



# KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2017 onwards)

## DEPARTMENT OF COMPUTER SCIENCE, CA & IT

**SUBJECT : DATA BASE MANAGEMENT SYSTEMS**

**SEMESTER : IV**

**L T P C**

**SUBJECT CODE: 17CAU402**

**CLASS : II BCA**

**4 0 0 4**

### Scope

The scope of this course is to teach the fundamentals of the database systems at a graduate student level. This course also emphasize database concepts, developments, use and management.

### Objectives

To teach students

- the fundamentals of database management systems, techniques for the design of databases, and principles of database administration.
- the role and nature of relational database management systems in today's IT environment.
- Translate written business requirements into conceptual entity-relationship data models.
- develop and manage efficient and effective database applications

### Unit-I

**Introduction:** Characteristics of database approach, data models, database system architecture and data independence. **Entity Relationship(ER) Modeling:** Entity types, relationships, constraints.

### Unit-II

**Relation data model:** Relational model concepts, relational constraints, relational algebra.

### Unit-III

**Relation data model:** SQL queries **Database design:** Mapping ER/EER model to relational database, functional dependencies, Lossless decomposition.

#### Unit-IV

**Database design:** Normal forms (upto BCNF). **Transaction Processing :** ACID properties, concurrency control

#### Unit-V

**File Structure and Indexing** (8 Lectures) Operations on files, File of Unordered and ordered records, overview of File organizations, Indexing structures for files( Primary index, secondary index, clustering index), Multilevel indexing using B and B+ trees.

#### Suggested Readings

1. Elmasri, R., Navathe S.B., (2013). *Database Systems Models, Languages, Design and application Programming*, (6<sup>th</sup> ed.), Pearson Education.
2. Elmasri, R., Navathe, S.B.,(2010). *Fundamentals of Database Systems*, (6<sup>th</sup> ed.), Pearson Education.
3. Ramakrishanan,R., Gehrke,J., (2002). *Database Management Systems* (3<sup>rd</sup> ed.), McGraw-Hill.
4. Silberschatz,A., Korth, H.F., Sudarshan,S.,(2010). *Database System Concepts* (6<sup>th</sup> ed.), McGraw Hill.

#### Websites

1. <http://en.wikipedia.org/wiki/RDBMS>
2. [http://aspalliance.com/1211\\_Relational\\_Database\\_Management\\_Systems\\_\\_Concepts\\_and\\_Terminologies](http://aspalliance.com/1211_Relational_Database_Management_Systems__Concepts_and_Terminologies)
3. [www.compinfo-center.com/apps/rdbms.html](http://www.compinfo-center.com/apps/rdbms.html)



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Coimbatore - 641021.

(For the candidates admitted from 2016 onwards)

### DEPARTMENT OF COMPUTER SCIENCE, CA & IT

#### LECTURE PLAN

**STAFF NAME: JENNETH A**

**SUBJECT NAME: DATA BASE MANAGEMENT SYSTEMS**

**SUB.CODE: 17CAU402**

**SEMESTER: IV**

**CLASS: II BCA**

S.No.	Lecture Duration	Topics to be Covered	Support Materials
<b>Unit - I</b>			
1.	1	<b>Introduction to DBMS</b>	T1:4-5
2.		Characteristics of database approach	T1:8-12
	1	data models	T1:24-25
4.	1	database system architecture and data independence	T1:27-29
5.	1	Entity Relationship(ER) Modeling	T1:41-45
6.	1	Entity types	T1:45-52
7.	1	Relationships, constraints.	T1:52-59
8.	1	Recapitulation and Discussion of important questions	
<b>Total No. of Hours Planned for Unit-I</b>			<b>8</b>
Textbook		T1: Elmasri, R., & Navathe, S.B. (2010). Fundamentals of Database Systems (3rd ed.). New Delhi: Pearson Education,.	

#### Unit – II

1.	1	<b>Relation data model</b>	R1:145
2.	1	Relational model concepts	R1:196-202
3.	1	relational constraints	R1:202-206
		Key Constraints	
4.	1	Foreign key Constraints	R1:206-211
		relational algebra.	
5.	1	Selection and Projections	R1:212-216
6.	1	Set Operations, Renaming	R1:216-219
7.	1	Joins and Division	R1:219-230
8.	1	Recapitulation and Discussion of important questions	
<b>Total No. of Hours Planned for Unit-II</b>			<b>8</b>

**Unit – III**

1.	1	Relation data model:	R1:243
2.	1	SQL queries	R1:244-288
3.	1	Database design	R1:289-290
4.	1	Mapping ER/EER model to relational database	R1:290-294
5.	1	functional dependencies	R1:476-483
6.	1	Lossless decomposition	R1:505-511
7.	1	Recapitulation and Discussion of important questions	
<b>Total No. of Hours Planned for Unit-III</b>			<b>7</b>
Textbook		R1: Elmasri, R., & Navathe, S.B. (2010). Fundamentals of Database Systems (3rd.ed ). New Delhi: Pearson Education,.	

**Unit – IV**

1.	1	<b>Database design</b>	R1:465-476
2.	1	Normal forms 1NF	R1:485-488
3.	1	Normal forms 2NF	R1:488-489
4.	1	Normal forms 3NF	R1:489-490
5.	1	Normal forms BCNF	R1:493-496
6.	1	Transaction Processing	R1:629-633
7.	1	ACID properties	R1:640-641
8.	1	concurrency control	R2:633-635
9.	1	Recapitulation and Discussion of important questions	
<b>Total No. of Hours Planned for Unit-IV</b>			<b>9</b>
Textbook		R1: Elmasri, R., & Navathe, S.B. (2010). Fundamentals of Database Systems (3rd ed.). New Delhi: Pearson Education,.	

**Unit – V**

1.	1	<b>File Structure and Indexing:</b> Operations on files	R1:133-135
2.	1	File of ordered and Unordered records	R1:135-139
3.	1	Overview of file Organizations, Indexing structures for files	R1:155-164
4.	1	Multilevel indexing using B trees and B+ trees	R1:148-149
5.	1	Recapitulation and Discussion of important questions	
6.	1	Recapitulation and Discussion of previous semester question papers	
7.	1	Recapitulation and Discussion of previous semester question papers	
8.	1	Recapitulation and Discussion of previous semester question papers	
<b>Total No. of Hours Planned for Unit-V</b>			<b>8</b>
Textbook		R1: Elmasri, R., & Navathe, S.B. (2010). Fundamentals of Database Systems (3rd ed.). New Delhi: Pearson Education,	
Web Sites		W1: <a href="https://www.tutorialpoint.com">https://www.tutorialpoint.com</a> W2: <a href="https://www.w3schools.com">https://www.w3schools.com</a>	



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I Batch: 2017-2020

### UNIT I

Basic concepts - Database & Database Users - Characteristics of the Database - Database Systems. Concepts & Architecture-Data Models. Schemas & Instances - DBMS Architecture & Data Independence - Data Base languages & Interfaces – Data Modelling using the Entity - Relationship Approach

#### Basic Concepts

##### Data:

It is a collection of information.

The facts that can be recorded and which have implicit meaning known as 'data'.

##### Example:

Customer -----

1.cname.

2.cno.

3.ccity.

##### Database:

It is a collection of interrelated data. These can be stored in the form of tables.

A database can be of any size and varying complexity.

A database may be generated and manipulated manually or it may be computerized. Example:

Customer database consists the fields as cname, cno, and ccity

Cname	Cno	Ccity

##### Database System:

It is computerized system, whose overall purpose is to maintain the information and to make that the information is available on demand.

##### Advantages:

- 1.Redundency can be reduced.
- 2.Inconsistency can be avoided.
- 3.Data can be shared.
- 4.Standards can be enforced.
- 5.Security restrictions can be applied.
- 6.Integrity can be maintained.
- 7.Data gathering can be possible.
- 8.Requirements can be balanced.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402 Unit I Batch: 2017-2020

### **Database Management System (DBMS):**

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

#### **Disadvantages in File Processing**

- Data redundancy and inconsistency. ■ Difficult in accessing data.
- Data isolation.
- Data integrity.
- Concurrent access is not possible. ■ Security Problems.

### **Advantages of DBMS:**

- 1.Data Independence.
- 2.Efficient Data Access.
- 3.Data Integrity and security.
- 4.Data administration.
- 5.Concurrent access and Crash recovery.
- 6.Reduced Application Development Time.

### **Applications**

Database Applications:

Banking: all transactions Airlines: reservations, schedules Universities: registration, grades  
Sales: customers, products, purchases  
Online retailers: order tracking, customized recommendations Manufacturing: production, inventory, orders, supply chain Human resources: employee records, salaries, tax deductions

### **People who deal with databases**

Many persons are involved in the design, use and maintenance of any database. These persons can be classified into 2 types as below.

#### **Actors on the scene:**

The people, whose jobs involve the day-to-day use of a database are called as 'Actors on the scene', listed as below.

#### **1.Database Administrators (DBA):**

The DBA is responsible for authorizing access to the database, for



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I Batch: 2017-2020

Coordinating and monitoring its use and for acquiring software and hardware resources as needed.

These are the people, who maintain and design the database daily. DBA is responsible for the following issues.

a. Design of the conceptual and physical schemas:

The DBA is responsible for interacting with the users of the system to understand what data is to be stored in the DBMS and how it is likely to be used.

The DBA creates the original schema by writing a set of definitions and is Permanently stored in the 'Data Dictionary'.

b. Security and Authorization:

The DBA is responsible for ensuring the unauthorized data access is not permitted.

The granting of different types of authorization allows the DBA to regulate which parts of the database various users can access.

c. Storage structure and Access method definition:

The DBA creates appropriate storage structures and access methods by writing a set of definitions, which are translated by the DDL compiler.

d. Data Availability and Recovery from Failures:

The DBA must take steps to ensure that if the system fails, users can continue to access as much of the uncorrupted data as possible.

The DBA also work to restore the data to consistent state.

e. Database Tuning:

The DBA is responsible for modifying the database to ensure adequate Performance as requirements change.

f. Integrity Constraint Specification:

The integrity constraints are kept in a special system structure that is consulted by the DBA whenever an update takes place in the system.

### 2.Database Designers:

Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.

### 3. End Users:

People who wish to store and use data in a database.

End users are the people whose jobs require access to the database for querying, updating and generating reports, listed as below.

a. Casual End users:



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit I      Batch: 2017-2020

These people occasionally access the database, but they may need different information each time.

### b. Naive or Parametric End Users:

Their job function revolves around constantly querying and updating the database using standard types of queries and updates.

