nature of the server layers differs, a common configuration contains an application server. The common types are Client Tier,

Application/web Tier and Enterprise Tier.

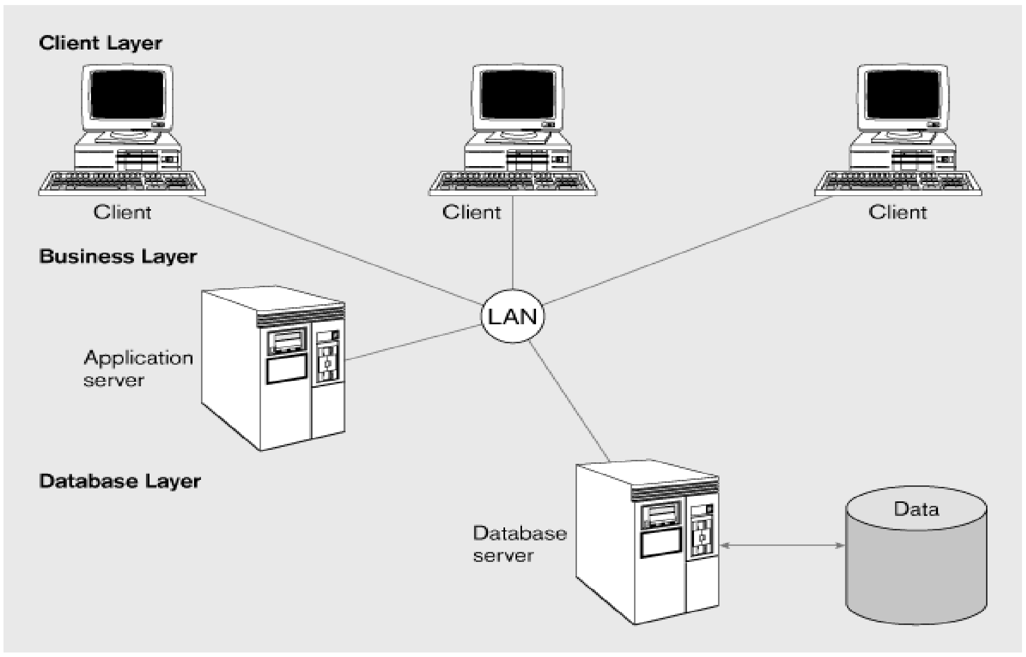
Client Tier: A computer, which concentrates on managing the user-system interface and localized data. This tier also called presentation tier.

.Application/web Tier: Processes HTTP protocol, scripting tasks, perform calculations and provides access to data. This tier also called process services tier.

Most application server vendors including SUN, IBM and ORACLE have adopted JAVA 2 ENTERPRISE EDITION (J2EE) as their standard for application servers.

Enterprise Tier: This Tier performs sophisticated calculations and manages the merging of data from multiple sources across the organization. This tier also called data services tier. A limited view of the client/server architecture considers only the client Tier and a general server tier.

Figure 13.4 Three-tier architecture



ADVANTAGES:

- *Scalability:* Three-tier architectures are more scalable than two-tier. Because of transaction processing monitor.

- *Technological flexibility:* It is easier to change DBMS engines.
- Lower *long term costs:* Because substitution of modules within an application rather than entire application.
- *Better Match of systems to business needs:* New modules can be built to support specific business needs rather than building more general, complete application
- *Improved customer service:* Multiple interfaces on different clients can access the same business process.
- *Competitive advantage:* The ability to implement small modules of code quickly by changing small modules of code.
- *Reduced Rick:* The ability to implement small modules of code quickly and combine them with code purchased from vendors limits the risk assumed with a large-scale development project.

Challenges / Disadvantages:

- *High short-term costs:* Implementing a three-tier architecture required that the presentation component be split from the process component.
- *Tools and training:* Because it is new tools for implementing them are not well developed.
- *Experience:* Because it is new, people have to be trained.
- *Incompatible standards*: Few standards have yet to been proposed for transaction processing monitors
- *Lack of end-user tools that work with middle-tier services:* End-user tools do not yet operate through middle-tier services.

13.4 Partitioning an Application

Placing portions of the application code in different locations (client vs. server) AFTER it is written. It is also possible to add transaction-processing (TP) monitors to client/server systems in order to improve performance.

A TP monitor is a program that controls data transfer b/w client and server in order to provide a consistent environment for on line processing. TP monitors are also useful in distributed environments. Advantages are

1.Improve performance

2.Improve interoperability

3.Balanced workloads

13.5 Role of Mainframe

The role of the mainframe has been uncertain over the last decade as distributed, client/server, and PC computing capabilities have developed. The mission-critical systems, which were resident on mainframe systems as decade ago, have tended to remain on mainframe systems.

Less mission-critical, frequently workgroup-level, systems have been developed using client/server architectures. The popularity of client/server architectures and businesses strong desire to achieve more effective computing in more distributed environments as their perspectives became broader and more global led to their expectation that mission-critical systems would be moved away from mainframes and on to client/server architectures.

- ☞ Mission-critical legacy systems have tended to remain on mainframes
- ☞ Distributed client/server systems tend to be used for smaller, workgroup systems
- ☞ Difficulties in moving mission critical systems from mainframe to distributed
  - ❈ Determining which code belongs on server vs. client
  - ❈ Identifying potential conflicts with code from other applications
  - ❈ Ensuring sufficient resources exist for anticipated load
- ☞ Rule of thumb
  - ❈ Mainframe for centralized data that does not need to be moved
  - ❈ Client for data requiring frequent user access, complex graphics, and user interface

13.6 Using Parallel Computer Architecture

Projects that involve very large databases have benefited from the use of parallel computer architectures. Some projects are possible only because parallel processing of the database gives an acceptably fast response time. Data warehouses have used parallelism to achieve more efficient data extraction, transformation and loading by running on multiple inexpensive servers in a multithreaded environment.

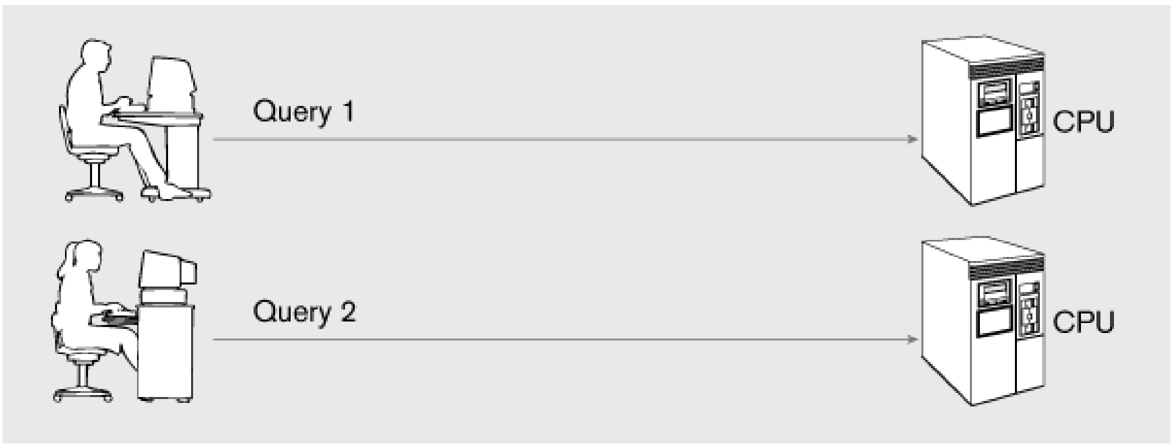
The ability to handle high transaction volumes, complex queries and new data types has proven problematic in many uniprocessor environments. But RDBMSs and the SQL language lend themselves to effecting a parallel environment in two ways.

1. SQL is a nonprocedural language. Therefore queries can be divided into parts, each of which can then be run on a different parallel processor simultaneously.

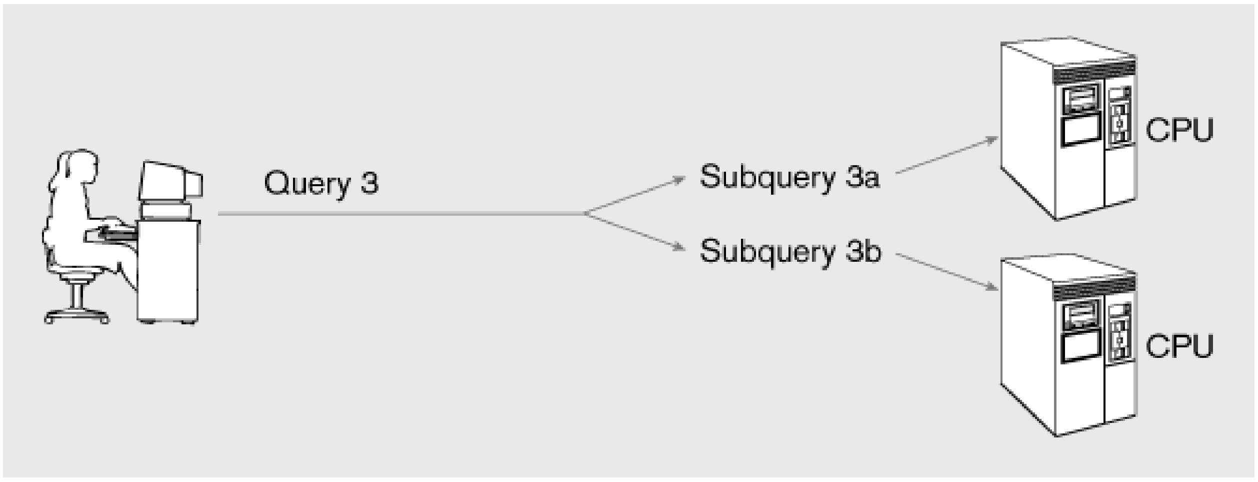
2. Multiple queries can be run in parallel on parallel processors.

Figure 13.5 Parallel Transaction and queries (adapted from Ferguson, 113134)

(a) parallel transactions



(b) Parallel query



13.6.1 Multiprocessor Hardware Architecture

Tightly coupled multiprocessor systems have a common shared memory among all processors (Figure 13.6). This architectures often called SYMMETRIC MULTIPROCESSING (SMP).

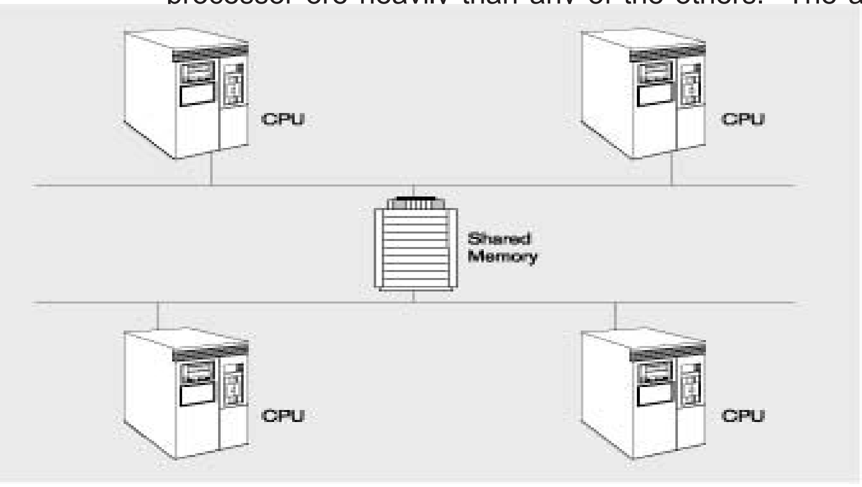
The operating system must be written to operate in such a shared environment so that it can run parts of itself in parallel on any of the processors without developing tendency to use on e processor ore heavily than any of the others. The advantage is that bottlenecks are lessened

essors in symmetric multiprocessing share

uations where data must remain in memory

level.

ure (adapted from Ferguson 113134)



Advantages

- ☞ Work can be divided among the processors if OS contains more than one processor.
- ☞ If one program on one processor crashes the OS running on the other processor can recover the situation.

Disadvantages

- It requires high BANDWITH for faster data rate.
- The conflicts arise when two programs request data at the same location at the same time have to be dealed.

Massively Parallel Processing (MPP): In these architectures each CPU has its own dedicated memory. It requires a complete copy of the OS to be resident in each dedicated memory.

Advantages

- In this system memory contention problems are unlikely.
- It possible to add node in single nodes depending on the budget. (Scalability)
- This system can be used as a data warehouse.
- lower cast.

Figure 13.7 Loosely coupled multiprocessor architecture (adapted from Ferguson, 113134)

13.6.2 Business related Uses of SMP and MPP Architectures

The Parallel processing is enabling organizations to begin to analyze the massive amounts of historical data through the development of data warehouses and data marts. The emergence of parallel processing capabilities underlies the development of data warehousing. Retail organizations, with large amounts of sales data to analyze, have embraced parallel processing systems.

There are several considerations, which organizations should take into account before moving into SMP or MPP systems.

Data analysis requirements: Parallel systems are going to be most effective at solving problems that involves handling large amounts of data (250 GB or larger), Executing complex queries, or processing data that are used by large numbers of concurrent users. The problems should be addressed only if they are of strategic significant to the organization, or else the significant hardware costs, software development costs, and retraining costs will not be worth it to the organization.

Cost Justification: Remember that parallel systems tend to complement rather than replace existing operational systems. Businesses use them to increase revenues rather than to reduce costs. Therefore, traditional cost justification procedures may not be appropriate. Since many of the applications are intended to improve an organization's marketing strategies, treating the parallel system as an investment in marketing capabilities is appropriate.

Traditional Technologies: If the proposed application can be accomplished using traditional, existing technology within the organization, it probably makes sense to continue to rely on that technology. MPP systems are far from mature, and their implementation is sure to challenge any organization.