### c. Sophisticated End Users:

These include Engineers, Scientists, Business analyst and others familiarize to implement their applications to meet their complex requirements.

### d. Stand alone End users:

These people maintain personal databases by using ready-made program packages that provide easy to use menu based interfaces.

## 4. System Analyst:

These people determine the requirements of end users and develop specifications for transactions.

## 5. Application Programmers (Software Engineers):

These people can test, debug, document and maintain the specified transactions.

### b. Workers behind the scene:

#### Database Designers and Implementers:

These people who design and implement the DBMS modules and interfaces as a software package.

#### 2. Tool Developers:

Include persons who design and implement tools consisting the packages for design, performance monitoring, and prototyping and test data generation.

#### 3. Operators and maintenance personnel:

These are the system administration personnel who are responsible for the actual running and maintenance of the hardware and software environment for the database system.

## 3. LEVELS OF DATA ABSTRACTION

This is also called as 'The Three-Schema Architecture', which can be used to separate the user applications and the physical database.

### 1. Physical Level:

This is a lowest level, which describes how the data is actually stored.





## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit I      Batch: 2017-2020

### Example:

Customer account database can be described.

### 2.Logical Level:

This is next higher level that describes what data and what relationships in the database.

### Example:

Each record

type customer = record cust\_name: sting;

cust\_city: string;

cust\_street: string;

end;

### 3.Conceptual (view) Level:

This is a lowest level, which describes entire database. Example:

All application programs.

## 1.2 Database Fundamentals

### **Information and Data**

Information is defined as the knowledge of something; particularly, an event, situation, or knowledge derived based on research or experience. Data is any information related to an organization that should be stored for any purpose according to the requirements of an organization.

### **What Is a Database?**

A database is a mechanism that is used to store information, or data. A legacy database is simply a database that is currently in use by a company.

### **What Are the Uses of a Database?**

One of the most traditional manual processes with which most of us are familiar is the management of information in a file cabinet.

Some of the most common uses for a database include

- Tracking of long-term statistics and trends
- Automating manual processes to eliminate paper shuffling
- Managing different types of transactions performed by an individual or business
- Maintaining historic information

There are two types of relational databases. A transactional, or Online Transactional Processing (OLTP), database is one that is used to process data on a regular basis. A good example of a transactional database is one for class scheduling and student registrations. An Online Analytical Processing (OLAP) database is one whose main purpose is to supply end-users with data in



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I Batch: 2017-2020

response to queries that are submitted. Typically, the only transactional activity that occurs in an OLAP database concerns bulk data loads. OLAP data is used to make intelligent business decisions based on summarized data, company performance data, and trends. The two main types of OLAP databases are Decision Support Systems (DSS) and Data Warehouses.

### 1.3 Database Elements

Several topics are discussed in the following sections. These topics include:

- The database schema
- Schema objects
- Tables
- Fields and columns
- Records and rows
- Keys
- Relationships
- Data types

#### Database Schema

A schema is quite simply a group of related objects in a database.

The three models associated with a schema are as follows:

- The conceptual model, also called the logical model, is the basic database model, which deals with organizational structures that are used to define database structures such as tables and constraints.
- The internal model, also called the physical model, deals with the physical storage of the database, as well as access to the data, such as through data storage in tables and the use of indexes to expedite data access. The internal model separates the physical requirements of the hardware and the operating system from the data model.
- The external model, or application interface, deals with methods through which users may access the schema, such as through the use of a data input form. The external model allows relationships to be created between the user application and the data model.

#### Table

A table is the primary unit of physical storage for data in a database. When a user accesses the database, a table is usually referenced for the desired data. Multiple tables might comprise a database, therefore a relationship might exist between tables. Because tables store data, a table requires physical storage on the host computer for the database.

#### Columns

A column, or field, is a specific category of information that exists in a table. A column is to a table what an attribute is to an entity

#### Rows

A row of data is the collection of all the columns in a table associated with a single occurrence. Simply speaking, a row of data is a single record in a table

#### Data Types

A data type determines the type of data that can be stored in a database column.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I Batch: 2017-2020

Although many data types are available, three of the most commonly used data types are

- Alphanumeric
- Numeric
- Date and time

Alphanumeric data types are used to store characters, numbers, special characters, or nearly any combination. If a numeric value is stored in an alphanumeric field, the value is treated as a character, not a number.

### Database Integrity

Data integrity is the insurance of accurate data in the database. Within the scope of the database, data integrity is controlled mostly by column constraints. Constraints validate the values of the data placed in the database. Constraints can be implemented at both the column level and the table level. Constraints can be used to ensure that duplicate data is not entered into the database. Constraints are also typically used to ensure that new or modified table data adhere to the business rules defined

Referential integrity is the process of ensuring that data is consistent between related tables. Referential integrity is a concept that deals with parent/child relationships in the database. Referential integrity constraints are created in order to ensure that data entered into one table is synchronized with other related tables. Values from one column are dependent on the values from another column in another table.

Referential integrity is controlled by keys. A key is a column value in a table that is used to either uniquely identify a row of data in a table, or establish a relationship with another table. A key is normally correlated with one column in table, although it might be associated with multiple columns. There are two types of keys: primary and foreign.

### Primary Keys

A primary key is the combination of one or more column values in a table that make a row of data unique within the table. Primary keys are typically used to join related tables. Even if a table has no child table, a primary key can be used to disallow the entry of duplicate records into a table. For example, an employee's social security number is sometimes considered a primary key candidate because all SSNs are unique.

### Foreign Keys

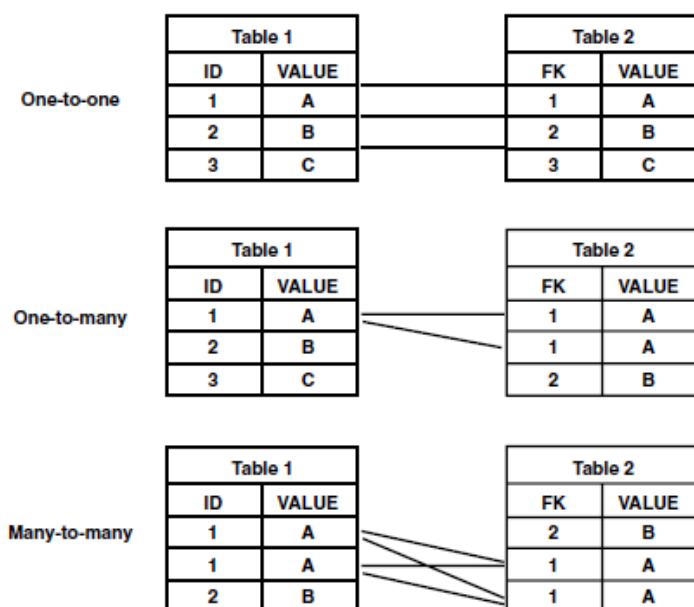
A foreign key is the combination of one or more column values in a table that reference a primary key in another table. Foreign keys are defined in child tables. A foreign key ensures that a parent record has been created before a child record. Conversely, a foreign key also ensures that the child record is deleted before the parent record.

### Relationships

Most databases are divided into many tables, most of which are related to one another. In most modern databases, such as the relational database, relationships are established through the use of primary and foreign keys.

Three types of table relationships that can be derived are as follows:

- One-to-one - One record in a table is related to only one record in another table.
- One-to-many - One record in a table can be related to many records in another table.
- Many-to-many - One record in a table can be related to one or more records in another table, and one or more records in the second table can be related to one or more records in the first table.



**Fig 1.1 – Types of Relationships**

## 1.4 Database Design Concepts

Before a design effort can proceed full speed ahead, the designer must first take time to understand the business. Understanding the business involves understanding the entities, data, and rules within an organization, and then converting these attributes of the business into a business model. Then, the designer must have a solid comprehension of the proposed database model. Finally, the designer will convert the business model into a database model, using a design methodology, whether automated or a manual process.

### Design Methodology

A design methodology is the approach taken toward the design of a database. It is the process of designing a database with a sound plan from the beginning.

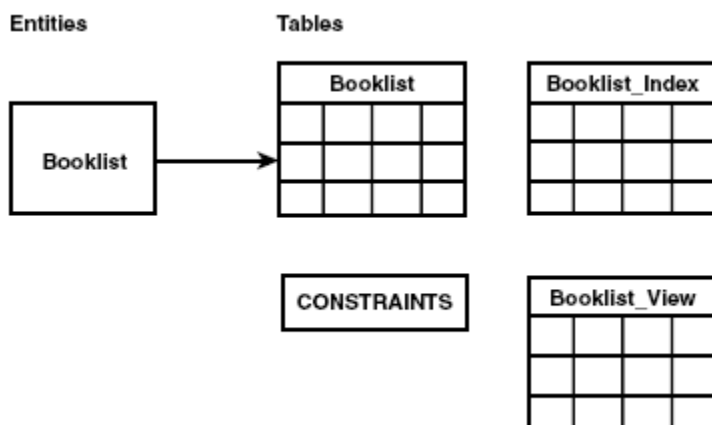
Some of the advantages of using a design methodology include

- It provides a step by step guide toward database design.

- Little or no trial and error is involved.
- It is easy to document the database and application with the availability of design plans, drawings depicting the organization's needs, and other deliverables specified.
- It is easy to modify the database in the future as organization and planning eases the tasks of managing changes.

### Converting the Business Model to Design

Database design is the process of converting business objects into tables and views. It is the process of actually designing a database based on a business model. Business model components such as entities and attributes are converted into tables and columns. Constraints are added to columns where necessary in order to enforce data and referential integrity. Views of tables might be created in order to filter the data that a user sees, or to simplify the query process



**Fig 1.2 – Table Structure**

### Application Design

Application design is the process of creating an interface for the end user through which the database can be accessed. It is the process of transforming business processes that have been defined into a usable application through which the end user can easily access information in the database. A typical application might consist of a set of forms that allow the end user to create new records in the database, update or delete existing records, or perform queries. The application might also include canned queries and reports. Common tools used to develop an application include Oracle Designer, Oracle Developer/2000, Visual Basic, C++, and PowerBuilder.

### 1.5 Components of DBMS

A DBMS is a complex software system that is used to manage, store and manipulate data and the metadata used to describe the data.