13.7 Using Middleware

Middleware is often referred to as the glue that holds together client/server applications. It is a term that is commonly used to describe any software component between the PC-client and the relational database in n-tier architecture. Simply put middleware is any of several classes of software that allows as application to interoperate with other software without requiring the user to understand and code the low level operations required to achieve interoperability, Middleware has existed decades.

Another consideration is whether the communication involved is synchronous or asynchronous. With synchronous system, the requesting wait for a response to the request in real time. An on-line banking system where the teller checks an account balance before cashing a check is an example of a synchronous system. Asynchronous systems send a request but do not wait for a response in real time. Rather, the response is accepted whenever it is received. Electronic mail is an example of an asynchronous system with which you are probably familiar.

Asynchronous remote procedure call (RPC): the client programs execute the functions, which are available on other computers, as though they are in their own computer. It can be asynchronous or synchronous. This type of middleware has high scalability but low recoverability, and has been largely replaced by synchronous.

Synchronous RPC : A distributed program using synchronous RPC may call services available on different computers. This middleware makes it possible to establish this facility without undertaking the detailed coding usually necessary to write an RPC.

Publish/Subscribe : This type of middleware monitors activity and pushes information to subscribers. It is asynchronous-the clients, or subscribers, perform other activities between notifications from the server.

Message oriented middleware (MOM): All the client requests are buffered in a queue. Server processes one by one message in the queue.

Object request broker (ORB): The client programs access the objects, which are available on other computers, as though they are in their own computer. ORB is asynchronous.

SQL data oriented data access: Different types of databases on different types of networks can be accessed using a common API.

Application program interface: It sets of routines that an application program uses to direct the performance of procedures by the computer's operating system.

- ☞ ODBC (Open Database Connectivity)
- ☞ OLE-DB (Object Linking and Embedding Database)
- ☞ JDBC (Java Database Connectivity)

13.8 Establishing Client/Server Security

Security measures that should be taken in a client/server environment include measures that are common to securing all systems, but should also include measures taken to secure the more distributed environment of clinet/server architectures.

System-level password security: User name and passwords are typically used to identify and authorize users when they wish to connect to a multi-user client/server system. Security standards should be include guidelines for password lengths, password naming conventions, frequency of password changes, and so on. Password management utilities should be included as part of the network and operating systems.

Database level password security: Most client/server DBMSs have database level password security that is similar so system-level password security. It is also possible to pass through authentication information from the operating system authentication capability. Administration that takes advantage of the pass through capabilities is easier, but external attempts to gain access will also be easier because using the pass-through capabilities reduces the number of password security layers from two to one.

Secure client/server communication: Encryption, transforming readable data (pain text) into unreadable can help to ensure secure client/server communication. Most clients send database users' plain text passwords to database servers. The larger RDBMSs such as Oracle and Sybase have secure network password transmission capabilities. Encryption of all data that are passed across the network is obviously desired, but the costs are high for encryption software. Encryption also affects performance negatively because of the time required to encrypt and decrypt the data.

13.9. Client Server Issues

To succeed, client server project should address a specific business problem with well-defined technology and cost parameters. Certain areas should be carefully addressed in order to improve the chances for building successful client/server application.

- ☞ Accurate business problem analysis
- ☞ Detailed architecture analysis

- ☞ Architecture analysis before choosing tools
- ☞ Appropriate scalability
- ☞ Appropriate placement of services
- ☞ Network analysis
- ☞ Awareness of hidden costs
- ☞ Establish client/server security

Benefits of Moving to Client/Server Architecture

- ☞ Staged delivery of functionality speeds deployment
- ☞ GUI interfaces ease application use
- ☞ Flexibility and scalability facilitates business process reengineering
- ☞ Reduced network traffic due to increased processing at data source
- ☞ Facilitation of Web-enabled applications

13.10 Summary

Connecting databases to the Web so that browsers may interact with sites by placing orders, accessing updated pricing information, and so forth, has also received recent attention. The Web is changing the distribution patterns of data, moving application logic to more centralized servers as browser interfaces are used.

Security is more complex in a client/server environment than in a centralized environment because networks must be secured in addition to the client workstations and the servers. Security measures to be included in a client/server environment include system-level password security, database-level password security, and secure client/server communications.

Client/server issues that should be addressed in order to improve the chances for building a successful client/server application include: accurate business problem analysis, detailed architecture analysis, avoidance of tool-driven architectures, achieving appropriate scalability, appropriate placement of services, adequate network analysis, and awareness of potential hidden costs.

Benefits that may be gained by moving to a client/server environment include deliverance of functionality in stages, flexibility, scalability, less network traffic, and development of Web-enabled applications.

13.11 Technical Terms

Client/server systems: A networked computing model that distributes processes between clients and servers, which supply the requested services. In a database system, the database generally resides on a server that processes the DBMS. The clients may process the application systems or request services from another server that holds the application programs.

Symmetric Multiprocessing (SMP): A parallel processing architecture where the process share a common shared memory.

Application Program Interface (API): Sets of routines that an application program uses to direct the performance of procedures by the computer's operating system.

13.12 Self Assessment Questions

1. Define the following terms
   a. Application partitioning
   b. Client/server architecture
   c. Three-tier architecture
2. List several major advantages of the client/server architecture, compared to other computing approaches.
3. Describe the limitations of file servers.
4. Describe the advantages and disadvantages of database servers.
5. Describe the advantages and disadvantages of three-tier or n-tier architectures.
6. Describe six categories of middleware

13.13 References

Modern Database System Concepts :
Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:
Ramez Elmasri and Shamkant B.Navathe
(Person Education Asia) 2002.

Database System Concepts:
Abraham Silberschatz.Henry F.Korth and S. Sudarshan.
Tata McGraw Hill 2002

Database Application Design and Development
Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:
Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:
Raghurama Krishna and Johannes Gherkin

- Y.SURESH BABU., M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-14

DATA AND DATABASE ADMINISTRATION - I

14.0 Objective

- ☞ Define the following key terms: Data administration, data base administration, database security, encryption, database recovery, transaction, concurrency control.
- ☞ List several major functions of data administration and of database administration.

Structure

14.1 Introduction

Data Administration: A high-level function that is responsible for the overall management of data resources in an organization, including maintaining corporate-wide definitions and standards.

Database Administration: A technical function that is responsible for physical database design and for dealing with technical issues such as security enforcement, database performance, and backup and recovery

14.2 The Role Of Data And Database Administration

Two factors are driving the changes in the data administration and database administration roles. The availability of more technologies and platforms that must be managed concurrently at many organizations, and the increased pace of business changes. Against the background of these changes, it is important to understand traditional role distinctions. This will help us to understand the ways in which the roles are being blended in organizations that have different information technology architecture.

14.2.1Traditional Data Administration

Databases are shared resources that belong to the entire enterprise; they are not the property of a single function or individual within the organization. Data administration is the organization's data in much the same sense that the controller is custodian of the financial resources. Like the controller, the data administrator must develop procedures to protect and control the resource. Also, data administration must resolve disputes that may arise when data are centralized and shared among users, and must plays a significant role in deciding where data will be stored and managed. Data administration is a high-level function that is responsible for the corporate-wide data definitions and standards.

Data administration is responsible for a wide range of functions, including database planning, analysis, design, implementation, maintenance, and protection. Data administration is also responsible for establishing procedures for improving database performance and for providing education, training, and consulting support to users. The data administrator must interact with top management, users, and computer applications specialists.

Selecting the data administrator and organizing the function are extremely important. The administrator must be highly skilled manager capable of resolving differences that normally arise when significant change is introduced into an organization. The data administrator should be a respected, senior-level manager selected from within the organization, rather than a technical computer expert or a new individual hired for the position.

The manager of the data administration function requires a high level of both managerial and technical skills. On the one hand, this person must be capable of eliciting the cooperation of users, who may resist the idea of giving up their private data to a shared database. Also, these users must be convinced of the benefits of adhering to a set of standard definitions and procedures for accessing the database.

On the other hand, the data administrator must be capable of managing a technical staff who deal with issues such as query optimization and concurrency control.

14.2.2Traditional Database Administration

Typically, the role of database administration is taken to be a more hands-on, physical involvement with the management of a database or database. Database administration is a technical function that is responsible for physical database design and for dealing with technical issues, such as security enforcement, database performance, and backup and recovery. The database administrator should be involved in every phase of database development, from database planning through the analysis, design, and implementation stages, and on through operation, maintenance, and modification. The DBA carries through the standards and procedures established by the data administrator, including enforcing programming standards, data standards, policies, and procedures. In organization where there is not a separate database administration function, the DBA also assumes the responsibilities of the data administrator.

Just as the data administrator needs a wide variety of job skills, so does the DBA. A broad technical background, including a sound understanding of current hardware architectures and capabilities and a solid understanding of data processing, is essential. An understanding of the database development life cycle, including traditional and prototyping approaches, is also necessary. Strong design and data modeling skills are essential at the conceptual, logical, and physical levels.

But managerial skills are also critical; the DBA must manage other information systems (IS) personnel as the database is analyzed, designed, and implemented, and the DBA must also interact with and provide support for the end users who are involved with the design and use of the database.

14.2.3 Evolving Approaches to Data and Database Administration

There are no universally accepted data administration and database administration structures, and organizations vary widely in their approaches to data administration. As business practices change, the roles are also changing within organizations. There is, however, a core set of data and database administration functions that must be met in every organization, regardless of the organization structure they have chosen.

These functions may be spread across data administrators and database administrators, or at the other extreme, all of these functions may be handled by a single person. Some of the functions are organization-level functions, and those tend to be classed as data administration functions in organizations that distinguish between data administration and database administration.

Other functions are must more specific to a database or application, and they tend to be classed as database administration functions. In any case, through, effective data and database administration today requires that the following functions be addressed.

- ☞ **Data policies, procedures, and standards** Every database application requires protection established through consistent enforcement of data policies, procedures, and standards. Data policies are statements that make explicit the goals of data administration, such as, “Every user must have a password.” Data procedures are written outline of actions to be taken in order to perform a certain activity. Backup and recovery procedures, for example, should be communicated to all involved employees. Data standards are explicit conventions and behaviors that are to be followed and that can be used to help evaluate database quality. Naming conventions for database objects should be standardized for programmers, for example. Also, increased use of external data sources and increased access to organizational database from outside the organization have increased the importance of employees’ understanding of data policies, procedures, and standards.

- ☞ **Planning** A key administration function is involvement with the development of the organization’s information architecture. Effective administration requires both the understanding of the needs of the organization for information and the ability to contribute to the development of an information architecture that will meet the diverse needs of the typical organization. It is crucial that those involved in data and database administration understand the information requirements of the organization and be able to contribute to the development of the information architecture that will meet those needs.

- ☞ **Data conflict resolution** Databases are intended to be shared, and usually involve data from several different departments of the organization. Ownership of data is a ticklish issue at least occasionally in every organization. Those in data and database administration are well placed to resolve data ownership issues because they are not typically associated with a certain department. Establishing procedures for resolving such conflicts is essential. If the administration function has been given sufficient authority to mediate and enforce the resolution of the conflict, they may be very effective in this capacity.

☞ **Internal marking** While the importance of data and information to the organization has become more widely recognized within the organization, it is not necessarily true that an appreciation for data standards has also evolved. The importance of following established procedures and policies must be proactively instituted through data and database administrators. Effective internal marketing may reduce resistance to change and data ownership problems.

☞ **Managing the information repository** Repositories contain the metadata that describe an organization's data and data processing resources. Information repositories are replacing data dictionaries in many organizations. While data dictionaries are simple data-element documentation tools, information repositories are used by data administrators and other information specialists to manage the total information-processing environment. The repository is an essential tool for data and database administration and is used throughout the entire database system life cycle. An information repository serves as a source of information for each of the following:

1. Users who must understand data definitions, business rules, and relationships among data objects.
2. Automated CASE tools that are used to specify and develop information systems.
3. Applications that access and manipulate data(or business information) in the corporate databases.
4. database management system, which maintain the repository and update system privileges, passwords, object definitions, etc.

Selection of hardware and software The evaluation and selection of hardware and software is critical to an organization's success. The increasingly rapid development of new hardware and software and the rapid change within organizations creates a very demanding situation. A vendor's database that is the turn out to be an unfortunate choice if a competitor release a new version that has different and more appropriate functionality. The current vendor may even go out of business, resulting in a production database that is a night mare to maintain. Data and database administrators are being expected to know more about hardware architectures and to be able to administer both in-house-developed applications and off-the-shelf applications, as organizations strive to meet their information needs more rapidly.

Installing and updating the DBMS Once the DBMS is selected, it must be installed. Before installation, benchmarks of the workload against the database on a computer supplied by the DBMS vendor should be run. Benchmarking anticipates issues that must be addressed during the actual installation.

A DBMS installation can be a complex process of making sure all the correct versions of different modules are in place, all the proper device frivers are present , and the DBMS works correctly with any third-party software products. DBMS vendors periodically update package modules; planning for, testing, and installing upgrades to ensure existing applications still work properly can be time-consuming and intricate. Once the DBMS is installed, user accounts must be created and maintained.

Timing database performance Because database are dynamic. It is improbable that the initial design of the database will be sufficient to achieve the best processing. Performance for he life of the database. The performance of the database needs to be constantly monitored. The design of a database must be frequently changed to meet new requirements and to overcome the degrading effects of many content updates. Periodically, the database must be rebuilt reorganized, and reindexed to recover wasted space, to correct poor data allocation and fragmentation across storage devices, to allocate different buffer space for concurrent processing , and to reset parameters that define the best recovery procedures. Various system tables, needed by the DBMS but not containing business data, periodically need to be rebuilt to be consistent with the new size and use of the database.

Improving database query processing performance T workload against a database most certainly will expand over time as more users find more ways to use the growing amount of data in a database,. Thus, some queries, which originally ran quickly against a small database, may need to be rewritten in a more efficient form to run in a satisfactory time against a fully populated database. Indexes may need to be added or deleted to balance performance across all queries. Data may need to be added or deleted to balance performance across all queries. Data may need to be relocated to different devices to aloe better concurrent processing of queries and updates. Arguably, the vast majority of the time spent by a DBA will be for tuning database performance and improving database query processing time.

Managing data security, privacy, and integrity Protecting the security, privacy and integrity of the organizational databases rests with the database administration function. Here it is important to realize that the advent of the Internet and intranets to which databases are attached, along with the possibilities for distributing data and databases to multiple sites, have complicated the management of data security , privacy, and integrity.

Data backup and recovery The DBA must ensure that backup procedures are established that will allow the recovery of all necessary data, should a loss occur through application failure hardware failure, physical or electrical disaster, or human error or malfeasance. These strategies must be fully tested and evaluated at regular intervals.

Reviewing these data and database administration functions should convince any reader of the important of proper data administration, both at the organizational and the project level. Failure to take the proper steps can greatly reduce an organization's ability to operate effectively, and may even result in its going out of business. Pressures to reduce application development time must always be reviewed to be sure that necessary quality is not being forgone in order to react more quickly, for such shortcuts are likely to have very serious repercussions. Figure 14.1 shows how data and database administration functions are typically viewed with respect to systems development phases.

14.2.4 Evolving Approaches to Data Administration

Many organizations now have blended the data administration and database administration roles together. These organization emphasize the capability to build a database quickly, tuning it for maximum performance, and being able to restore it to production quickly when problems develop. These databases are more likely to be departmental, client/server databases that are developed quickly using newer development approaches such as prototyping, which allow changes to be made more quickly. The blending of data administration and database administration roles also means that DBAs in such organizations must be able to create and enforce data standards and policies. DBAs in such environments are expected to deliver high-quality, robust systems quickly. Across all organizations, database administrators will be expected to maintain required quality levels while decreasing the time required to build a reliable system. Quinlan (1996)) has suggested some changes in data administration and database administration practices that can be made at each stage of the traditional database development life cycle to accomplish these goals:

Database planning Improve selection of technology by selectively evaluating possible products. Be sure to consider each technology's fir with the enterprise data model, seeking to find ways to reduce time required in later stages as a result of careful selection of technology at the database planning stage.

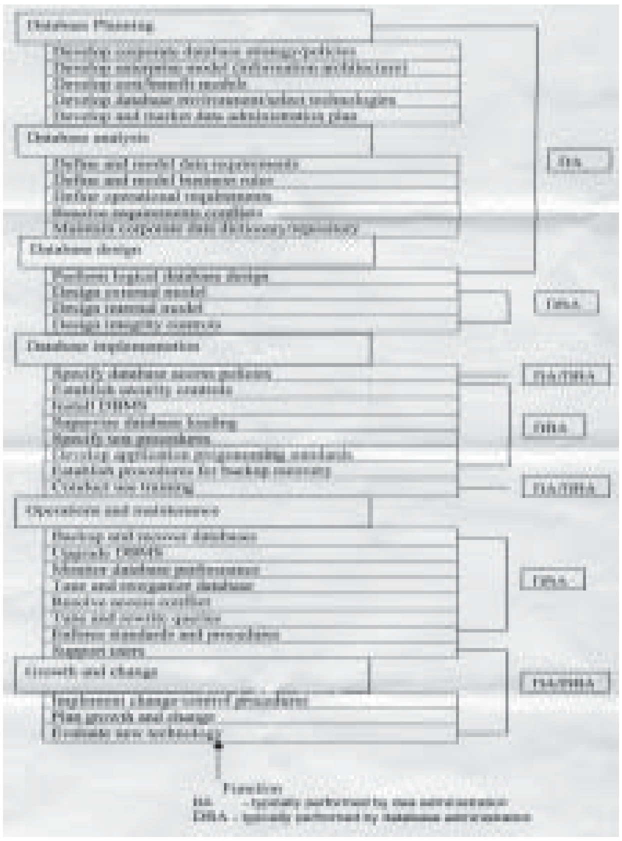
Database analysis Try to work on physical design in parallel with development of the logical and physical models.

Prototyping the application now may well lead to change in the logical and physical data models earlier in the development process.

Database design prioritize application transactions by volume, importance, and complexity. These transactions are going to be most critical to the application. Specifications for them should be reviewed as quickly as the transactions are developed. Logical data modeling, physical database modeling, and prototyping may occur in parallel. DBAs should strive to provide adequate control of the database environment while allowing the developers space and opportunity to experiment.

Data implementation Institute database change-control procedures so that development and implementation are supported rather than slowed. Wherever possible, segment the model into modules that can be analyzed and implemented more quickly. Testing may be moved earlier in the development; use testing and change control tools to build and manage the test and production environments.

Figure 14-1 Functions of data administration and database administration


Database Planning
Database analysis
Database design
Design external model
Design internal model
Design integrity controls
Database implementation
Specify database access policies
Establish security controls
Install DBMS
Supervise database loading
Specify test procedures
Develop application programming standards
Establish procedures for backup recovery
Conduct user training
Operations and maintenance
Backup and recover database
Upgrade DBMS
Monitor database performance
Tune and reorganize database
Resolve access conflicts
Tune and rewrite queries
Enforce standards and procedures
Support users
Growth and change
Implement change control procedures
Plan growth and change
Evaluate new technology
DA
DBA
DA/DBA
Function
DA - typically performed by data administration
DBA - typically performed by database administration

Operation and maintenance Review all timesaving measures that have been taken to ensure that database quality has not been compromised. Consider using third-party tools and utilities whenever possible to save work; other tools such as Lotus Notes may reduce the need for meetings, thus saving time.

The DBA role will continue to evolve. Most see the DBA role as becoming more specialized , evolving into specialties such as distributed database/network capacity planning DBAs, off-the-shelf customizing DBAs, or data warehousing DBAs. The ability to work with multiple databases, communication protocols, and operating systems will be highly valued. Those DBAs who gain broad experience and develop the ability to adapt quickly to changing environment will have many opportunities. It is possible that some current DBA activities, such as tuning, will be replaced by decision support systems able to tune systems by analyzing usage patterns. Opportunities in large companies to continue working with very large databases and opportunities in small and midsize companies to manage desktop and midrange servers should remain strong.

Data Warehouse Administration The significant growth in data warehousing in the past five years for the past five years has caused a new role to emerge, that of a data warehouse administrator (DWA). Inmon (1999) in several articles on his Website has outlined the duties of the DWA. As you might expect, two generalizations are true about the DWA role.

1. A DWA plays many of the same roles as do DAs and DBAs for the data warehouse and data mart databases for the purpose of supporting decision making applications .

2. The role of a DWA emphasizes integration and coordination of metadata and data across many data sources, not necessarily the standardization of data across theses separately managed data source outside the control and scope of the DWA.

Specifically, Inmon suggests that a DWA has a unique charter to :

- ☞ Build and administer an environment supportive of decision- support applications; a DWA is more concerned with time to make a decision than query response time.

- ☞ Build a stable architecture for the data warehouse- a corporate information factory; thus a DWA is more concerned with the effect of data warehouse growth than redesigning existing applications.

- ☞ D3evelop service level agreements with suppliers consumers of data for the data warehouse; thus a DWA works more closely with end users and operational systems administrators to coordinate vastly different objectives and to oversee the development of new applications than do Das and DBAs.

These responsibilities are in addition to the responsibilities typical of any data or database administrator, such as selecting technologies, communicating with users about data needs, making performance and capacity decision, and budgeting and planning data warehouse requirements.

It has estimated that every 100 gigabytes of data in the enterprise data warehouse (EDW) necessitates another DWA. Another metric is that a DWA is needed for each year of data kept in the EDW. The use of custom-built tools for extraction / transformation /loading usually increases the number of DWAs needed.

Data warehouse administrators typically report through the IT unit of an organization, but have strong relationships to marketing and other business areas that depend on the EDW for applications, such as customer or suppler relationship management, sales analysis, channel management, and other analytical applications. DWAs should not be part of traditional systems development organizations, as are many DBAs because data warehousing applications are developments are developed differently from operational systems and need to be viewed as independent from any particular operational systems. Alternatively, DWAs can be placed in the primary end-user organization for the EDW, but this runs the risk of creating many data warehouses or marts, rather than leading to a true, scalable EDW.

14.3 MODELING ENTERPRISE DATA

In the current business environment DBAs are pressured to adapt to change while providing high-quality systems quickly. Key to achieving these expectations is developing an enterprise architecture or Information system Architecture. The development of information system is similar to other complex product development and manufacturing processes. He has studied those processes in order to apply their principles and procedures to development of information systems enterprise architecture. Building enterprise architecture enables an organization to make the step from business strategy to implementation more effectively. An enterprise architecture as " that set of descriptive representations that are relevant for describing an enterprise such that it can be produced to management's requirements (quality) and maintained over the period of its useful life(change)".

All too often in systems development, the enterprise view has been de-emphasized as developers concentrated on building a piece of the enterprise system. It has often been problematic to fit these pieces together into a smoothly functioning enterprise system. Some systems are no longer relevant when completed, or do not remain relevant for long. System maintenance is a major portion of the overall cost of a new system. Problems such as the Year 2000 problem, estimated by the Gartner Group to have cost $400 billion worldwide, arise from past failure to address data element definitions within the enterprise architecture.

Ideally, every database development projects should fit within the enterprise architecture. This means developing conceptual data models that are congruent with the needs of the business and that are then translated into logical data models, physical data models, and implemented systems. Such congruence has been clearly promoted and is a part of every database design life cycle. But the conceptual data models must also fir across Zachman's framework, meshing with the business process model, business network logistics, work flow models within the enterprise business plan. Each component, such as the conceptual data model, has an architectural structure of its own, but it must also fir with the architectural structures of the other components of the framework. The enterprise-wide impacts of execution of each component must be considered, and problems and discontinuities resolved.

Organizations that articulate enterprise architecture and develop standards and procedures to ensure that systems development projects fit within that enterprise architecture should experience benefits. Code can be reused, systems mesh together rather than conflicting with each other, and business objectives are met. As the articulation and incorporation of business rules becomes more sophisticated those organizations with articulated enterprise architecture should be able to implement those business rules more consistently and more quickly than similar organization that have not developed enterprise architecture. Such possibilities provide obvious strategic benefits to organizations in today's competitive environment.

14.4 PLANNING FOR DATABASES

The initial phase of the database development life cycle is planning. It is at this point that a proposed system should be carefully considered from the viewpoint of the enterprise architecture, database design occurs within the constraints of the enterprise information system. During the planning phase, an initial assessment of the proposed database system is made. The proposed systems should contribute to meeting the business's goals and strategies, which have been developed as part of the enterprise architecture. The other components of the Scope layer in the enterprise architecture are also important.

- What data ill be included? have they previously been identified as being important to the business ?
- What processes will the system perform? have these processes been identified as important organization processes ?
- What business locations will the system affect? Should the locations identified be affected by such as system, or have some locations that will be unintentionally affected been identifies?
- Which work units and people will use and be affected by the system, and what is the nature of these effects? What effects on work units beyond the work unit requesting the new system have been identified?
- How will the system fir with significant business events? Will the new system fit with existing schedules?
- How much data will be kept in the database? Do candidate DBMS easily scale as the database grows after installation .

The initial assessment will also include consideration of conducting of continuing use of the exiting system, modifying the exiting system, or replacing the existing system. The decision will rest with the extent to which flaws in the current system exist, and the possibilities for correcting those flaws.

If a new system is a possibility , then study should be conducted. The feasibility study should address the technical aspects of the project, including hardware and software purchase or development costs, operating environment, system size and complexity, and level of experience with similar systems. At this point it is not necessary to identify particular vendors or products, but classes of implementation, such as PC or mainframe hardware, database model, and programming language should be considered. The feasibility study should also address the estimated costs of the project, both tangible and intangible. Costs should be identified as one-time costs associated with the project or recurring costs that will continue throughout the life of the system. Operational feasibility, an assessment of the likelihood that the project will meet its expected objectives, should also be determined. The likelihood of completing the project within the expected time schedule should also be estimated. Any potential legal or contractual problems should be identified now. And, political implications of the project should be considered. Key stakeholder support for the project will greatly improve its prospects of success.

Once the feasibility information has been collected, a formal review of the project that includes all interested parties can be conducted. The focus of this review is to verify all information an assumptions in the baseline plan that has been developed before moving ahead with the project. After the project is approved, problems, defined during the planning phase will be studied further during the analysis phase of the project.

14.5 Summary

The importance of managing data is emphasized in this lesson. The function of data administration, which takes responsibility for the overall management of data resources, includes developing procedures to protect and control data. The functions of database administration, on the other hand, are those associated with the direct management of a database or databases, including DBMS installation and upgrading, database design issues and technical issues such as security enforcement, database performance, and backup and recovery. The data administration and database administration roles are changing in today's business environment, with pressure being exerted to maintain data quality while building high-performing systems quickly. Key to achieving these expectations is developing an enterprise architecture (ISA) and performing adequate database planning.

14.6 Technical Terms

Data administration: A high-level function that is responsible for the overall management of data resources in an organization, including maintaining corporate-wide definitions and standards.

Database administration: A technical function that is responsible for physical database design and for dealing with technical issues, such as security enforcement , database performance, and backup and recovery.

Database Security: Protection of the data against accidental or intentional loss, destruction, or misuse.