#### 1.5.1 Classification of DBMS Users

The users of a database system can be classified in the following groups, depending on their degree of expertise or the mode of their interactions with the DBMS.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit I      Batch: 2017-2020

- a) Naive Users - Users who need not be aware of the presence of the database system or any other system supporting their usage are considered as naïve users. Some examples for Naïve users are i) user of an ATM machine ii) end users of the database who work through a menu-oriented application program.
- b) Online Users – These are users who may communicate with the database directly via an online terminal or indirectly via a user interface and application program.
- c) Application programmers – These are professional programmers who are responsible for developing application programs or user interface utilized by the naïve users and online users.
- d) Database Administrator – Centralized control of the database is exerted by a person or group under the supervision of a high-level administrator. this person or group is referred to as database administrator (DBA).

### Responsibilities of DBA

- Create modify and maintain the above three levels of users
- Controls the database structure and custodian for data.
- Sets up the definition of conceptual (global) view of the database.
- Defines and implements internal level of database.
- Maintain integrity of database
- Grant permission to users and maintain profile of each user
- Restrict unauthorized users from accessing the database
- Define procedures to recover database from failures due to human, natural or hardware courses.

### 1.5.2 DBMS Facilities

Two types of facilities are provided

#### a) Data definition facility or Data Definition Language (DDL)

DDL defines the conceptual scheme and give details about how to implement this scheme in physical devices to store the data. The definition includes Entity sets, Attributes, Entity relationships and Constraints. These definitions are described as metadata and expressed in a compiled form. This compiled form of definition is called data dictionary, directory or system catalog.

#### b) Data manipulation facility or Data Manipulation Language (DML)

The language used to manipulate data in the database is called DML. It involves retrieval of data from database (query), insertion of new data, deletion or modification of existing data. A query is a statement in DML that requests retrieval of data from the database. The DML can be procedural (the user indicates not only what to retrieve but how to go about retrieving it) or non-procedural (user indicate only what is to be retrieved).



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit I      Batch: 2017-2020

### 1.5.3 Structure of a DBMS

The major components of DBMS are

a) Data definition language compiler – it converts data definition statements into a set of tables. These tables contain metadata concerning database that can be used by other components of database.

b) Data Manager – It is the central component of DBMS also called as database control system. Its functions are

- Convert operation in user's queries coming directly from query processor or indirectly from application program to a physical file system.
- Enforce constraints to maintain consistency, integrity and security
- Synchronization of simultaneous operations
- Backup and recovery operations

c) File Manager – responsible for structure of files and managing file space. It locates block containing the required record, requests this block from disk manager and transmits required record to data manager.

d) Disk manager – it transfers block or page requested by file manager. It performs all physical I/O operations.

e) Query processor – It interprets online users query and convert it into an efficient series of operations in the form capable of being sent to the data manager for execution.

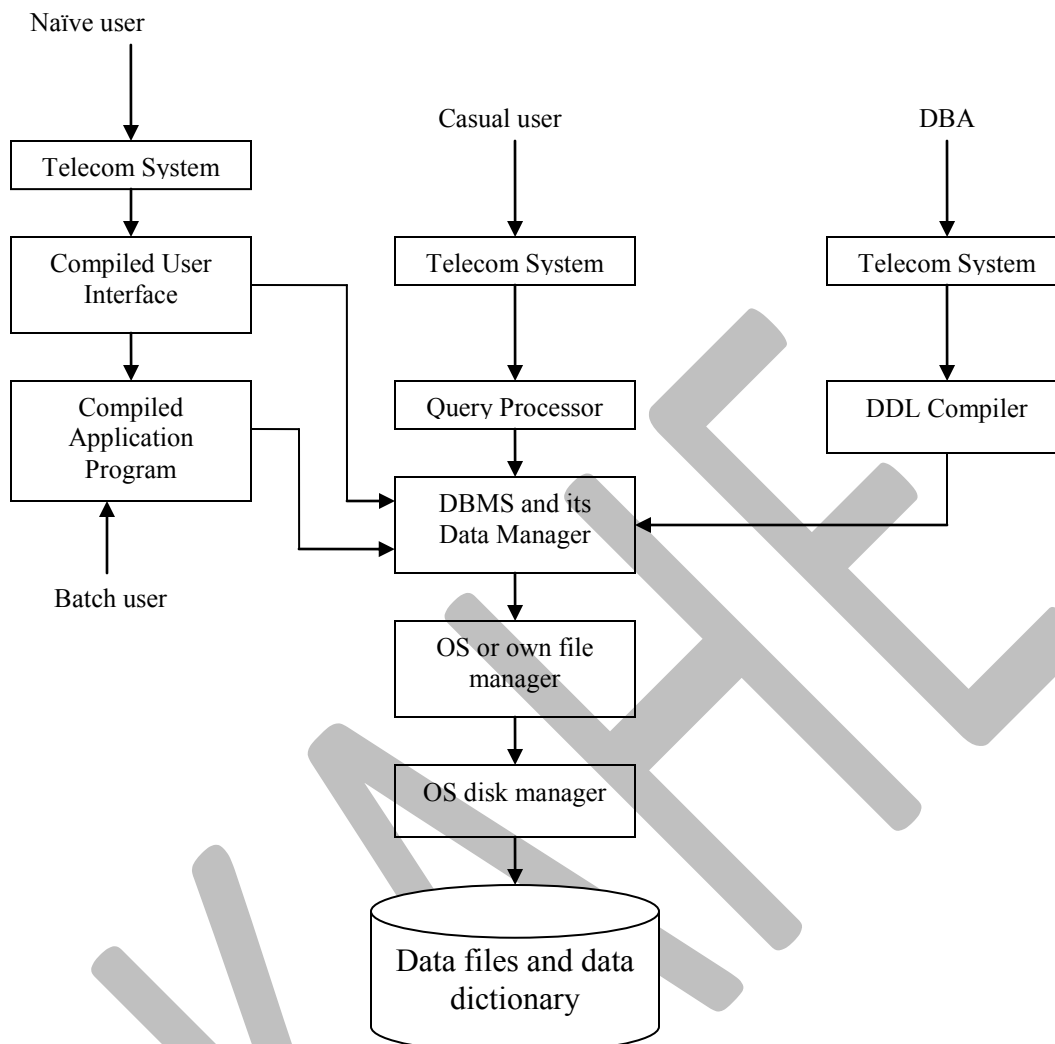
f) Telecommunication system – online users of a computer system whether remote or local communicate with it by sending and receiving message over communication lines. These messages are routed via an independent software system called a telecommunication system or a communication control program.

g) Data files – It contains data portion of database

h) Data dictionary or system catalog – information pertaining to the structure and usage of data contained in the database, the metadata, is maintained in the data dictionary.

i) Access aids – a set of access aids in the form of indexes are usually provided in a database system. Commands can be provided to build and destroy additional temporary indexes.





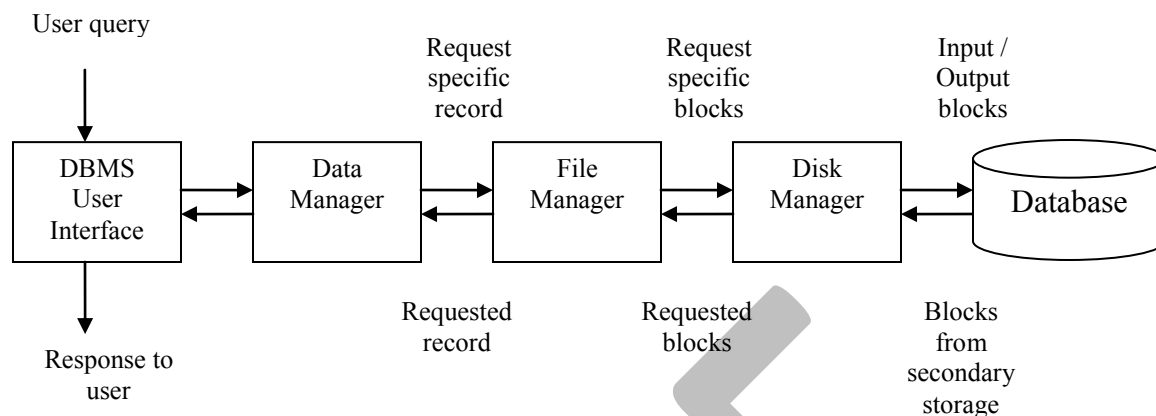
**Fig1.3 Structure of Database Management System**

### 1.5.4 Database Access

#### Steps

- Users request for data is received by data manager
- Data manager sends request for specific record to the file manager
- The file manager decides which physical block of secondary storage device contains required record.
- File manager sends request for appropriate block to appropriate disk manager
- Disk manager retrieves the block and sends it to the file manager, which sends required record to data manager.





**Fig1.4 Steps in data access**

## 1.6 Advantages and Disadvantages of a DBMS

### 1.6.1 Advantages of DBMS

- Reduction of redundancies
- Shared data
- Integrity
- Security
- Conflict resolution
- Data independence

### 1.6.2 Disadvantages of a DBMS

- Problems associated with centralization
- Cost of software/hardware migration
- Complexity of backup and recovery

## DATA MODELS

The entire structure of a database can be described using a data model. A data model is a collection of conceptual tools for describing

Data models can be classified into following types.

- 1.Object Based Logical Models.
- 2.Record Based Logical Models.
- 3.Physical Models.

Explanation is as below.

### 1.Object Based Logical Models:



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I Batch: 2017-2020

These models can be used in describing the data at the logical and view levels.

These models are having flexible structuring capabilities classified into following types. a) The entity-relationship model.

b) The object-oriented model.

c) The semantic data model.

d) The functional data model.

### **2. Record Based Logical Models:**

These models can also be used in describing the data at the logical and view levels.

These models can be used for both to specify the overall logical structure of the database and a higher-level description.

These models can be classified into,

1. Relational model.
2. Network model.
3. Hierarchical model.

### **3. Physical Models:**

These models can be used in describing the data at the lowest level, i.e. physical level.

These models can be classified into

1. Unifying model
2. Frame memory model.

#### **Entity Relational Model (E-R Model)**

The E-R model can be used to describe the data involved in a real world enterprise in terms of objects and their relationships.

Uses:

These models can be used in database design.

It provides useful concepts that allow us to move from an informal description to precise description.

This model was developed to facilitate database design by allowing the specification of overall logical structure of a database.

It is extremely useful in mapping the meanings and interactions of real world enterprises onto a conceptual schema.

These models can be used for the conceptual design of database applications.

### **OVERVIEW OF DATABASE DESIGN**

The problem of database design is stated as below.

'Design the logical and physical structure of 1 or more databases to accommodate the



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I

Batch: 2017-2020

information needs of the users in an organization for a defined set of applications'.

The goals database designs are as below.

1. Satisfy the information content requirements of the specified users and applications.
2. Provide a natural and easy to understand structuring of the information.
3. Support processing requirements and any performance objectives such as 'response time, processing time, storage space etc.. ER model consists the following 3 steps.

### **a. Requirements Collection and Analysis:**

This is the first step in designing any database application.

This is an informal process that involves discussions and studies and analyzing the expectations of the users & the intended uses of the database.

Under this, we have to understand the following.

1. What data is to be stored in a database?
2. What applications must be built?
3. What operations can be used?

Example:

For customer database, data is cust-name, cust-city, and cust-no.

### **b. Conceptual database design:**

The information gathered in the requirements analysis step is used to develop a higher-level description of the data.

The goal of conceptual database design is a complete understanding of the database structure, meaning (semantics), inter-relationships and constraints.

Characteristics of this phase are as below.

#### **1. Expressiveness:**

The data model should be expressive to distinguish different types of data, relationships and constraints.

#### **2. Simplicity and Understandability:**

The model should be simple to understand the concepts.

#### **3. Minimality:**



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I Batch: 2017-2020

The model should have small number of basic concepts.

### 4. Diagrammatic Representation:

The model should have a diagrammatic notation for displaying the conceptual schema.

### 5. Formality:

A conceptual schema expressed in the data model must represent a formal specification of the data. Example:

```
Cust_n  
ame   :  
string;  
Cust_n  
o     :  
integer  
;  
Cust_c  
ity   :  
string;
```

### c. Logical Database Design:

Under this, we must choose a DBMS to implement our database design and convert the conceptual database design into a database schema.

The choice of DBMS is governed by number of factors as below.

1. Economic Factors.
2. Organizational Factor

### 1. Economic Factors:

These factors consist of the financial status of the applications. a. Software Acquisition

#### Cost:

This consists buying the software including language options such as forms, menu, recovery/backup options, web based graphic user interface (GUI) tools and documentation.

#### b. Maintenance Cost:

This is the cost of receiving standard maintenance service from the vendor and for keeping the DBMS version up to date.

#### c. Hardware Acquisition Cost:

This is the cost of additional memory, disk drives, controllers and a specialized DBMS storage.

#### d. Database Creation and Conversion Cost:



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I

Batch: 2017-2020

This is the cost of creating the database system from scratch and converting an existing system to the new DBMS software.

### e. Personal Cost:

This is the cost of re-organization of the data processing department.

### f. Training Cost:

This is the cost of training for Programming, Application Development and Database Administration.

### g. Operating Cost:

The cost of continued operation of the database system.

## **2.Organizational Factors:**

These factors support the organization of the vendor, can be listed as below.

### a. Data Complexity:

Need

of a DBMS.

### b. Sharing among

### applications:

The greater the sharing among applications, the more the redundancy among files and hence the greater the need for a DBMS.

### c. Dynamically evolving or growing data:

If the data changes constantly, it is easier to cope with these changes using a DBMS than using a file system.

### d. Frequency of ad hoc requests for data:

File systems are not suitable for ad hoc retrieval of data.

### e. Data Volume and Need for Control:

These 2 factors needs for a DBMS.

### Example:

Customer database can be represented in the form of tables or diagrams.

## **3. Schema Refinement:**

Under this, we have to analyze the collection of relations in our relational database schema to identify the potential problems.

## **4.Physical Database Design:**

Physical database design is the process of choosing specific storage structures and access paths for the database files to achieve good performance for the various database applications.

This step involves building indexes on some tables and



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I

Batch: 2017-2020

clustering some tables. The physical database design can have the following options.

### 1. Response Time:

This is the elapsed time between submitting a database transaction for execution and receiving a response.

### 2. Space Utilization:

This is the amount of storage space used by the database files and their access path structures on disk including indexes and other access paths.

### 3. Transaction Throughput:

This is the average number of transactions that can be processed per minute.

## **5. Security Design:**

In this step, we must identify different user groups and different roles played by various users.

For each role, and user group, we must identify the parts of the database that they must be able to access, which are as below.

## **2. ENTITIES**

1. It is a collection of objects.
2. An entity is an object that is distinguishable from other objects by a set of attributes.
3. This is the basic object of E-R Model, which is a 'thing' in the real world with an independent existence.
4. An entity may be an 'object' with a physical existence.
5. Entities can be represented by 'Ellipses'.

Example:

- i. Customer, account etc.

## **ATTRIBUTES**

Characteristics of an entity are called as an attribute.

The properties of a particular entity are called as attributes of that specified entity. Example:

Name, street\_address, city --- customer database.

Acc-no, balance ---

account database.

### Types:

These can be classified into following types.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I

Batch: 2017-2020

- 1.Simple Attributes.
- 2.Composite Attributes.
- 3.Single Valued Attributes.
- 4.Mutivalued Attributes.
- 5.Stored Attributes.
- 6.Derived Attributes

Explanati  
on is as  
below.

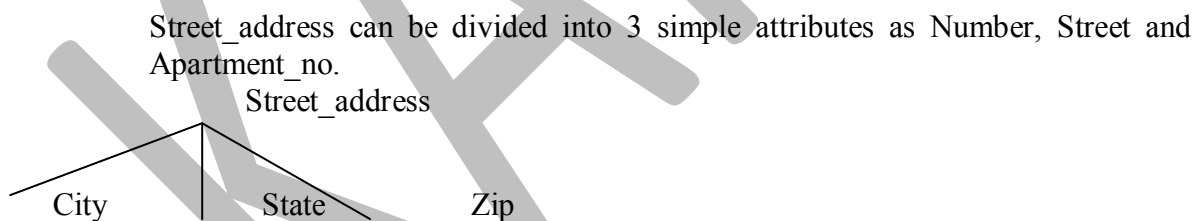
### 1.Simple Attributes:

The attributes that are not divisible are called as 'simple or atomic attributes'. Example:  
cust\_name, acc\_no etc..

### 2.Composite Attributes:

The attributes that can be divided into smaller subparts, which represent more basic attributes with independent meaning. These are useful to model situations in which a user sometimes refers to the composite attribute as unit but at other times refers specifically to its components.

#### Example:



### 3.Single Valued Attribute:

The attributes having a single value for a particular entity are called as 'Single Valued Attributes'.

#### Example:

'Age' is a single valued attribute of 'Person'.

### 4.Muti Valued Attribute:

The attributes, which are having a set of values for the same entity, are called as 'Multi Valued Attributes'. Example:

A 'College Degree' attribute for a person.i.e, one person may not have a college degree, another



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I

Batch: 2017-2020

person may have one and a third person may have 2 or more degrees.

A multi-valued attribute may have lower and upper bounds on the number of values allowed for each individual entity.

### 5. Derived Attributes:

An attribute which is derived from another attribute is called as a 'derived attribute'.

Example:

'Age' attribute is derived from another attribute 'Date'.

### 6. Stored Attribute:

An attribute which is not derived from another attribute is called as a 'stored attribute'.

Example:

In the above example, 'Date' is a stored attribute.

### Entity Type:

A collection entities that have the same attributes is called as an 'entity type'. Each entity type is described by its name and attributes.

### Entity Set:

Collection of all entities of a particular entity type in the database at any point of time is called as an entity set.

The entity set is usually referred to using the same name as the entity type.

An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name. Example:

Collection of customers.

## **5. Relationships**

It is an association among entities.

## **6. Relationship Sets**

It is a collection of relationships.

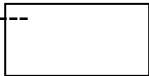
Primary Key:

The attribute, which can be used to identify the specified information from the tables.

### Weak Entity:

A weak entity can be identified uniquely by considering some of its attributes in conjunction with the primary key of another entity.

The symbols that can be used in this model are as follows.

1. Rectangles ---  --- Entities.





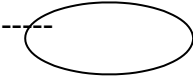





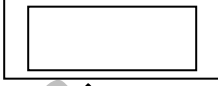
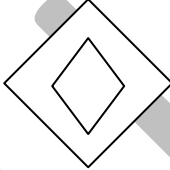
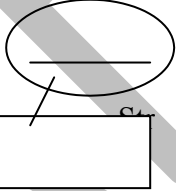

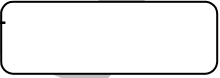
## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit I

Batch: 2017-2020

2. Ellipses -----  ----- Attributes.
3. Lines -----  ----- Links.
4. Diamonds -----  Relationships.
5. Under Lined Ellipse -----  Primary key.
6. Doubled Lined Ellipse -----  Multi Valued Attribute.
7. Dashed Ellipse -----  Derived Attributes.
8. Double Lined Rectangle -----  entity Set.
9. Double Lined Diamond -----  Weak Entity Relationship.  
Identifying Relationship.
10. Entity Set having a Primary Key -----  ong Entity Set.
11. Cylinder -----  Database.
12. Curved Inside Rectangle -----  End Users.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit I      Batch: 2017-2020

### POSSIBLE QUESTIONS

#### UNIT 1

##### PART – A (20 MARKS)

(Q.NO 1 TO 20 Online Examinations)

##### Part - B (5X 6 =30 Marks) (Answer ALL the Questions)

1. List out the major characteristics of the database approach.
2. Discuss on the different categories of data model.
3. Write short notes on schemas, instances and database state.
4. Elaborate the three schema architecture of DBMS in detail.
5. Write a detailed note on the two types of data independence.
6. Discuss on DBMS languages.
7. What is meant by DBMS interfaces? Explain them in detail.
8. Illustrate the main phases of database design.
9. Describe the different types of attributes in the ER model.
10. Write short notes on entity types and entity sets.