14.7 Self Assessment Questions

1. Define the following terms

   Data administration, Database administration, Data steward, information repository
2. What is the function of data steward?
3. Briefly describe six stages in the life cycle of a typical database system.
4. Describe the changing roles of the data administrator and database administrator in the current business environment.

5. List four common problems of ineffective data administration
6. List four job skills necessary for system data administration or data administration. List four job skills necessary for project data administration or database administration.
7. List and describe eleven function that should be addressed in order to achieve effective database administration.
8. What changes can be made administration at each stage of the traditional database development life cycle in order to deliver high-quality, robust systems more quickly?

14.9 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- **C.V.P.R.Prasad,** M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson-15

DATA AND DATABASE ADMINISTRATION - II

15.0 Objective

- ☞ Describing problem of data security, and techniques that are used to enhance security
- ☞ Describing the problem of database recovery, and basic facilities that are included with a DBMS to recover databases.

Structure

15.1 Introduction

Data administrative activities have been developed to help achieve organizations' goals for the effective management of data. Effective data administration provides support for managerial decision making at all levels in the organization.

Database recovery is data administrations response to MURPHY'S law. Data loss or damage may occur because of human error, hardware failure, incorrect or invalid data. The database management system must provide mechanisms for restoring a database quickly and accurately after loss or damage.

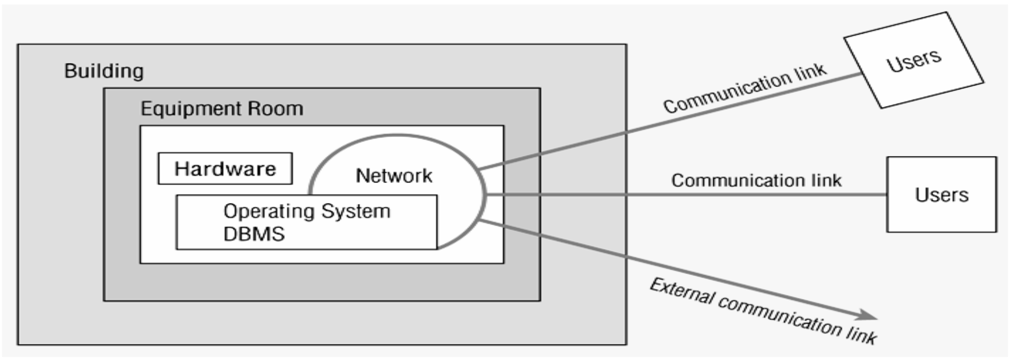
15.2 MANAGING DATA SECURITY

The goal database security is the protection of data from accidental or intentional threats to its integrity and access. The database environment has grown more complex, with distributed databases located on client/server architectures rather than mainframes. Access to data has become more open through the Internet and corporate intranets. As a result, managing data security effectively has become more difficult and time consuming. Data administration is responsible for developing overall policies and establishing adequate data security are discussed below, but first it is important to review potential threats to data security.

15.2.1 Threats to data Security

Threats to data security may be direct threats to the database. For example, those who gain unauthorized access to a database may then browse, change, or even steal the data to which they have gained access. Focusing on database security alone, however, will not ensure a secure database. All parts of the system must be secure, including the database, the network, the operating system, the building(s) in which the database resided physically, and the personnel who have any opportunity to access the system. Figure 15.2 diagrams many of the possible locations for data security threats. Accomplishing this level of security requires careful review, establishment of security procedure and policies, and implementation and enforcement of those procedures and policies. The following threats must be addressed in comprehensive data security plan.

Figure 15.2 Possible of data security threats



Accidental losses, including human error, software, and hardware-caused breaches Establishing operating procedures such as user authorization, uniform software installation procedures, and hardware maintenance schedules are examples of actions that may be taken to address threats from accidental losses. As in any effort that involves human beings, some losses are inevitable, but well thought out policies and procedures should reduce the amount and security of losses. Of potentially more serious consequence are the threats that are not accidental.

Theft and fraud These activities are going to be perpetrated by people, quite possibly through electronic means, and may or may not alter data. Attention here should focus on each possible location shown in Figure 15.2. For example, control of physical security, so that unauthorized personnel are not able to gain access to the machine room, should be established.. Data access policies that restrict altering data immediately prior to a payroll run will help to secure the data. Establishment of a firewall to protect unauthorized access to inappropriate parts of the database through outside communication links is another example of a security procedure that will hamper people who are intent on theft of fraud.

Loss of privacy or confidentiality Loss of privacy is usually taken to mean loss of protection of data about individuals, while loss of confidentiality is usually taken to mean loss of protection of critical organizational data, which may have strategic value to the organization. Failure to control privacy of information may lead to blackmail, bribery, public embarrassment, or stealing of user passwords. Failure to control confidentiality may lead to loss of competitiveness. State and federal laws now exist to require some types of organizations to create and communicate policies to ensure privacy of customer and client data. Security mechanisms must enforce these policies, and failure to do so can mean significant financial and reputation loss.

Loss of data integrity When data integrity is compromised , data will be invalid or corrupted. Unless data integrity can be restored through established backup and recovery procedures, an organization may suffer serious losses or make incorrect and expensive decisions based on the invalid data.

Loss of availability Sabotage of hardware, network, or applications may cause the data to become unavailable to users , which again may lead to severe operational difficulties.

A comprehensive data security will include establishing administrative policies and procedures, physical protections, and data management software protections. The most important security features of data management software follow:

1. Views or subschema's, which restrict user views of the database.
2. Domains, assertions, checks, and other integrity controls defined as database objects, which are enforced by the DBMS during database querying and updating.
3. Authorization rules, which identify users and restrict the actions they may take against a database.
4. User-defined procedures, which define additional constraints or limitations in using a database.
5. Encryption procedures, which encode data in an unrecognizable or limitations in using a database.
6. Encryption procedures, which encode data in an unrecognizable form.
7. Authentication schemes, which positively identify a person attempting to gain access to a database.
8. Backup, journaling, and check pointing capabilities, which facilities recovery procedures.

15.2.2 Views

A view is created by querying one or more of the base tables, producing a dynamic result table for the user at the time of the request, thus, a view is always based on the current data in the base tables that it is built from. The advantages of a view is that it can be built to present only the data to which the user requires access, effectively preventing the user from viewing other data that may be private or confidential. The user may be granted the right to access the view, but not to access the base tables upon which the view is based. So, confining a user to a view may be more restrictive for that user than allowing him access to the involved base tables.

For example , we could build a view for a Pine Valley employee that provides information about materials needed to build a Pine Valley furniture product without providing other information, such as unit price, that is not relevant to the employee's work.. This command creates a view that will list the wood requires and the wood available for each product.

```
CREATE VIEW MATERIALS_V
        AS
        SELECT PRODUCT_T.PRODUCT_ID, PRODUCT_NAME, FOOTAGE,
        FOOTAGE_ON_HAND
                FROM PRODUCT_T, RAW_MATERIALS_T,USES_T
                        WHERE  PRODUCT_T.PRODUCT_ID=USES_T.PRODUCT_ID
                        AND RAW_MATERIALS_T.MATERIAL_ID=
                        USES_T.MATERIAL_ID;
```

The contents of the view created will be updated each time the view is accessed, but here are the current contents of the view, which can be accessed with the SQL command

```
        SELECT * FROM MATERIALS_V
        PRODUCT_ID,        PRODUCT_NAME,    FOOTAGE,    FOOTAGE_ON_HAND
                1       End Table                4             1
                2       Coffee Table             6            11
                3       Computer Desk           15            11
                4        Entertainment Center   20            84
                5       Writer's Desk           13            68
                6       8-Drawer Desk           15            66
                7       Dining Table            15            11
                8       Computer Desk           15             9
                        8 rows selected.
```

The user can write SELECT statements against the view , treating it as though it were a table. Although views promote security by restricting user access to data, they are not adequate security measures, because unauthorized persons may gain knowledge of or access to a particular view. Also, several persons may share a particular view; all may have authority to read the data, but only a restricted few may be authorized to update the data. Finally, with high-level query languages, an unauthorized person may gain access to data through simple experimentation. As a result, more sophisticated security measures are normally required.

15.2.3 Integrity Controls

Integrity controls protect data from unauthorized use and update. Often integrity controls limit the values a field may hold, limit the actions that can be performed on data , or trigger the execution of some procedure, such as placing an entry in a log to record which users have done what which data.

One form of integrity control is a domain. In essence, a domain is a way to create a user-defined data type. Once a domain is defined, any field can be given that domain as its data type. For example, the following Price Change domain can be used as the data type of any database field, such as PriceIncrease and PriceDiscount, to limit the amounts standard process can be augmented in one transaction:

```
CREATE DOMAIN PriceChange AS DECIMAL

CHECK(VALUE BETWEEN, ,001 AND. 15);
```

Then, in the definition of, say, apricing transaction table, we might have

```
PriceIncrease PriceChange NOT NULL,
```

One advantage of a domain is that, if it ever has to change, it can be changed in one place-the domain definition- and all fields with this domain will be changed automatically. Alternatively, the same CHECK clause could be included in a constraint on both the PriceIncrease and PriceDiscount fields, but in this case if the limits of the check were to change, a DBA would have to find every instance of this integrity control and change it in each place separately.

Assertions are powerful constraints that enforce certain desirable database conditions. Assertions are checked automatically by the DBMS when transactions are run involving tables of fields on which assertions exist. For example, assume that an employee table has fields of EmpID, EmpName, SupervisiorID, and SpouseID. Suppose that a company rule is that no employee may supervise his or her spouse. The following assertion enforces this rule.

```
CREATE ASSERTION SpouseSupervision

Check(SupervisorID <> SpouseID);
```

If the assertion fails, the DBMS will generate an error message.

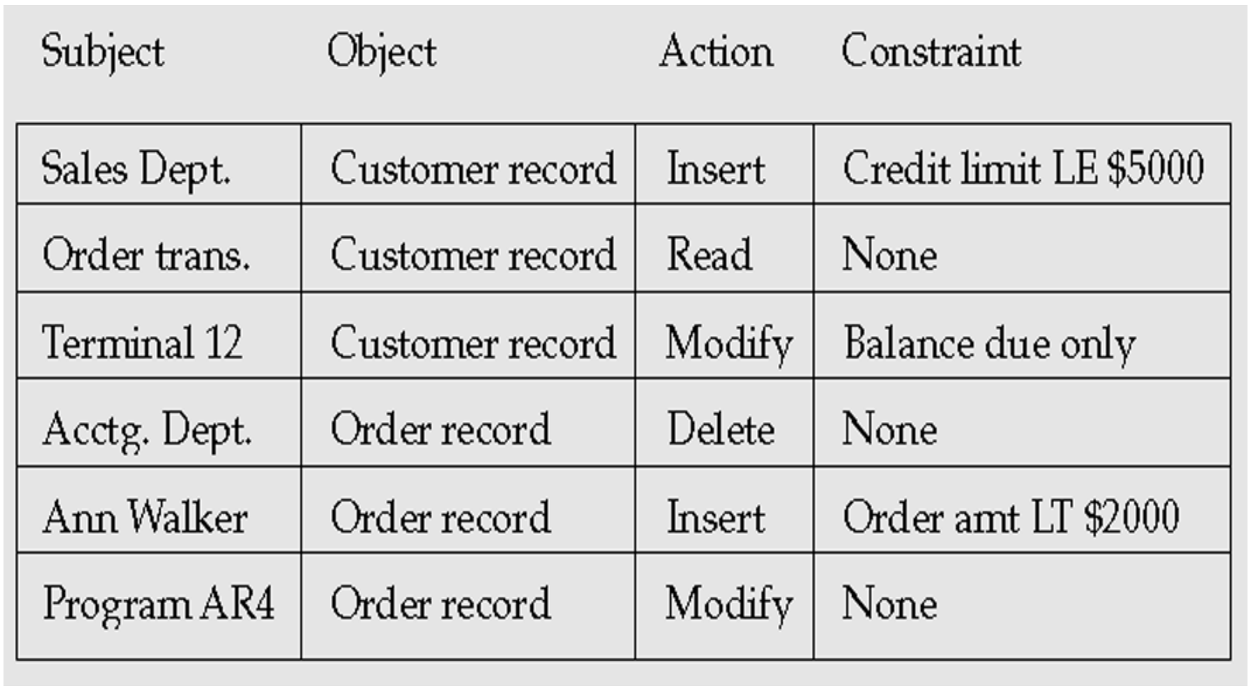
Assertions can become rather complex. Suppose that Pine Valley Furniture has a rule that no two salespersons can be assigned to the same territory at the same time. Suppose a Salesperson table includes fields of SalespersonID and TerritoryID. This assertion can be written using a correlated subquery as

```
CREATE ASSERTION Territory Assignment
CHECK(NOT EXISTS
(SELECT * FROM Salesperson SP WHERE SP.TerritoryID IN
SELECT SPP.TerritoryID FROM Salesperson SSP WHERE
SSP.SalespersonID<>SP.SallespersonID)));
```

Finally, trigger can be used for security purposes. A trigger, which includes an event, condition, and action, is potentially more complex than an assertion. For example, a trigger can do the following:

- Prohibit inappropriate actions(e.g., changing a salary value outside of the normal business day.
- Cause special handling procedures to be executed(e.g., if a customer invoice payment is received after some due data, a penalty can be added to the account balance for that customer).
- Cause a row to be written to a log file to echo important information about the user and a transaction being made to sensitive data, so that the log can be reviewed by human or automated procedures for possible inappropriate behavior (e.g., the log can record which user initiated a salary change for which employee)

As with domains, a powerful benefit of a trigger, like any stored procedure, is that the DBMS enforces these controls for all users and all database activities. The control does not have to be coded into each query or program. Thus, individual users and programs can not circumvent the necessary controls.