##### Part - C (10 X 1 =10 Marks) (Compulsory Question)

1. Define the term locking. Explain the different types of locking used in concurrency control.
2. Explain in detail the 13 object oriented database management system Commandments.
3. Compare and contrast the traditional and object oriented DBMS.
4. Elaborate different Data Manipulation Language in SQL. Explain the different statements used in DML.
5. Discuss on functional dependencies of relational database design in detail with a suitable example.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

(Established Under Section 3 of UGC Act 1956)

COIMBATORE – 641 021

**DEPARTMENT OF COMPUTER APPLICATIONS**

Batch 2017 – 2020

**Subject : Database Management Systems****Subject Code:****17CAU402****UNIT I**

S.No	Questions	opt1	opt2	opt3	opt4	Answer
1	_____ was adopted by the ANSI and ISO.	PSQL	SQL	R-SQL	Sequel	SQL
2	_____ is a collection of high-level data description constructs that hide many low-level storage details	Data Model	ER Model	Network Model	none	Data Model
3	Database management System based on _____	Network model	Hierarchical model	Relational model	Object-based model	Relational model
4	A widely used Semantic model called _____	Network model	ER Model	Object-based model	Hierarchical model	ER Model
5	_____ is a more abstract	ER model	Semantic data model	Conceptual data Model	Physical data model	Semantic data model
6	_____ model is used to pictorially denote entities & relationships	Physical data model	ER model	network model	structure chart	ER model
7	A description Of data in terms of a data model is called _____	Schema	relation	record	entities	Schema
8	Field is otherwise known as _____	Column	Entity	Relationship	Relation	Column
9	Column is otherwise known as _____	Entity	Relationship	Relation	attribute	attribute
10	_____ is a software designed to assist in maintaining and utilizing large collections of data.	Database	DBMS	Entities	attributes.	DBMS
11	_____ model used Object store & versant.	Network Model	Hierarchical Model	Object Oriented Model	Record based Model	Object Oriented Model
12	_____ is used to define the external and conceptual model	DDL	DML	DCL	TCL	DDL
13	Conceptual model otherwise called as _____	Physical Schema	Internal Schema	Logical Schema	relations	Logical Schema
14	Physical model specifies _____ details	Information	data	Storage	relationships	Storage
15	The _____ enviroment involves dumb terminals	mainframe	client/server	internet computing	LAN	mainframe
16	A host computer in internet computing eniroment is called ____	server	data server	PC	web server	web server
17	_____ is the primary unit of storage in a database	table	column	row	number	table
18	database design involves conversion of _____ to stuctured database model.	business process	business model	entity	relationships	business model
19	The relationship between tables in the network model is called a	parent/child	set structure	client/server	tree/node	set structure

20	Set structures can represent a _____ relationship between tables	one-to-one	one-to-many	many-to-many	many-to-one	one-to-many
21	SQL has been developed and used for _____ model	relational	Hierarchical	network	flat file	relational
22	A class is the equivalent of a _____ in a relational database	row	column	table	primary key	table
23	SQL3 is also referred to as	SQL97	SQL98	SQL99	SQL100	SQL99
24	_____ is the process of creating an interface for the end user through which the database can be accessed	database design	business model	interface design	Application design	Application design
25	The process of reducing data redundancy in a relational database is called	data security	data accuracy	data protection	normalization	normalization
26	Static, or _____ data is seldom or never modified once stored in the database.	dynamic	historic	information	transactional	historic
27	_____ or transactional data, is data that is frequently modified once stored in the database.	dynamic	historic	information	transactional	dynamic
28	BPR is	Business product re-engineering	Business product repair	Business process re-engineering	Business procedure re-engineering	Business process re-engineering
29	_____ is the process of ensuring that data is consistent between related tables	primary key	database security	performance	Referential integrity	Referential integrity
30	Foreign keys are defined in _____ tables	parent	child	one	database	child
31	_____ is an object in the real world	Entity	Attribute	Relationship	Property	Entity
32	in database model the data is stored in objects	hierarchical	network	relational	object_oriented	object_oriented
33	In relational model the data is stored in _____	table	files	objects	sets	table
34	Information about the conceptual, external and physical schemas is stored in _____	Directory	System Catalogs	IMS	Information System	System Catalogs
35	Conceptual schema otherwise called as _____	Physical Schema	Internal Schema	Logical Schema	relations	Logical Schema
36	Physical Schema specifies _____ details	Information	data	Storage	relationships	Storage
37	_____ is used to speed up data retrieval operations.	DML Operations	Select Operation	Indexes	select operation with where condition	Indexes
38	A Structure of database using the given data model is called a	Database	Relation	Schema	design	Schema
39	SQL was developed as an integral part of	A hierarchical database	A relational database	A OO database	A network database	A relational database
40	Which of the following is CORRECT about database management system's languages?	Data definition languages are used to specify the conceptual schema only.	Data manipulation languages are used to create the databases.	Data manipulation languages are used for retrieval, insertion, deletion and modification of data.	Data definition languages are only used to update data in the DBMS	Data manipulation languages are used for retrieval, insertion, deletion and modification of data.
41	An E-R modelling for given application leads to	conceptual data model	logical data model	external data model	internal data model	conceptual data model
42	A conceptual data model is converted using a Relational Data Base Management System to a	logical data model	external data model	internal data model	an entity-relation data model	logical data model

43	A subset of logical data model accessed by programmers is called a	conceptual data model	external data model	internal data model	an entity-relation data model	external data model
44	When a logical model is mapped into a physical storage such as a disk store the resultant data model is known as	conceptual data model	external data model	internal data model	disk data model	internal data model
45	By data integrity we mean	maintaining consistent data values	integrated data values	banning improper access to data	not leaking data values	maintaining consistent data values
46	Data integrity is ensured by	good data editing	propagating data changes to all data items	preventing unauthorized access	preventing data duplication	propagating data changes to all data items
47	By data security in DBMS we mean	preventing access to data	allowing access to data only to authorized users	preventing changing data	introducing integrity constraints	allowing access to data only to authorized users
48	By redundancy in a file based system we mean that	unnecessary data is stored	same data is duplicated in many files	data is unavailable	files have redundant data	same data is duplicated in many files
49	Data integrity in a file based system may be lost because	the same variable may have different values in different files	files are duplicated	unnecessary data is stored in files	redundant data is stored in files	the same variable may have different values in different files
50	Data availability is often difficult in file based system	as files are duplicated	as unnecessary data are stored in files	as one has to search different files and these files may be in different update states	redundant data are stored in files	as one has to search different files and these files may be in different update states
51	_____ activity deals with defining the interface for end users	view design	conceptual design	functional design	design	view design
52	which the enterprises is examined to determine entity types and relationship among these entities	view design	conceptual design	functional design	design	conceptual design
53	_____ is concerned with determing the entities and their attributes and the relationship among them	entity analysis	functional analysis	relationship analysis	Entitiy relationship	entity analysis
54	Determining the fundamental functions with which the modeled enterprises involved _____	entity analysis	functional analysis	relationship analysis	ER Diagramme	functional analysis
55	The data independence are basically _____ types.	One type	Two type	Six Type	Ten type	Two type
56	Logical data Independence deals with	Physical Structure	Logical Structure	Database Structure	Network Structure	Logical Structure
57	Physical data independence deals with hiding the details of	Physical Structure	Logical Structure	Database Structure	Network Structure	Network Structure
58	_____ is final form of transparency in DDBS	Network transparency	Replication Transparency	Data Independence	Fragmentation transparency	Fragmentation transparency
59	which is commonly done for reasons of performance ,avialability ,and reliablity	fragmentation	replication	network transparecy	data independence	fragmentation
60	How many types of fragmentation alternatives generally followed	one	two	three	four	two
61	Horizontal fragmentation has a subset of _____ relation	tuples	columns	Rows and columns	vertical	tuples
62	Vertical fragmenation has a subset of _____ relation	Rows and coulumns	centrally	logically	columns	columns

63	The data to be stored in close proximity to use is called_____	Data Localization	Centralization	Conceptual data base	Improved performance	Data Localization



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit II Batch: 2017-2020

### UNIT II

Relational Model. Languages & Systems - Relational Data Model & Relational Algebra - Relational Model Concepts - Relational Model Constraints-Relational Algebra - SQL - A Relational Database Language - Data Definition in SQL - View & Queries in SQL - Specifying Constraints & Indexes in SQL - Specifying Constraints & Indexes in SQL - a Relational Database Management Systems - ORACLE/INGRES

### RELATIONAL MODEL

A database is a collection of 1 or more 'relations', where each relation is a table with rows and columns.

This is the primary data model for commercial data processing applications. The major advantages of the relational model over the older data models are,

- 1.It is simple and elegant.
- 2.simple data representation.
- 3.The ease with which even complex queries can be

expressed. Introduction:

The main construct for representing data in the relational model is a 'relation'.

A relation consists of

- 1.Relation Schema.
- 2.Relation Instance.

Explanation is as below.

#### 1.Relation Schema:

The relation schema describes the column heads for the table.

The schema specifies the relation's name, the name of each field (column, attribute) and the 'domain' of each field.

A domain is referred to in a relation schema by the domain name and has a set of associated values. Example:

Student information in a university database to illustrate the parts of a relation schema.

Students (Sid: string, name: string, login: string, age: integer, gross: real)

This says that the field named 'sid' has a domain named 'string'.

The set of values associated with domain 'string' is the set of all character strings.

## 2.Relation Instance:

This is a table specifying the information.

An instance of a relation is a set of 'tuples', also called 'records', in which each tuple has the same number

of fields as the relation schemas.

A relation instance can be thought of as a table in which each tuple is a row and all rows have the same number of fields.

The relation instance is also called as 'relation'.

Each relation is defined to be a set of unique tuples or rows.

## **Database Models**

### **1.7.1 Flat-File Database Model**

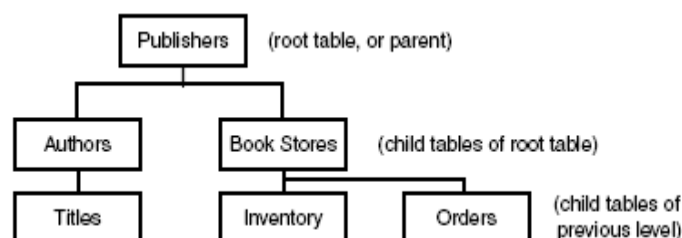
Before vendors such as Oracle and Microsoft started developing database management systems that run on a computer, many companies that were using computers stored their data in flat files on a host computer. The use of flat files to store data was predominant in the mainframe era. A flat-file database consists of one or more readable files, normally stored in a text format. Information in these files is stored as fields, the fields having either a constant length or a variable length separated by some character (delimiter).

### **Drawbacks of a flat-file database**

- Flat files do not promote a structure in which data can easily be related.
- It is difficult to manage data effectively and to ensure accuracy.
- It is usually necessary to store redundant data, which causes more work to accurately maintain the data.
- The physical location of the data field within the file must be known.
- A program must be developed to manage the data.