Subject	Object	Action	Constraint
Sales Dept.	Customer record	Insert	Credit limit LE $5000
Order trans.	Customer record	Read	None
Terminal 12	Customer record	Modify	Balance due only
Acctg. Dept.	Order record	Delete	None
Ann Walker	Order record	Insert	Order amt LT $2000
Program AR4	Order record	Modify	None

Figure 15.3 Authorization matrix

15.2.4 Authorization Rules

Authorization Rules are controls incorporated in the data management system that restrict access to data and also restrict the actions that people may take hen they access data. For example, a person who can supply a particular password may be authorized to read any record in a database but cannot necessary modify any of those records.

Fernandez, Summers, and Wood(1981) have developed a conceptual model of database security. Their model expresses authorization rules in the form of a table (or matrix) that includes subjects, objects, actions, and constraints. Each row of the table indicates that a particular subject is authorized to take a certain action on an object in the database, perhaps subject to some constraint. figure 15.3 shows an example of such an authorization matrix. This table contains several entries pertaining to records in an accounting database.

For example, the first row in the table indicates that anyone in the sales Department is authorized is authorized to insert a new customer record in the data base, provided that the customer's credit limit does not exceed $5,000. The last row indicates that the program AR4 is authorized to modify order records without restriction. Data administration is responsible for determining and implementing authorization rules that are implemented at the database level. Authorization schemes can also be implemented at the operating system level or the application level.

	Customer records	Order records
Read	Y	Y
Insert	Y	Y
Modify	Y	N
Delete	N	N

Figure 15.4 Implementation rules (a) Authorization table for subjects (salespersons)

Accounting (password TRACY)
Y
N
Y
Y

ler records)

Privilege	Capability
SELECT	Query the object.
INSERT	Insert records into the table/view. Can be given for specific columns.
UPDATE	Update records in table/view. Can be given for specific columns.
DELETE	Delete records from table/view.
ALTER	Alter the table.
INDEX	Create indexes on the table.
REFERENCES	Create foreign keys that reference the table.
EXECUTE	Execute the procedure, package, or function.

ot implement an authorization
plified versions. There are two
tables for objects. Figure 15.4
see that salespersons are al-
these records. In Figure 15.4b, we see that users
ecords, but salespersons cannot. A given DBMS
ypes of facilities.

Figure 15.5 Oracle8i privileges

Authorization tables such as those shown in Figure 15.4 are attributes of an organization's data and their environment; they are therefore properly viewed as metadata. Thus, the tables should be stored and maintained in the repository. Since authorization tables contain highly sensitive data, they themselves should be protected by stringent security rules. Normally, only selected persons in data administration have authority to access and modify these tables.

For example, in Oracle8i, the privileges included in Figure 15.5 can be granted to users at the database or table level. INSERT and UPDATE can be granted at the column level. Where many users, such as those in a particular job classification, need similar privileges, roles may be created that contain a set of privileges, and then all of the privileges, can be granted t a user simply by granting the role. To grant the ability to read the product table and update prices to a user with the login ID of smith, the following SQL command may be given:

```
GRANT SELECT, UPDATE (unit_price) ON PRODUCT_T TO SMITH;
```

There are eight data dictionary views that contain information about privileges that have been granted. In this case DBA_TAB_PRIVS contains users and objects for every user who has been granted privileges on objects, such as tables. DBA_COL_PRIVS contains users who have been granted privileges on columns of tables.

15.2.5 User-Defines Procedures

Some DBMS products provide user exits (or interfaces) that allow system designers or users to create their own user-defined procedures for security, in addition to the authorization rules we have just described.

For example, a user procedure might be designed to provide positive user identification. In attempting to log on to the computer, the user might be required to supply a procedure name in addition to a simple password. If valid password and procedure names are supplied, the system then calls the procedure, which asks the user a series of questions whose answers should be known only to that password holder (such as mother's maiden name).

15.2.6 Encryption

For highly sensitive data such as company financial data, data encryption can be used. Encryption is the coding or scrambling of data so that humans cannot read them. Some DBMS products include encryption routines that automatically encode sensitive data when they are stored or transmitted over communications channels. For example, encryption is commonly used in electronic funds transfer (EFT) systems. Other DBMS products provide exits that allow users to code their own encryption routines.

Any system that provides encryption facilities must also provide complementary routines for decoding the data. These decoding routines must be protected by adequate security, or else the advantages of encryption are lost, and they also require significant computing resources.

Two common forms of encryption exist; one key and two key. With a *one-key* method, also called data encryption standard (DES), both the sender and the receiver need to know the key that is used to scramble the transmitted or stored data. A *two-key* method, also called a duel-key method, public-key method, or asymmetric encryption, employs a private and a public key. Two-key methods are especially popular in electronic commerce applications to provide secure transmission and database storage of payment data, such as credit card numbers. With this method, all users know each other's public keys, but only each individual knows the private key. A first-time user is assigned the pair of keys by the system of trust, for example, a DBMS or security system working with the DBMS. A message (such as a transaction to update the database) is encrypted using the sender's private key and the receiver's public key; this message can be decrypted using only the receiver's private key and the sender's public key. The sender and receiver may be two users (or user programs) or the DBMS and a user (or user program). A two-key method depends on the public key directory being secure and not able to be corrupted. Thus, the public key must be maintained in a highly secure environment, possibly by a third parity. Companies such as Verisign exist for the purpose of maintaining public keys and certifying new keys. A variation of the two-key method is digital signatures, which provide evidence, formalism, approval, efficiency, and authentication for electronic commerce (see Web Resource http://abanet.org/scitech/ec/isc/deg-turotial.html for an excellent tutorial on digital signatures and public key certificates).

15.2.7 Authentication Schemes

A long-standing problem in computer circles is how to positively identify persons who are trying to gain access to a computer or its resources. The first line of defense is the use of passwords, which provides some protection. However, people assigned passwords for different devices quickly devise ways to remember these passwords, ways that tend to decrease the effectiveness of the password scheme. The passwords get written down, where others may find them. They get shared with other users; it is not unusual for an entire department to use one common password for access. Passwords get included in automatic logon scripts, which removes the inconvenience of remembering them and typing them but also eliminates their effectiveness. And passwords usually traverse a network in cleartext, not encrypted, so if intercepted they may be easily interpreted. Also, passwords cannot, of themselves, ensure the security of a computer and its databases, because they give no indication of who is trying gain access. The current distributed and networked environments require additional security measures. Different approaches and combinations of approaches are being tried to deal with this problem.

First, industry has developed devices and techniques to positively identify any prospective user. The most promising of these appear to be biometric devices, which measure or detect personal characteristics such as fingerprints, voiceprints, eye pictures, or signature dynamics. The use of retina prints has declined because of concerns about possible damaging effects to the eye by the laser optics used to capture a retina print. To implement the biometric approach, several companies have developed a smart card that embeds an individual's unique biometric data (such as fingerprints or hand geometry) permanently on the card. To access a computer, the user inserts the card into a reader device (a biometric device) that reads the person's fingerprint or other characteristic. The actual biometric data are then compared with data in a database, and the two must match for the user to gain computer access. A lost or stolen card would be useless to another person, since the biometric data would not match. Such a smart cared is useful with an ATM, credit card purchasing, and even electronic commerce.

Another approach is the use of third-party mediated authentication systems, which establish user authenticity through a trusted authentication agent, such as Kerberos. Developed at MIT, Kerberos is primarily used in application-level protocols, such as TELNET or FTP, to provide user-to-host security. Kerberos works by providing a secret key (Kerberos ticket) to a user that can then be embedded in any other network protocol. Any process which implements that protocol can then be certain of the source of the request. Sun Microsystems also has an authentication mechanism, called DES Authentication, which is based on the sender encrypting a time-stamp when the message is sent, which is then checked against the receiver's internal clock. This requires agreement between the agents as to the current time and both must be using the same encryption key. Liability issues have kept third-party authentication schemes from achieving widespread acceptance. Public key, certificate-based authentication schemes which issue to parties identification certificates that they can then exchange without further involvement of a third party are also being used. Such digital certificates will be used extensively in electronic commerce transaction involving credit card or digital cash purchases.

A last piece of the authentication problem is establishing nonrepudiation. That is after a message has been sent, the user should not be able to repudiate having sent the message, or say that it was not sent. Biometric devices coupled with the sending of messages are used to establish nonrepudiation.

15.3 Backing Up Databases

Database recovery is data administration's response to Murphy's law. Inevitably, databases are damaged or lost because of some system problem that may be caused by human error, hardware failure, incorrect or invalid data, program errors, computer viruses, network failures, conflicting transactions, or natural catastrophes. Since the organization depends to heavily on its database, the database management system must provide mechanisms for restoring a database quickly and accurately after loss or damage.

15.3.1 Basic Recovery Facilities

A database management system should provide four basic facilities for backup and recovery of a database:

- Backup facilities, which provide periodic backup copies of portions of or the entire database.
- Journalizing facilities, which maintain an audit trail of transactions and database changes.
- A checkpoint facility, by which the DBMS periodically suspends all processing and synchronizes its files and journals.
- A recovery manager, which allows the DBMS to a correct condition and restart processing transactions

Backup Facilities The DBMS should provide backup facilities that produce a backup copy (or save) of the entire database plus control files and journals. Typically, a backup copy is produced at least once per day. The copy should be stored in a secured location where it is protected from loss or damage. The backup copy is used to restore the database in the event of hardware failure, catastrophic loss, or damage. Some DBMSs provide backup utilities for the DBA to use to make backups; other systems assume the DBA will use the operating system commands, export commands, or SELECT … INTO SQL commands to perform backups. Since performing the nightly backup for a particular database is repetitive, creating a script that automates regular backups will save time and result in fewer backup errors.

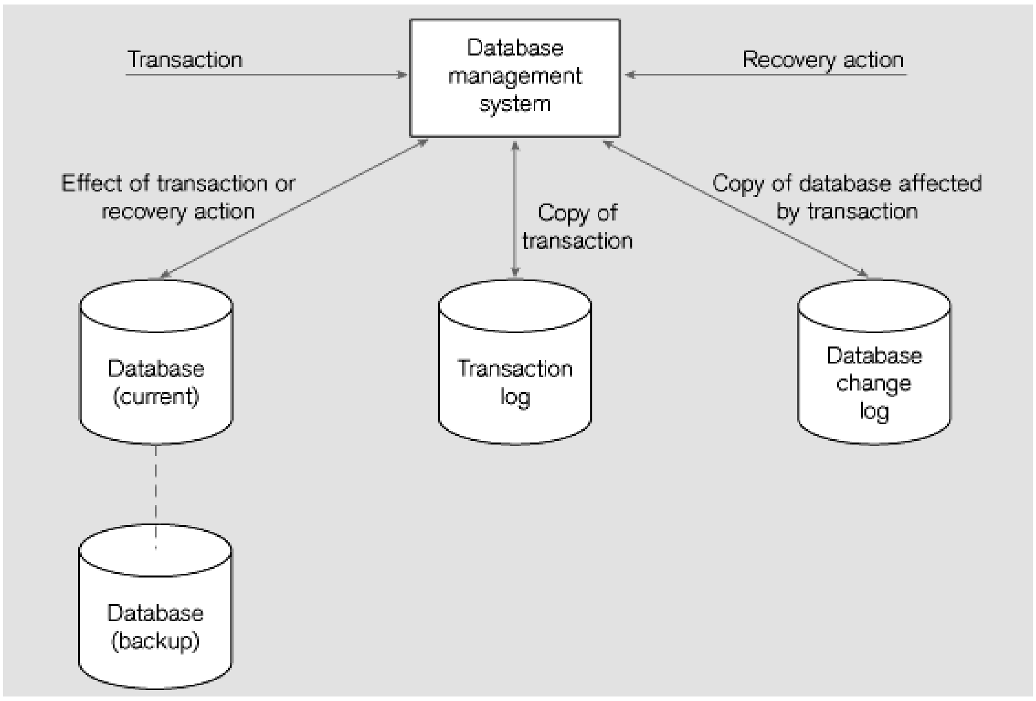
With large databases, regular backups may be impractical, as the time required to perform the backup may exceed that available. Or, a database may be a critical system that must always remain available; so, a cold backup, where the database is shut down, is not practical. As a result, backups may be taken of dynamic data regularly (a so-called hot backup in which only a selected portion of the database is shut down from use) but backups of static data, which don't change frequently, may be taken less often. Incremental backups, which record changes made since the last full backup, but which do not take so much time to complete, may also be taken on an interim basis, allowing for longer periods of time between full backups. Determining backup strategies, thus, must be based on the demands being placed on the database systems.

Journalizing Facilities A DBMS must provide journalizing facilities to produce an audit trail of transactions and database changes. In the event of a failure, a consistent database state can be reestablished using the information in the journals together with the most recent complete backup. As Figure 15.6 shows, there are two basic journals, or logs. The first is the transaction log, which contains a record f the essential data for each transaction that is processed against the database. Data that are typically recorded for each transaction include the transaction code or identification, action or type of transaction (e.g., insert), time of the transaction, terminal number or user ID, input data values, table and records accessed, records modified, and possibly the old and new field values.

The second kind of log is the database change log, which contains before- and after-images of records that have been modified by transactions. A before-image is simply a copy of a record before it has been modified, and an after-image is a copy of the same record after it has been modified.

Some systems also keep a security log, which can alert the DBA to any security violation that occur or are attempted.

Figure 15.6 Database audit trail



Checkpoint Facility A checkpoint facility in a DBMS periodically refuses to accept any new transactions. All transactions in progress are completed, and the journal files are brought up to date. AT this point, the system is in a quiet state, and the database and transaction logs are synchronized. The DBMS writes a special record (called a checkpoint record) to the log file, which is like a snapshot of the state of the database. The checkpoint record contains information necessary to restart the system. Any dirty data blocks (pages of memory that contain changes that have not yet been written out to disk) are written from memory to disk storage, thus ensuring that all changes made prior to taking the checkpoint have been written to long-term storage.

A DBMS may perform checkpoints automatically (which is preferred) or in response to commands in user application programs. Checkpoints should be taken frequently (say, several times an hour). When failures do occur, it is often possible to resume processing from the most recent checkpoint. Thus, only a few minutes of processing work must be repeated, compared with several hours for a complete restart of the day's processing.

Recovery Manager The recovery manager is a module of the DBMS which restores the database to a correct condition when a failure occurs and which resumes processing user requests. The type of restart used depends on the nature of the failure. The recovery manager uses the logs shown in figure 15.6 to restore the database.

15.3.2 Recovery and Restart Procedures

The type of recovery procedure that is used in a given situation depends on the nature of the failure, the sophistication of the DBMS recovery facilities, and operational policies and procedures. Following is a discussion of the techniques that are most frequency used.

Switch In order to be able to be able to switch to an existing copy of the database, the database must be mirrored. That is, at least two copies of the database must be kept and updated simultaneously. When a failure occurs, processing is switched to the duplicate copy of the database. This strategy allows for the fastest recovery and has become increasingly popular as the cost of long-term storage has dropped. Level 1 RAID systems implement mirroring. Rather than being stored on a large drive, the database is distributed across several smaller, less expensive disks and mirrored to a second set of such disks. When a disk failure occurs, the system immediately switches to the mirrored disk. The defective or damaged disk can then be removed and a new disk put in place. That disk can be rebuilt from the mirrored disk with no disruption in service to the user. Such disks are referred to as being hot-swappable. This strategy does not protect against loss of power or catastrophic damage to both databases, though.

Restore/Rerun The restore/rerun technique involves reprocessing the day's transactions (up to the point of failure) against the backup copy of the database or portion of the database being recovered. First, the database is shut down and then the most recent copy of the database or file to be recovered (say, from the previous day) is mounted, and all transactions that have occurred since that copy (which are stored on the transaction log) are rerun. This may also be a good time to make a backup copy and clear out the transaction, or redo, log.

The advantage of restore/rerun is its simplicity. The DBMS does not need to create a database change journal, and no special restart procedures are required. However, there are two major disadvantages. First, the time to reprocess transactions may be prohibitive. Depending on the frequency of making backup copies, several hours of reprocessing may be required. Processing new transactions will have to be deferred until recovery is completed, and if the system is heavily loaded, it may be impossible to catch up. ?The second disadvantage is that the sequencing of transactions will often be different from when they were originally processed, which may lead to quite different results.

Transaction Integrity A database is updates by processing transactions that result in change to one or more database records.

Is an error occurs during the processing of a transaction, the database may be compromised, and some from of database recovery is required. Thus, to understand database recovery, we must first understand the concept of transaction integrity.

A business transaction is a sequence of steps that constitute some well-defined business activity. Examples of business transactions are " Admit Patient" in a hospital and "Enter Customer Order" in a manufacturing company. Normally, a business transaction requires several actions against the database. For example, consider the transaction " Enter Customer Order." When a new customer order is entered, the following steps may be performed by an application program:

- Input order data
- Read Customer record
- Accept or reject the order. Id balance Due plus Order Amount does not exceed Credit Limit, accept the order, otherwise, reject it.
- If the order is accepted, Increase Balance Due by order Amount. Store the updated Customer record. Insert the accepted ORDER record in the data base.

When processing transactions, the DBMS must ensure that the transactions follow four well-accepted properties, called the ACID properties.

Atomic, meaning that the transaction cannot be subdivided, and hence, it must be processed in its entirety or not at all. Once the whole transaction is processed, we say that the changes are committed. If the transaction fails at any midpoint, we say that it has aborted.

Consistent, Meaning any database constraints that must be true before the transaction must also be true after the transaction.

Isolated, Meaning changes to the database are not revealed to users until the transaction is committed.

Durable, meaning changes are permanent. Thus , once a transaction is committed, no subsequent failure of the database can reverse the effect of the transaction..

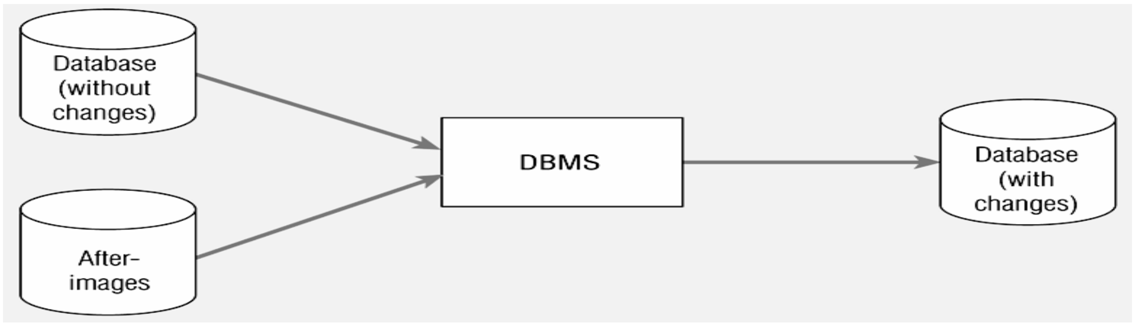
Transaction boundaries: The logical beginning and end of transactions.

Backward recovery (rollback): The back out, undo, of unwanted changes to the database. Before image of the records that have been changed are applied to the database, and the database is returned to an earlier state. Used to reverse the changes made by transactions that have been aborted or terminated abnormally.

Backward Recovery with backward recovery (also called rollback), the DBMS backs out of or undoes unwanted changes to the database. As Figure 15.7a shows before-images of the records that have been changed are applied to the database.

Figure 15.7 Basic recovery techniques
(a) Rollback

(b) Rollforward



As a result, the database is returned to an earlier state; the unwanted changes are eliminated.

Backward recovery is used to reverse the changes made by transactions that have aborted, or terminated abnormally.

To illustrate the need for backward recovery, suppose that a banking transaction will transfer $100 in funds from the account for customer A to the account for customer B. These steps are performed:

- ☞ The program reads the record for customer A and subtracts $100 from the account balance.
- ☞ The program then reads the record for customer B and adds $100 to the account balance. Now the program writes the updates record for customer A to the database. However, in attempting to write the record for customer B, the program encounters an error condition and cannot write the record. Now the database is inconsistent- record A has been updated but record B has not-and the transaction must be aborted. An UNDO command will cause the recovery manager to apply the before image for record A to restore the account balance to its original value .

Forward recovery (rollforward) : A technique that starts with an earlier copy of the database.

After-images are applied to the database, and the database is quickly moved forward to a later state.

Forward recovery with forward recovery (also called rollforward), the DBMS starts with an earlier copy of the database. Applying after-images quickly moves the database forward to a later state(see the Figure 15.7b).

Forward recovery is much faster and more accurate than restore /return, for the following reasons.

1. the time-consuming logic of reprocessing each transaction does not have to be repeated.
2. Only the most recent after-images need to be applied. A database record may have a series of after-images, but only the most recent, " good" after-image is required for rollforward.

The problem of different sequencing of transactions is avoided, since the results of applying the transactions are used.

15.3.3 Types of Database Failure

A wide variety of failure can occur in processing a database, ranging from the input of an incorrect data value to complete loss of destruction of the database. Four of the most common types of problems are aborted transactions, incorrect data, system failure, and database loss or destruction. Each of these types of problems is described in the following sections, and possible recovery procedures are indicated(see the Table 15.1).

- ***Aborted transactions***
  - Preferred recovery: rollback
  - Alternative: Rollforward to state just prior to abort
- ***Incorrect data***
  - Preferred recovery: rollback
  - Alternative 1: re-run transactions not including inaccurate data updates
  - Alternative 2: compensating transactions
- ***System failure (database intact)***
  - Preferred recovery: switch to duplicate database
  - Alternative 1: rollback
  - Alternative 2: restart from checkpoint
- ***Database destruction***
  - Preferred recovery: switch to duplicate database
  - Alternative 1: rollforward
  - Alternative 2: reprocess transactions

15.4 Summary

Threats to data security include accidental losses, theft and fraud, loss of privacy; loss of data integrity; and loss of availability. A comprehensive data security plan will address all of these potential threats, partly through the establishment of views, authorization rules, user-defined procedures, and encryption procedures.

15.5 Technical Terms

Database Security: Protection of the data against accidental or intentional loss, destruction, or misuse.

Authorization rules: Controls incorporated in the data management systems that restrict access to data and also restrict the actions that people may take when they access data.

User-defined procedures: User exits that allows system designers to define their own security procedures in addition to the authorization rules.

Encryption: The coding or scrambling of data so that humans cannot read them.

Biometric device: Techniques that measures or detect personal characteristics such as fingerprints, voiceprints, eye pictures, or signature dynamics.

Database recovery: Mechanisms for restoring a database quickly and accurately after loss or damage.

Backup facilities: An automatic dump facility that procedures a backup copy of the entire database.

Journalizing facilities: An audit trail of transactions and database changes.

Transaction: A discrete unit of work that must be completely processes or not processed at all within a computer system. Entering a customer order is an example of a transaction.

Transaction log: A record of the essential data for each transaction that is processed against the database.

Database change log: Before and after images of records that have been modified by transactions.

Before-images: A copy of a record before it has been modified.

After-images: A copy of a record after it has been modified.

Checkpoint facility: A facility by which the DBMS periodically refuses to accept any new transactions. The system is in a quiet state, and the database and transaction logs are synchronized.

Recovery manager: QA module of the DBMS which restores the database to a correct condition when a failure occurs and which resumes processing user requests.

Restore/return: A technique that involves reprocessing the days transactions against the backup copy of the database.

Transaction boundaries: The logical beginning and end of transaction.

Aborted Transaction: A transaction in progress that terminates abnormally.

15.6 Self Assessment Questions

1. List and discuss five areas where threats to data security may occur.
2. Explain how creating a view may increase data security . Also explain why one should not rely completely on using views to enforce data security.
3. List and briefly explain how integrity controls can be used for database security.
4. What is the difference between an authentication scheme and an authorization scheme?
5. Briefly describe four DBMS facilities that are required for database backup and recovery.
6. What is transaction integrity? Why is it important?
7. List and describe four common types of database failure.

15.7 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe
(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.
Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- C.V.P.R.Prasad, M.C.A

Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.

Lesson– 16

DATA AND DATABASE ADMINISTRATION -III

16.0 Objective

To be able to

- ☞ Compare the optimistic and pessimistic systems of concurrency control.
- ☞ Describe the problem of tuning a database to achieve better performance, and areas where changes may be made when tuning a database.

Structure

16.1 Introduction

Databases are shared resources. Database administrators must expect and plan for the likelihood that several users will attempt to access and manipulate data at the same time. With concurrent processing involving updates, a database without concurrency control will be compromised due to interference between users. There are two basic approaches to concurrency control: a pessimistic approach and optimistic approach.

16.2 Controlling Concurrent Access

Concurrency Control :The process of managing simultaneous operations against a database so that data integrity is maintained and the operations do not interfere with each other in a multi-user environment.

16.2.1 The problem of Lost Updates

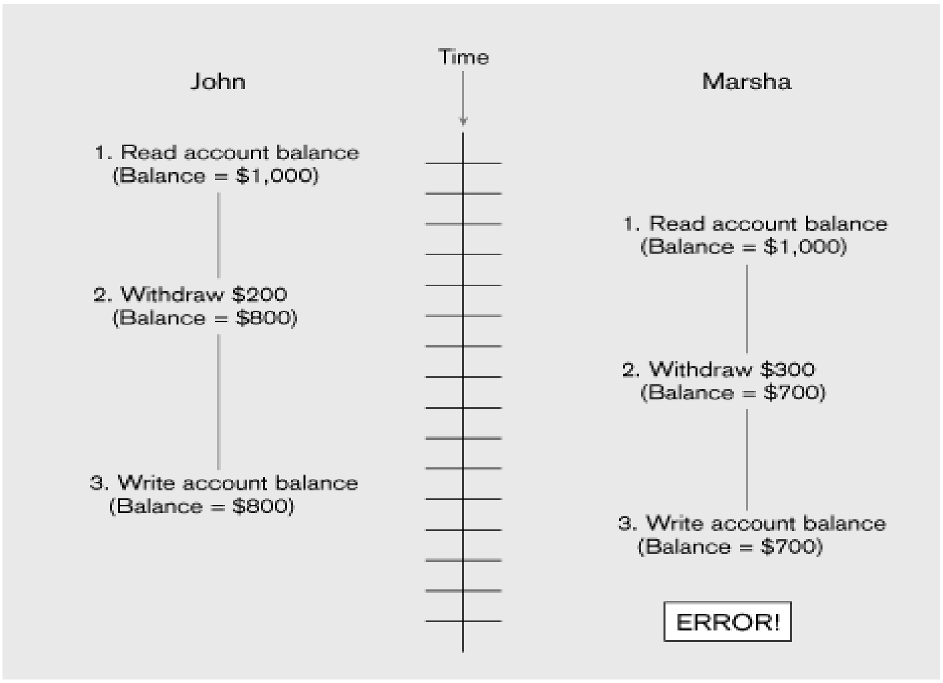
The most common problem encountered when multiple users attempt to update a database without adequate concurrency control is that of lost updates Figure 16,8 shows a common situation. John and Marha have a joint checking account and both want to withdraw some cash at the same time, each using an ATM terminal in a different location. Figure 16.8 shows the sequence of events that might occur, in the absence of a concurrency control mechanism. John's transaction reads the account balance (which is $1,000) and he proceeds to withdraw $200. Before the transaction write the new account balance ($800), Marasha's transaction reads the account balance (which is still $1,000). She then withdraws 300, leaving a balance of $700. Her transaction then writes this account balance, which replaces the one write by John's transaction. The effect of John's update has been lost due to interface between the transactions, and the bank is unhappy.