### **1.7.2 Hierarchical Database Model**

A hierarchical database is a step above that of a flat-file database, mainly because of the ability to establish and maintain relationships between groups of data. The architecture of a hierarchical database is based on the concept of parent/child relationships. In a hierarchical database, a root table, or parent table, resides at the top of the structure, which points to child tables containing related data. The structure of a hierarchical database model appears as an inverted tree, as shown in Figure







## **KARPAGAM ACADEMY OF HIGHER EDUCATION**

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402      Unit II      Batch: 2017-2020

**Fig 1.5 Hierarchical Model**

### **Benefits of the hierarchical model over the flat-file model**

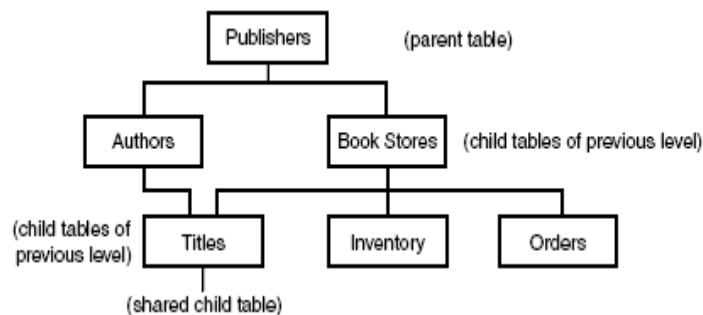
- Data can be quickly retrieved.
- Data integrity is easier to manage.

### **Drawbacks of the hierarchical model**

- Users must be very familiar with the database structure.
- Redundant data is stored.

### 1.7.3 Network Database Model

Improvements were made to the hierarchical database model in order to derive the network model. As in the hierarchical model, tables are related to one another. One of the main advantages of the network model is the capability of parent tables to share relationships with child tables. This means that a child table can have multiple parent tables. Additionally, a user can access data by starting with any table in the structure, navigating either up or down in the tree. The user is not required to access a root table first to get to child tables. The relationship between tables in the network model is called a set structure, where one table is the owner and another table is a member.



**Fig 1.6 Network Model**

#### **Benefits of the network database model**

- Data is accessed very quickly.
- Users can access data starting with any table.
- It is easier to model more complex databases.
- It is easier to develop complex queries to retrieve data.

#### **Drawbacks of the network database model**

- The structure of the database is not easily modified.
- Changes to the database structure definitely affect application programs that access the database.
- The user has to understand the structure of the database.

### 1.7.4 Relational Database Model

The relational database model is the most popular database model used today. Many improvements have been made to prior database models that simplify data management, data retrieval, and change propagation management. Data is easier to manage, mainly through the use of integrity constraints. The retrieval of data is also a refined process, allowing the user to visualize the database through relational table structures and to ask for specific data without a detailed knowledge of the database layout. Changes are also easier to propagate, thanks to

features such as integrity constraints and the benefits that normalization (reduction of data redundancy) provides.



**Fig 1.7 Relational Model**

### **Benefits of the relational model**

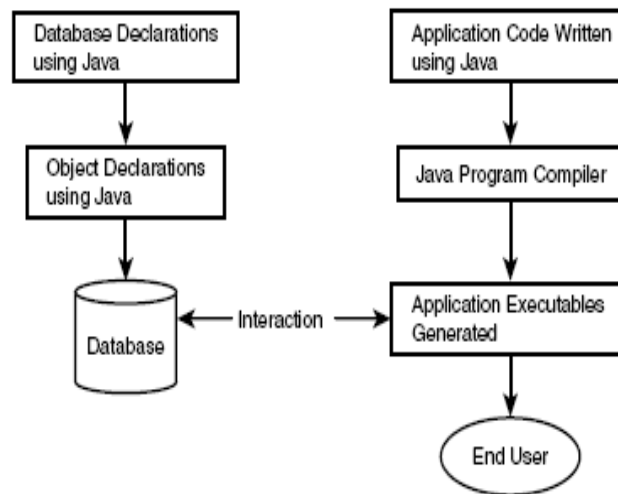
- Data is accessed very quickly.
- The database structure is easy to change.
- The data is represented logically, therefore users need not understand how the data is stored.
- It is easy to develop complex queries to retrieve data.
- It is easy to implement data integrity.
- Data is generally more accurate.
- It is easy to develop and modify application programs.
- A standard language (SQL) has been developed.

### **Drawbacks of the relational database model**

- Different groups of information, or tables, must be joined in many cases to retrieve data.
- Users must be familiar with the relationships between tables.
- Users must learn SQL.

### **1.7.5 Object-Oriented (OO) Database Model**

During the last few years, object-oriented programming has become popular with languages such as C++, Visual Basic, and Java. An OO programming language allows the programmer to work with objects to define an application that interacts with a relational database. An object-oriented database is a database in which data can be defined, stored, and accessed using an OO programming approach. For an OO database, a select OO programming language is used to define the structure of the database as well as create an application through which to interact with the database.



**Fig 1.8 Object oriented Model**

The two basic structures in an OO database are as follows:

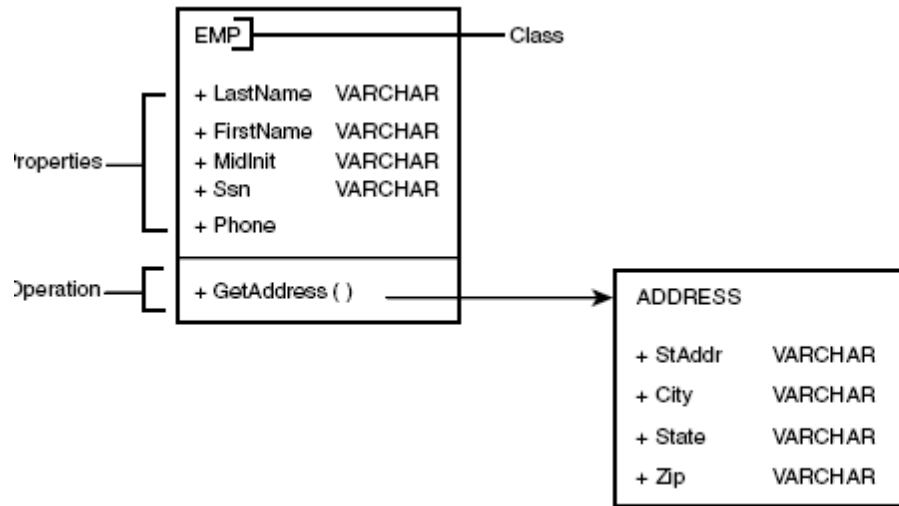
- Objects
- Literals

Objects are structures that have identifiers through which an object can be associated with other objects. Literals are values associated with objects, and have no identifiers. Objects and literals are organized by types, where all elements of a given type have the same set of properties, which can be modified for each individual object. A class is the equivalent of a table in a relational database. Operations are used to retrieve values from other classes, to add values, and to remove values.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit II Batch: 2017-2020



**Fig 1.9 Class and Object**



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit II Batch: 2017-2020

### Features of Object oriented Database Management System

The Object Oriented DBMS has the following features as mandatory for a system to support before it can be called an OODBMS;

#### Feature of Persistence

This feature of OODBMS includes the survival of data as well as persistence should be orthogonal and implicit. Orthogonal implies each object should be persistent as such and the user should not have to explicitly move or copy data to make it persistent. In particular, a database can store, individual objects and the volatile main [memory](#) of an application can contain collections of objects.

#### Able to handle large databases

This feature includes the optimal management of very large databases using techniques like Data clustering, Data buffering, Query optimization, Access path selection and Index management.

#### Controlled Concurrency

This feature guarantees harmonious coexistence among users. Working simultaneously on the database and enjoying controlled sharing. By allowing multiple transactions to run concurrently will improve the performance of the system in terms of increased throughput or improved response time. Ensuring consistency in spite of concurrent execution of transaction require additional effort which is performed by the concurrency controller system of DBMS.

#### Restoring or Data Recovery

This feature indicates the restoration of the system to a state that existed before the software or hardware based crash such as processor or disk failure. The recovery refers to the various strategies and procedures involved in protecting your database against data loss and reconstructing the data such that no data- is lost after failure.

#### Query facility on basis

This feature includes the facility of applying query that should be efficient using query optimization and application independent that can work on any database.

#### Construction of Complex Objects

This feature enables the OODBMS to construct complex objects like tuples sets, lists and arrays from the simple objects like integers, characters, byte strings Boolean and float using the constructors and appropriate operators.

#### Identity of an object

This feature ensures that each object is assigned an Object Identifier (OID) when it is created. Object identity assists OODBMS to uniquely identify an object, thereby automatically providing entity integrity. In fact, as object identity ensures system-wide uniqueness, it provides a stronger constraint than the relational data model's entity integrity, which requires any uniqueness within a relation.



## **KARPAGAM ACADEMY OF HIGHER EDUCATION**

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402      Unit II      Batch: 2017-2020

### **Feature of Classes and types**

This feature supports the notion of classes and types for defining a set of similar objects. Objects that have the same attributes and respond to the same messages can be grouped together to form a class. The attributes and associated methods are defined once for the 'class' rather than separately for each object. The type of variables and expressions help to do the type checking at compile time, to check the correctness of the programs.

### **Property of encapsulation**

This property of OODBMS implies that an object contains both the data structure and the set of operations that can be used to manipulate it. An object is said to encapsulate (hide) data and program. This means that the user cannot see the inside of the object but can use the object by calling the program part of the object.

### **Property of Inheritance**

This property of OODBMS implies that feature of objects by which instances of a class can have access to data and programs contained in a previously defined class, without those definitions being restarted. The different types of inheritance used for reusing the code are substitution inheritance, inclusion inheritance, constraint inheritance and specialization.

### **Property of overriding combined with late binding**

This property of OODBMS implies the ability to use the same message to objects of different classes and have them behave differently. Thus we can define the message "+" for both the addition of numbers and the concatenation (-joining) of characters, even though both these operations are completely different. This feature provides the ability to use the same word to invoke different methods, according to similarity of meaning. Here the late binding is being done as the system cannot bind operation names to programs at compile time and thus, operation names are resolved at run-time.

### **Property of Extensibility**

This property of OODBMS implies that new data types can be built from existing types. The ability to factor out common properties of several classes and form them into a super class that can be shared with subclasses can greatly reduce redundancy within a system. The usage of both system-defined types and user-defined types is the same.

### **Property of Computational Completeness**

This feature of OODBMS implies that it can employ any computable function using the reasonable connectivity to any existing programming language. This feature makes OODBMS more powerful than a database system which only stores and retrieves data and performs simple computations on atomic values.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402      Unit II      Batch: 2017-2020

### **Benefits of the object-oriented model**

- The programmer need only understand OO concepts as opposed to the combination of OO concepts and relational database storage.
- Objects can inherit property settings from other objects.
- Much of the application program process is automated.
- It is theoretically easier to manage objects.
- OO data model is more compatible with OO programming tools.

### **Drawbacks of the object-oriented model**

- Users must learn OO concepts because the OO database does not work with traditional programming methods.
- Standards have not been completely established for the evolving database model.
- Stability is a concern because OO databases have not been around for long.

### **1.7.6 Features of Distributed DBMS**

A distributed database is a logically interrelated collection of shared data (and a description of this data) physically distributed over a [computer](#) network.

Distributed [DBMS](#) is a software system that permits the management of the distributed database and makes the distribution transparent to users.

A Distributed Database Management System (DDBMS) consists of a single logical database that is split into a number of fragments. Each fragment is stored on one or more [computers](#) under the control of a separate DBMS, with the computers connected by a communications network. Each site is capable of independently processing user requests that require access to local data (that is, each site has some degree of local autonomy) and is also capable of processing data stored on other computers in the network. Users access the distributed database via applications. Applications are classified as those that do not require data from other sites (local Applications) and those that do require data from other sites (global applications). We require a DDBMS to have at least one global application.

A DDBMS has the following features:

- A collection of logically related shared data;
- The data is split into a number of fragments;
- Fragments may be replicated;
- Fragments/replicas are allocated to sites;
- The sites are linked by a communications network;
- The data at each site is under the control of a DBMS;
- The DBMS at each site can handle local applications, autonomously;
- Each DBMS participates in at least one global application





## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit II Batch: 2017-2020

### Comparison of DBMS & DDBMS

A database management system, or DBMS, is software that stores, retrieves and updates files from a centralized database. It acts as an intermediary between programs and the database, and allows multiple users or programs to access a data file at once. However, reliability and efficiency issues in larger networks prompted the implementation of a distributed database management system, or DDBMS, in which data files and processing functions are managed through several sites on a computer network.

#### a) Data and Process Distribution

Larger corporations may require an enterprise database to support many users over multiple departments. This would require the implementation of a multiple process, multiple data scenario, or MPMD, in which many computers are linked to a fully distributed client/server DDBMS.

#### b) Reliability

The DDBMS offers more reliability by decreasing the risk of a single-site failure. If one computer in the network fails, the workload is distributed to the rest of the computers. Furthermore, a DDBMS allows replication of data among multiple sites; data from the failed site may still be available at other sites. A centralized DBMS differs because a failed computer that houses the database will debilitate the entire system.

#### c) Transparency

A DDBMS can support three levels of transparency to hide certain complexities from the user, effectively managing the database as if it were centralized. Fragmentation transparency, the highest level of transparency, divides the original database into fragments and disperses them throughout the DDBMS. Therefore, the user does not need to specify fragment names or locations to gain access. Location transparency only requires the user to know the names of the fragments. Local mapping transparency, the lowest level of transparency, requires the user to know the name and location of a fragment.

#### d) Network Expansion

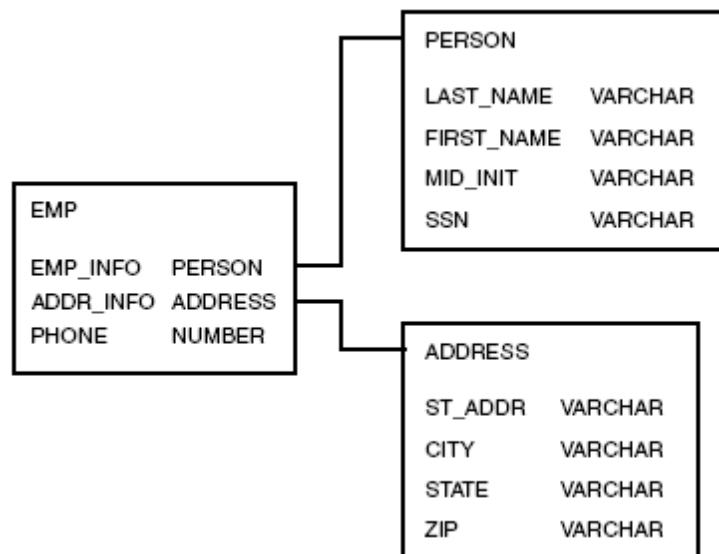
Adding a new site to a DDBMS is easier than in a DBMS. Expanding or modifying a DDBMS occurs on a local level, and does not significantly hinder the operations of the other sites. However, making changes to a DBMS can be time-consuming and complex, since the network is centralized.

#### e) Efficiency

The efficiency of a DDBMS is increased through data localization, which disperses data where it is most often needed to match business requirements. This increases the speed of data access, because the user only has to query a local subset of the database instead of the entire database.

### 1.7.7 Object-Relational (OR) Database Model

Although some rough seams exist between the object-oriented and relational models, the object-relational model was developed with the objective of combining the concepts of the relational database model with object-oriented programming style. The OR model is supposed to represent the best of both worlds (relational and OO), although the OR model is still early in development. As we speak, vendors are implementing OR concepts into their relational databases, as the International Standards Organization (ISO) has integrated OR concepts into the new SQL standard, referred to as SQL3. SQL3 is also referred to as SQL99.



**Fig 1.10 Object Relational Model**

#### Benefits of the object-relational model

- The relational database has more of a 3D architecture.
- User-defined types can be created.

#### Drawbacks of the object-relational model

- The user must understand both object-oriented and relational concepts.
- Some vendors that have implemented OR concepts do not support object inheritance.

### 1.8 Entity-relationship model

An entity-relationship model describes data in terms of the following:

1. Entities
2. Relationship between entities

### 3. Attributes of entities

We graphically display an E-R model using an **entity-relationship diagram**.

An **entity** is an object that exists and which is distinguishable from other objects. An entity can be a person, a place, an object, an event, or a concept about which an organization wishes to maintain data. It is important to understand the distinction between an entity type, an entity instance, and an entity set. An **entity type** defines a collection of entities that have same attributes. An **entity instance** is a single item in this collection. An **entity set** is a set of entity instances. We represent an entity with a set of attributes. An **attribute** is a property or characteristic of an entity type that is of interest to an organization.

Some attributes of common entity types include the following:

STUDENT = {Student ID, SSN, Name, Address, Phone, Email, DOB}

ORDER = {Order ID, Date of Order, Amount of Order}

ACCOUNT = {Account Number, Account Type, Date Opened, Balance}

CITY = {City Name, State, Population}

#### Types of attributes

##### Simple and Composite Attributes

- A **simple** or an **atomic attribute**, such as City or State, cannot be further divided into smaller components.
- A **composite attribute**, however, can be divided into smaller subparts in which each subpart represents an independent attribute

##### Single-Valued and Multi-Valued Attributes

- Most attributes have a single value for an entity instance; such attributes are called **single-valued attributes**.
- A **multi-valued attribute**, on the other hand, may have more than one value for an entity instance.
- we denote a multi-valued attribute with a double-lined ellipse.

##### Stored and Derived Attributes

- The value of a **derived attribute** can be determined by analyzing other attributes.
- An attribute whose value cannot be derived from the values of other attributes is called a **stored attribute**.
- Derived attributes are depicted in the E-R diagram with a dashed ellipse.

##### Key Attribute

- A **key attribute** (or identifier) is a single attribute or a combination of attributes that uniquely identify an individual instance of an entity type. No two instances within an entity set can have the same key attribute value.

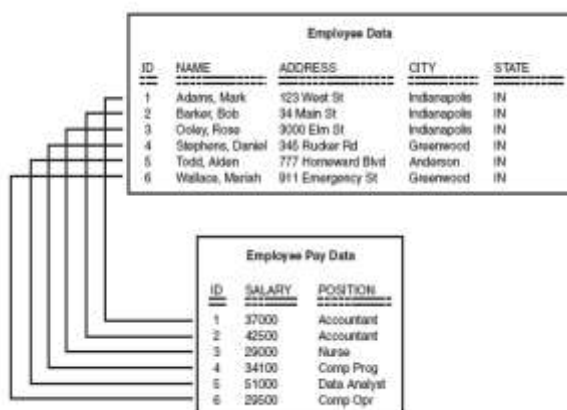
- Sometimes no single attribute can uniquely identify an instance of an entity type. In this case the key attribute, also known as **composite key**, is not a simple attribute, but a composite attribute that uniquely identifies each entity instance.

## 1.9 Relationships

Entities in an organization do not exist in isolation but are related to each other. We define a **relationship** as an association among several entities. A **relationship set** is a grouping of all matching relationship instances, and the term **relationship type** refers to the relationship between entity types. In an E-R diagram, we represent relationship types with diamond-shaped boxes connected by straight lines to the rectangles that represent participating entity types. A relationship type is a given name that is displayed in this diamond-shaped box

### 1.9.1 One-to-One Relationship

A one-to-one relationship represents a relation between entities in which one occurrence of data in one entity might have one occurrence of data in the related entity. Entity A might have only one occurrence of related data in entity B, and entity B might have only one occurrence of related data in entity A. The following figure illustrates a one-to-one relationship, which shows sample data. Notice that all employees listed under Employee Data have a corresponding occurrence of data (record) under Employee Pay Data. It makes sense to track an employee's name, address, and other personal information only one time. It also makes sense that every employee should have a pay record, but only one pay record.

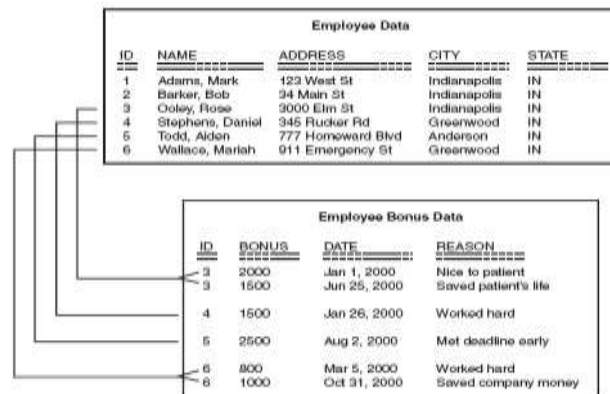


**Fig 1.11 One-One Relationship**

### 1.9.2 One-to-Many Relationship

In most relational databases that we have seen, the one-to-many relationship seems to be the most common relationship that exists. A one-to-many relationship represents a relation between entities in which one occurrence of data in one entity might have one or more occurrences of data in the related entity. For example, entity A might have several occurrences of related data in entity B.

The following figure illustrates a one-to-many relationship, which shows sample data. Here, we have employee data and employee bonus data. Based on an employee's performance, a bonus might be rewarded from time to time. Some employees might have never been issued a bonus, some employees might have been issued a bonus on one occurrence, and some employees might have received multiple bonus checks.