Another, similar type of problem that may occur when concurrency control is not established is the inconsistent read problem.

Inconsistent read Problem: An unrepeatable read, once that occurs when one user reads data that have been partially updated by another user.

This problem occurs when one user reads data that have been partially updated by another user. The read will be incorrect, and is sometimes referred to as a dirty read or an unrepeatable read. The lost updates and inconsistent read problems arise when the DBMS does not isolate transaction, part of the ACID transaction properties.

Figure 16.8 Lost update (no concurrency control in effect)


Time
John
Marsha
1. Read account balance (Balance = $1,000)
2. Withdraw $200 (Balance = $800)
3. Write account balance (Balance = $800)
1. Read account balance (Balance = $1,000)
2. Withdraw $300 (Balance = $700)
3. Write account balance (Balance = $700)
ERROR!

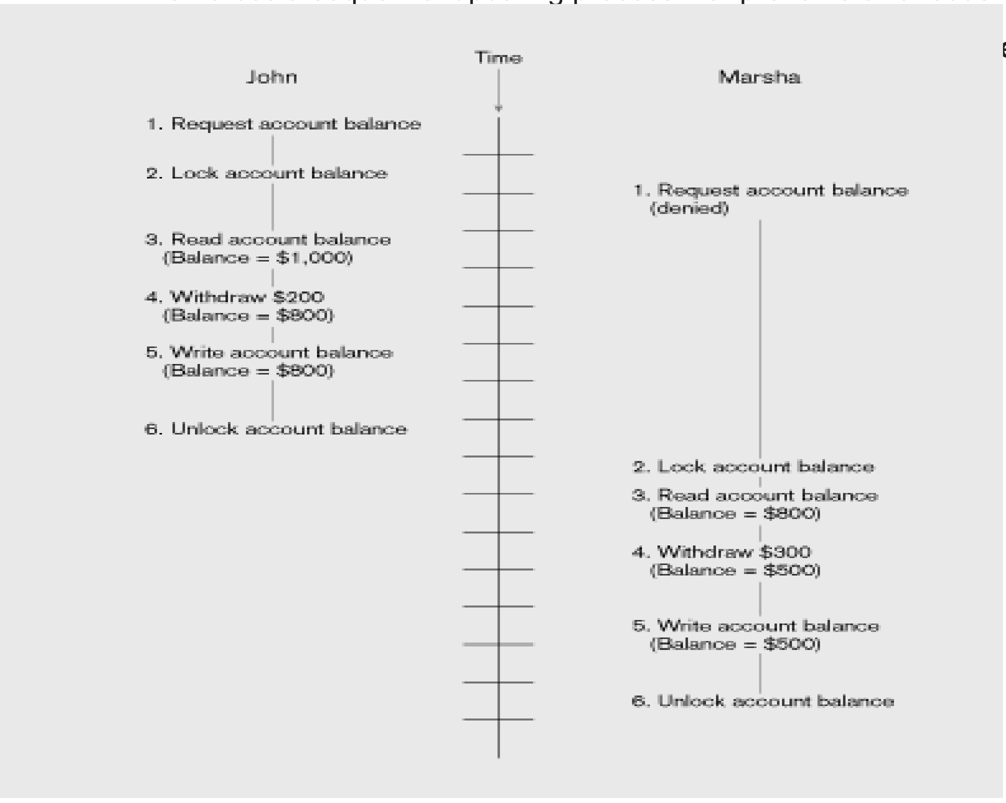
16.2.2 Serializability

Concurrent transactions need to be processed in isolation so that they do not interfere with each other; If one transaction were entirely processed at a time before another transaction, no interference would occur. Procedures that process transactions so that the outcome is the same as this are called serializable. Processing transactions using a serializable schedule will give the same results as if the transactions had been processed one after the other. Schedules are designed so that transactions that will not interfere with each other can still be run in parallel.

16.2.3 Locking Mechanisms

- The most common way of achieving serialization
- Data that is retrieved for the purpose of updating is locked for the updater
- No other user can perform update until unlocked

Figure 16.9 shows the use of record locks to maintain data integrity. John initiates a withdrawal transaction from an ATM. Since John's transaction will update this record, the application program locks this record before reading it into main memory. John proceeds to withdraw $200, and the new balance ($800) is computed. Marsha has initiated a withdrawal transaction shortly after John, but her transaction cannot access the account record until John's transaction has returned the updated record to the database and unlocked the record. The locking mechanism thus enforces a sequential updating process that prevents erroneous updates.


John
Time
Marsha
1. Request account balance
2. Lock account balance
3. Read account balance (Balance = $1,000)
4. Withdraw $200 (Balance = $800)
5. Write account balance (Balance = $800)
6. Unlock account balance
1. Request account balance (denied)
2. Lock account balance
3. Read account balance (Balance = $800)
4. Withdraw $300 (Balance = $500)
5. Write account balance (Balance = $500)
6. Unlock account balance

ency control)

Locking Level : an important consideration in implementing concurrency control is choosing the locking level. The locking level is the extent of the database resource that us included with each lock. Most commercial products implement locks at one of the following levels.

Database – used during database updates

Table – used for bulk updates

Block or page – very commonly used

Record – only requested row; fairly commonly used

Field – requires significant overhead; impractical

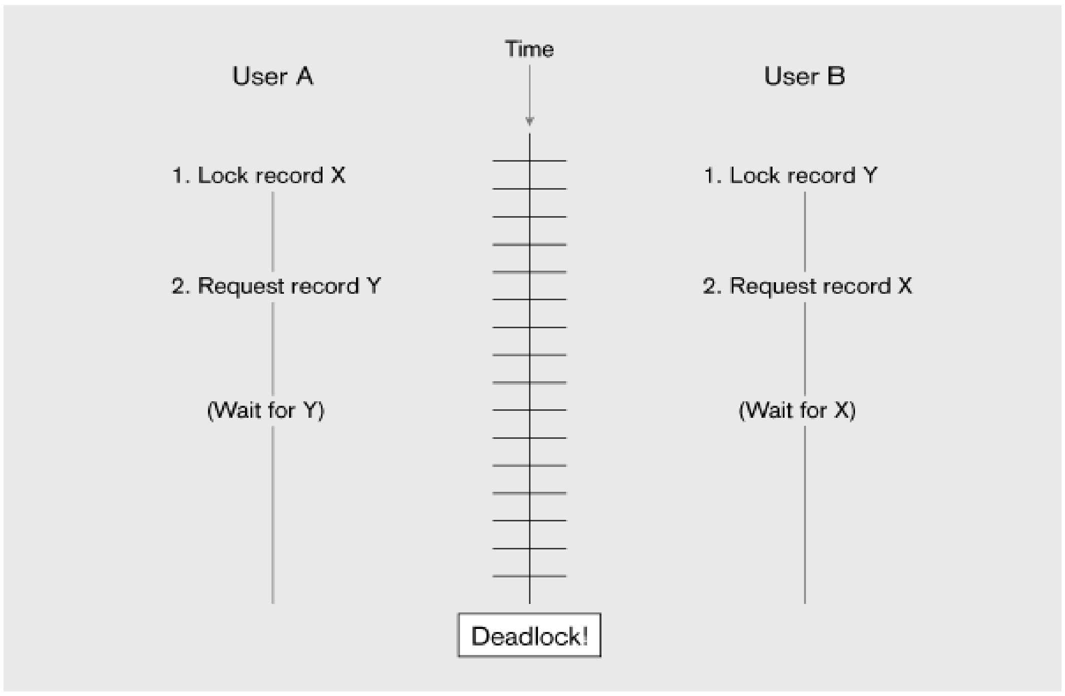
Types of locks: So , far we have discussed only locks that prevent all access to locked items. In reality, the database administrator can generally choose between two types of locks: shared and exclusive.

1. **Shared lock** - Read but no update permitted. Used when just reading to prevent another user from placing an exclusive lock on the record
2. **Exclusive lock** - No access permitted. Used when preparing to update

Deadlock : An impasse that results when two or more transactions have locked common resources, and each waits for the other to unlock their resources.

figure 16-11 shows a slightly more complex example of deadlock. In this example, User A has locked X and User B has locked record Y. User A then requests record Y amd User B requests record X. Both Requests are denied, since the requested records are already locked. Unless the DBMS intervenes, both users will wait indefinitely.

Figure 16.11 Another example of deadlock



Managing Deadlock

There are two basic ways to resolve deadlocks.

Deadlock prevention: Lock all records required at the beginning of a transaction

Two-phase locking protocol : A procedure for acquiring the necessary locks for a transaction where all necessary locks are acquired before any locks are released, resulting in a growing phase, when locks are acquired, and a shrinking phase, when they are released.

May be difficult to determine all needed resources in advance

Deadlock Resolution: An approach that allows deadlocks to occur but builds mechanisms into the DBMS for detecting and breaking them deadlocks.

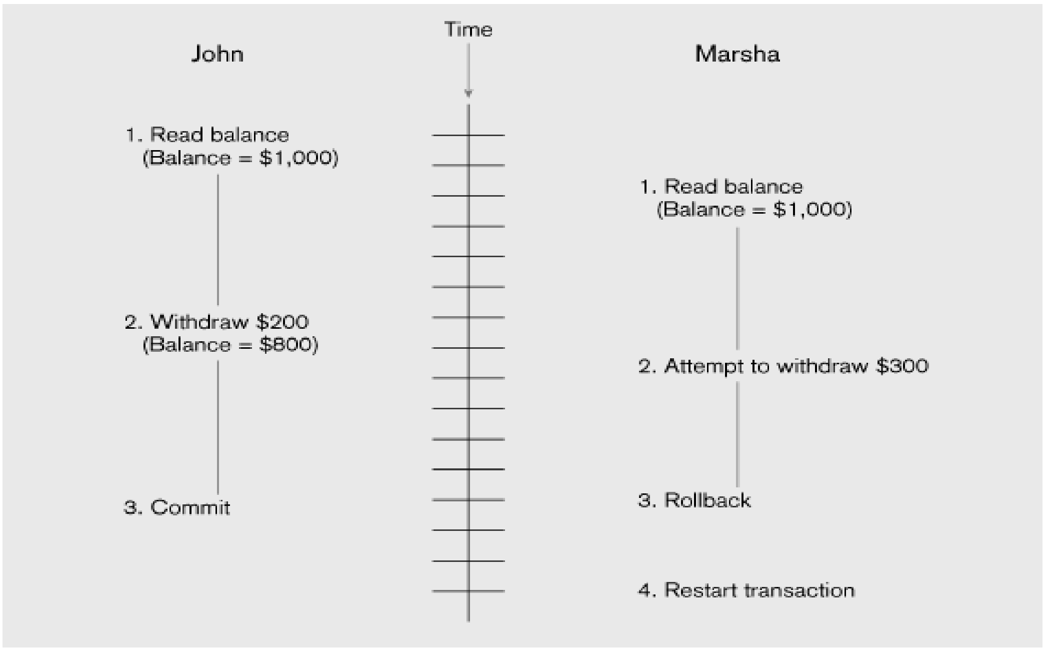
16.2.4 Versioning

Each transaction is restricted to a view of the database as of the time that transaction started, and when a transaction modifies a record, the DBMS creates a new record version instead of overwriting the old record . Hence no form of locking is required.

Figure 16.16 shows a simple example of the use of versioning for the checking account example. John reads the record containing the account balance, successfully withdraws $200, and the new balance ($800) is posted to the account with a COMMIT statement. Meanwhile, Marsha has also read the account record and requested a withdrawal, which is posted to her local version of the account record. However, when the transaction attempts to COMMIT it discovers the up- dates conflict. And her transaction is aborted. She can then restart the transaction, working from the correct starting balance of $800.

The main advantage of versioning over locking is performance improvement Read-only transaction can run concurrently with updating transactions, without loss of database consistency.

Figure 16.16 The use of Versioning


Time
John
Marsha
1. Read balance (Balance = $1,000)
2. Withdraw $200 (Balance = $800)
3. Commit
1. Read balance (Balance = $1,000)
2. Attempt to withdraw $300
3. Rollback
4. Restart transaction

16.3 Managing Data Quality

High quality data, that is , data that are accurate, consistent, and available in a timely fashion are essential to the management of organizations today. Organizations must strive to identify the data are relevant to their decision making, to develop business policies and practices that ensure the accuracy and completeness of the data, and to facilities enterprise-wide data sharing.

Data Steward: A person assigned the responsibility of ensuring that organizational applications properly support the organization's enterprise goals

Five Data Quality Issues:

1. Security policy and disaster recovery
2. Personnel controls
3. Physical access controls
4. Maintenance controls (hardware & software)
5. Data protection and privacy

16.3.1 Security Policy and Disaster Recovery

Every organization, no matter its size, should establish comprehensive security policies and make detailed disaster recovery plans. The security policies will determine how the organization is going to maintain a secure system.

Responsibilities and duties of the employees with regard to those polices should be established, and the consequences should the policies not be followed stated.

Personnel Controls.

Contingency plans must consider data, equipment, personnel, and management issues involved in transferring operations to another location and safeguarding operations. Copies of the plans should be kept in multiple locations and reviewed periodically.

Larger organizations may arrange for space that can be used in the event of a disaster, either in the same general physical area, or in an area remote from the organization site being covered.

16.3.2 Personnel Controls

Adequate controls of personnel must be developed and followed, for the greatest threat to business security is often internal rather than external organizations should develop procedures to ensure a selective hiring process that validates potential employees' representations bout their backgrounds and capabilities. Monitoring to ensure that personnel are following established practices, taking regular vacations, working with other employees, and so forth, should be followed. Employees should be trained in those aspects of security and quality that are relevant to their jobs, and be encouraged to be aware of and follow standard security and data quality measures. Standard job controls, such as separating duties so no one employee has responsibility for an entire business process, or keeping application developers from having access to production systems, should also be enforced. Should an employee need to be let go, there should be an orderly set of procedures to remove authorizations and authentications and to notify other employees of the status change.

16.3.3 Physical Access Controls

Limiting access to particular areas within a building is usually a part of controlling physical access. Sensitive equipment, including hardware and peripherals, such as printers (which may be used to print classified reports) can be controlled by placement in secure areas. Other equipment may be locked to a desk or cabinet, or may have an alarm attached. Backup data tapes should be kept in fireproof data safes and/or kept off-site at a safe location. Procedures that make explicit the schedules for moving media and disposing of media, and that establish labeling and indexing of all materials stored, ad so forth, must be established.

Placement of computer screens so that they cannot be seen from outside the building may also be important. Control procedures for areas external to the office building should also be developed. Companies frequently use security guards to control access to their buildings, or use a card swipe system or handprint recognition system to automate employee access to the building. Visitors may be given an identification card and required to be accompanied throughout the building.

16.3.4 Maintenance Controls

An area of control that helps to maintain data quality but that is often overlooked is maintenance control. Organizations should review external maintenance agreements for all hardware and software they are using to ensure that appropriate response rates are agreed to for maintaining data quality.

It is also important to consider reaching agreements with the developers of all critical software so that organization can get access to source code should the developer go out of business or stop supporting the programs. One way to accomplish this is by having a third party hold the source code, with an agreement that it will be released if such a situation develops.

16.3.5 Data Protection and Privacy

Concerns about the rights of individuals to not have personal information collected and disseminated casually or recklessly have intensified as more of the population has become familiar with computers and as communications among computers have proliferated. Information privacy legislation generally gives individuals the right to know what data have been collected about them, and to correct any errors in those data. As the amount of data exchanged continues to grow, the need is also growing to develop adequate data protection. Also important are adequate provisions to allow the data to be used for legitimate legal purposes, so that organizations who need the data can access them and rely upon their quality.

16.4 Data dictionaries And Repositories

Data dictionary : A repository of information about a database that documents data elements of a database.

System catalog : A system-created database that describes all database objects, including data dictionary information, and also includes user access information.

16.4.1 Repositories

While the data dictionaries are simple data element document tools, information repositories are used by data administrators and other information specialists to manage the total information processing environment. The information repository is an essential component of both the development environment and the production environment. In the application development environment, people use CASE tools, high-level language, and other tools to develop new application. CASE tools may tie automatically to the information repository. In the production environment, people use applications to build databases, keep the data current, and extract data from databases. Each of these activities requires that the information repository be accessed and that it be current.

Information Repository : A component that stores metadata describing data and data processing resources, manages the total information processing environment, and combines information about an organization's business information and its application portfolio.

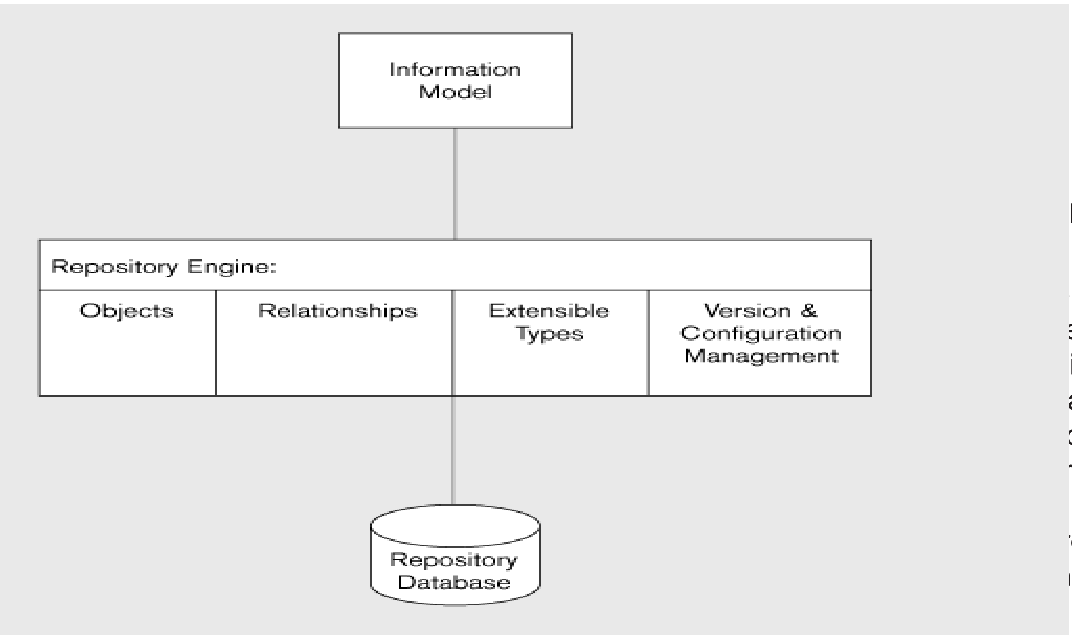
Information Repository Dictionary System (IRDS) : A computer Software tool managing/controlling access to information repository.

Figure 16.13 shows the three components of a repository system architecture (Bernstein, 1996). First is an information model. This model is a schema of the information stored in the repository, which can then be used by the tools associated with the database to interpret the contents of the repository. Next is the repository engine, which manages the repository objects. Services such as reading and writing repository objects, browsing, and extending the information model are included. Last is the repository database, where the repository objects are actually stored. Notice that the repository engine supports five core functions (Bernstein, 1996).

- ☞ *Object management* Object-oriented repositories store information about objects. As databases become more object-oriented, developers will be able to use the information stored about database objects in the information repository. The repository can be based on an object-oriented database or it can add the capability to support objects.
- ☞ *Relationship management* The repository engine contains information about object relationships that can be used to facilitate the use of software tools that attach to the database.
- ☞ *Dynamic extensibility* The repository information model defines types, which should be easy to extend, that is, to add new types or to extend the definitions of those that already exist. This capability can make it easier to integrate a new software tool into the development process.
- ☞ *Version management* During development it is important to establish version control. The information repository can be used to facilitate version control for software design tools. Version control of objects is more difficult to manage than version control of files, since there are many more objects than files in an application, and each version of an object may have many relationships.
- ☞ *Configuration management* It is also necessary to group the versioned objects into configurations that represent the entire system, which are also versioned. It may help you to think of a configuration as similar to a file directory, except configurations can be versioned and

they contain objects rather than files. Repositories often use checkout systems to manage objects, versions, and configurations. A developer who wishes to use an object checks it out, makes the desired changes, and then checks the object back in. At that time, a new version of the object will be created and the object will become available to other developers.

Figure 16.13 Three components of repository system architecture (Adapted from Bernstein, 1996)



mance

performance is not subject
re optimal performance is
e time of DBMS installation
isregarded. Rather, perfor-
ase, as hardware and soft-
of DBMS management that
re addressed here: installa-
and application tuning. The
eas will vary across DBMS
is section, but it should be

Tuning a database application requires familiarity with the system environment, the DBMS, the application, and the data used by the application. It is here that the skills of even an experienced database administrator are tested.

Achieving a quiet environment, one that is reliable and allows users to secure desired information in a time manner, requires skills and experience that are obtained by working with databases over time. The areas discussed below are quite general and are intended to provide an initial understanding of the scope of activities involved in tuning a database rather than providing the type of detailed understanding necessary to tune a particular database application.

16.5.1 Installation of the DBMS

Correct installation of the DBMS product is essential to any environment. Production often include README files, which may include detailed installation instruction, revisions of procedures, notification of increased disk space needed for installation, and so on. A quick review of any README files may time during the installation process and result in a better installation. General installation instruction should also be reviewed. Failing to make such a review may result in default parameter values being set during installation that are not optimal for the situation. Some possible considerations are listed here.

Before beginning installation, the database administration should ensure that adequate disk space is available. You will need to refer to translate logical database size parameters into actual physical space requirements. It is possible that the space allocation recommendations are low, as change made to a DBMS tend to make it larger, but the document may not reflect that change.

Allocation of disk space for the database should also receive consideration.

16.5.2 Memory Usage

Efficient usage of main memory involves understanding how the DBMS uses main memory, what buffers are being used, and what needs the programs in main memory have.

16.5.3 Input/Output Contention

Achieving maximum I/O performance is one of the most important aspects of tuning a database. Database applications are very I/O intensive—a production database will usually both read and write large amounts of data to disk as it works.

When hot spots, physical disk locations that are accessed repeatedly, develop, understanding the nature of the activity that is causing the hot spot affords the database administrator a much better chance of reducing the I/O contention being experienced. Oracle allows the DBA to control the placement of table spaces, which contain data files. An overall objective of distributing I/O activity evenly across disks and controllers should guide the DBA in tuning I/O.

16.5.4 CPU Usage

Most database operations are going to require CPU work activity. Because of this, it is important to evaluate CPU usage when tuning a database. Using multiple CPUs allows query processing to be shared when the CPUs are working in parallel, and performance may be dramatically improved. DBAs need to maximize the performance of their existing CPUs while planning for the gains that may be achieved with each new generations of CPUs.

Monitoring CPU load so that typical load throughout a 24-hour period is known provides DBAs with basic information necessary to begin to rebalance CPU loading. The mixture of online and background processing may need to be adjusted for each environment.

16.5.5 Application tuning

The Previous sections have concentrated on activities to tune the DBMS. Examining the applications that end users are using with the database may also increase performance. While normalization to at least 3NF is expected in many organizations that are using relation data models, Carefully planned demoralization may improve performance, often by reducing the number of tables that must be joined when running an SQL query.

Examination and modification of the SQL code in an application may also lead to performance improvement. Queries that do full table scans should be avoided.

Similarly, statements containing views and those containing sub quires should be actively reviewed, Tuning of such statements so that components are resolved in the most efficient manner possible may achieve significant performance gains.

The preceding brief description of potential areas where database performance may be affected should convince the reader of the importance of effective database management and tuning.

16.6 Summary

The problems of managing concurrent access in multi-user environments must also be addressed. The DBMS must ensure that database transactions possess the ACID properties: atomic, consistent, isolated, and durable. Proper transaction boundaries must be chosen to achieve these properties at an acceptable performance. If concurrency controls on transactions are not established, lost updates may occur; which will cause data integrity to be impaired. Locking mechanisms, including shared and exclusive locks, can be used. Deadlocks may also occur in multi-user environments, and may be managed by various means, including using a two-phase locking protocol or other deadlock resolution mechanism. Versioning is an optimistic approach to concurrency control.

There are five areas of concern when maintaining high data quality: security policy and disaster recovery, personnel controls, physical access controls, maintenance controls, and data protection and privacy concerns.

Managing the data dictionary, which is part of the system catalog in most relational database management systems, and the information repository help the DBA maintain high-quality data and high-performing database Dictionary System (IRDS) standard has helped with the development of repository information that can be integrated from multiple sources, including the DBMS itself, CASE tools, and software development tools.

Effective data administration is not easy and encompasses all of the areas summarized above. Increasing emphasis on object-oriented development methods and rapid development are changing the data administration function, but better tools to achieve effective administration and database tuning are becoming available.

16.7 Technical Terms

Concurrency Control: the process of managing simultaneous operations against a database so that data integrity is maintained and the operations do not interface with each other in a multi-user environment.

Inconsistent read problem: An unrepeatable read, one that occurs when one user reads data that have been partially updated by another user.

Locking: Any data that are retrieved by a user for updating must be locked, or defined to other users, until the update is completed or aborted.

Locking Level: The extent of the database resource that is included with each lock.

Shared locks: A technique that allows other transactions to read but not update a record or other resource.

Exclusive lock: A technique that prevents another transaction from reading and therefore updating a record until it is unlocked.

Deadlock: An impasse that results when two or more transactions have locked a common resource, and each waits for the other to unlock that resource.

Deadlock prevention: user programs must lock all records they required at the beginning of a transaction.

Two-phase locking protocol: A procedure for acquiring the necessary locks for a transaction where all necessary locks are acquired before any locks are released, resulting in a growing phase, when locks are acquired, and a shrinking phase, when they are related.

16.8 Self Assessment Questions

1. What is the advantage of optimistic concurrency control, compared with pessimistic concurrency control?
2. What is the difference between deadlock prevention and deadlock resolution?
3. What is the difference between shared locks and exclusive locks ?
4. What is the difference between deadlock prevention and deadlock resolution?
5. list and describe five areas that should be addressed in order to achieve high data quality.
6. what is an Information Resource Dictionary System?
7. List and briefly explain the ACID properties of a database transaction.
8. Explain the two common forms of encryption.
9. Outline six database protection procedures.

16.9 References

Modern Database System Concepts :

Jeffrey A. Hoffer . Mary B. Prescott Fred R. McFadden

Database Systems:

Ramez Elmasri and Shamkant B.Navathe

(Person Education Asia) 2002.

Database System Concepts:

Abraham Silberschatz.Henry F.Korth and S. Sudarshan.

Tata McGraw Hill 2002

Database Application Design and Development

Michael V.Manino. McGraw Hill Irwin.

Database Management Systems:

Gerald V.Post.Tata McGraw Hill 2002

Database Management Systems:

Raghurama Krishna and Johannes Gherkin

- C.V.P.R.PRASAD, M.C.A.,

Lecturer,
Dept. Of Computer Science,
JKC College, GUNTUR.