**Fig 1.12 One-Many Relationship**

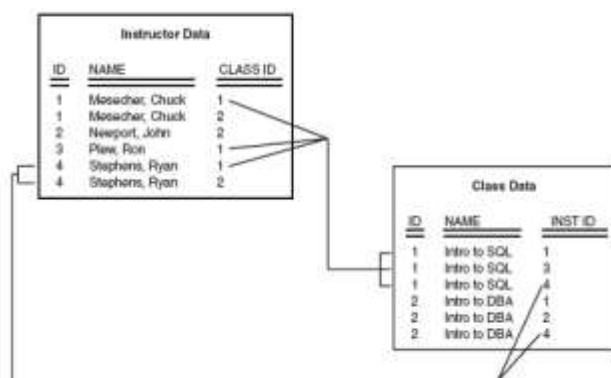
Other examples of one-to-many relationships include the following, where Entity A contains one record and Entity B contains many records per occurrence in Entity A.

### 1.9.3 Many-to-Many Relationship

A many-to-many relationship exists if multiple occurrences of related data are allowed to exist between two entities, in either direction. For instance, entity A might have many occurrences of related data in entity B, and entity B might have many occurrences of related data in entity A.

The following figure illustrates many-to-many relationship, showing sample data in which two basic entities exist for instructor and class data. An instructor might teach many classes, and a class can be taught by many instructors. A class can be taught during the day or in the evening.

Multiple instructors exist as backups to one another, and for scheduling purposes. By studying the relationship between the two entities and sample data in the figure, you can see that Ryan Stephens teaches the Intro to SQL and Intro to DBA classes. If you are looking for the classes a particular instructor teaches, you would look under Instructor Data. If you are looking for instructors who teach a particular class, you would look under Class Data.

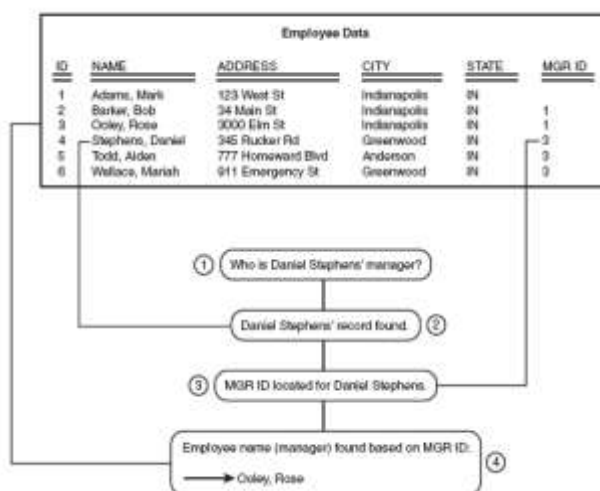


**Fig 1.13 Many-Many Relationship**

### 1.9.4 Recursive Relationships

Sometimes it makes sense to relate data to other data in a single entity. A recursive relationship is a circular relationship that exists between two attributes in the same entity. Recursive relationships are rare, but useful. The most common example used to illustrate a recursive relationship is employee and manager names. Every employee has a manager, who is also an employee.

The figure illustrates a recursive relationship to derive a manager's name from employee data. In this example, we have added an attribute called MGR ID to Employee Data. Notice that every employee has a value associated with MGR ID except for Mark Adams, who happens to be the big cheese. The value associated with MGR ID happens to be a value associated with an occurrence of ID. It is not necessary to store a manager's name separate from employees because a manager must also be an employee. In the figure, we are seeking the Daniel Stephens' manager. First, Daniel Stephens' record must be found. Once found, the value associated with MGR ID is found. The value of MGR ID is used to reference ID. After the matching ID is found, the manager's name is apparent.



### **Fig 1.14 Recursive Relationship**

#### **1.9.5 Mandatory Relationships**

A mandatory relationship represents data that is required to have associated data, or you could say that a relationship must exist. A mandatory relationship typically uses the word must.

Following are examples of one-sided mandatory relationships

- An employee pay record must match an employee personnel record. (An employee pay record cannot exist without a corresponding employee personnel record.)
- An order must be placed by a customer. (Every order must be associated with one customer.)
- An order must correspond to an available product. (Every order must be associated with one product.)
- An author must be associated with one or more publishers.
- A book must be associated with one or more authors and one or more publishers.

#### **1.9.6 Optional Relationships**

An optional relationship does not require a relationship to exist, which means that data might exist that isn't directly associated with data from another entity. An optional relationship typically uses the word may.

Following are examples of one-sided optional relationships:

- A customer may place one or more orders. (A customer may not be required to place an order, but may cease to be considered a customer after a certain period of time with no account activity.)

#### **1.10 How an ERD Is Used**

A complete enterprise system ERD provides a picture of the logical side of your database. Such an ERD is a good planning and integration tool for defining on an enterprise level the overall and potentially shared data requirements for multiple, separate but coexisting information systems within the enterprise.

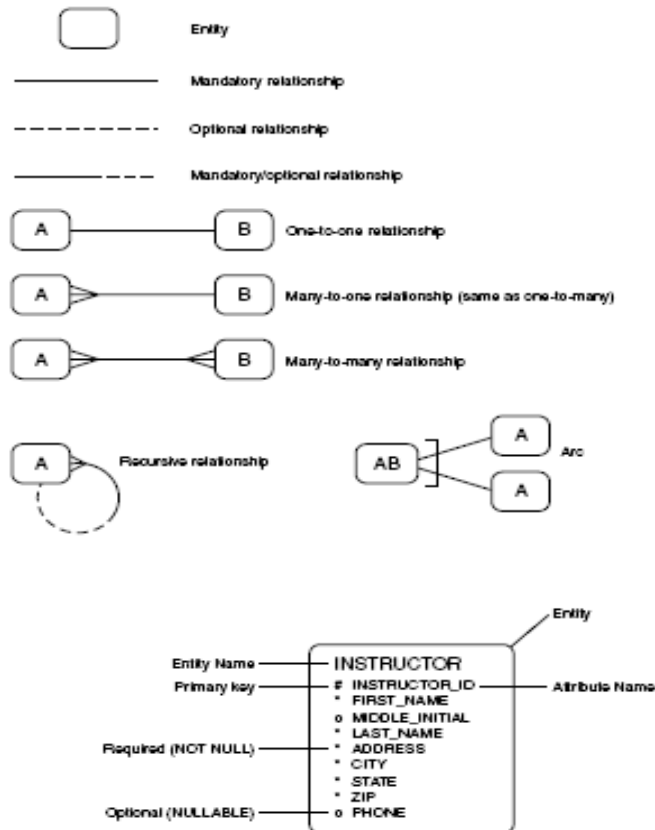
An ERD is also good for showing the scope of data requirements for an individual information system project within the enterprise. Complete system ERDs can be invaluable to the sophisticated user trying to create ad hoc reports or spot potential new or optimal uses for the data this system will capture.

ERD uses can range from simple back-of-the-envelope ERDs used as the basis for communication between developers and between developers and functional users, to ERDs produced by fully integrated GUI components of sophisticated automated design (AD) products such as Oracle's Designer.



### Typical ERD Symbols

The most common symbols used to create an ERD are shown in this section. These symbols have been discussed throughout this chapter in examples that use them. The following Figure shows the symbols most typically used during entity relationship modeling.



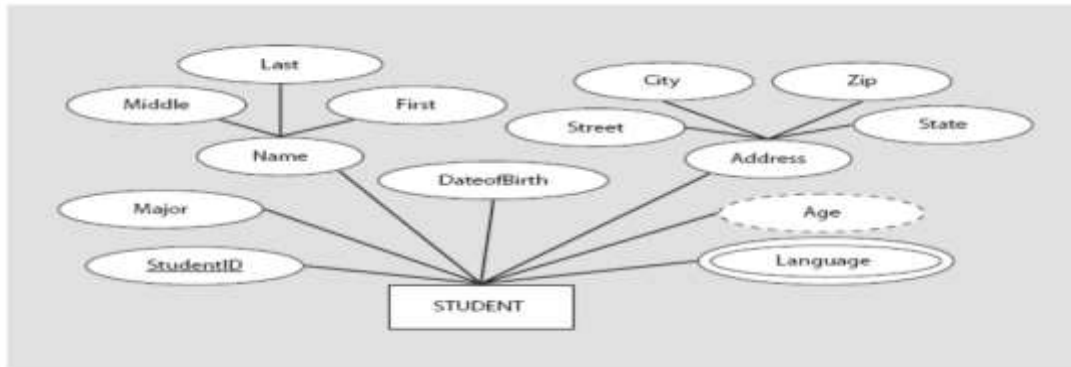
**Fig 1.15 ERD Symbols**

### Cardinalities

The number of entity sets that participate in a relationship is called the **degree of relationship**. The three most common degrees of a relationship in a database are unary (degree 1), binary (degree 2), and ternary (degree 3). Cardinality of a relationship is the count of the number of entities involved in that relationship. For example, if the entity types A and B are connected by a relationship, then the **maximum cardinality** represents the maximum number of instances of entity B that can be associated with any instance of entity A. maximum cardinality refers to only two possible values: one or many.



## Sample ERD



**Fig 1.16 Sample ER Diagram**

E-R diagrams depict an attribute inside an ellipse and connect the ellipse with a line to the associated entity type. The above figure illustrates some of the possible attributes in an E-R diagram for the entity STUDENT. StudentID attribute is the primary key attribute that uniquely identifies a student. So it is underlined. The Age attribute is derived from data of birth. So it is indicated as dotted ellipse. The language attribute has double lined ellipse because it can hold more than one value.

## 2.3 Relational Algebra

### 2.3.1 Basic Operations

Basic operations are the traditional set operations : Union, Difference, Intersection and Cartesian Product.. Three of these four basic operations – union, intersection and difference require that operand relations be union compatible. Two relations are union compatible if they have the same arity and one-one correspondences of the attributes with the corresponding attributes defined over the same domain. The Cartesian product can be defined on any two relations. Two relations P(P) and Q(Q) are said to be union compatible if both P and Q are of the same degree n and the domains of the corresponding n attributes are identical.

Example

P	
ID	Name
101	Jones
103	Smith
104	Lalonde
107	Evan
110	Drew
112	Smith

## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402    Unit II      Batch: 2017-2020

**Q**

ID	Name
103	Smith
104	Lalonde
106	Byron
110	Drew

### UNION ( $\cup$ )

The union of P(P) and Q(Q) is the set theoretic union of P(P) and Q(Q). The resultant relation is  $R=P \cup Q$ . The result relations R contains tuples that are in either P or Q or in both of them. The duplicate tuples are eliminated.

**P  $\cup$  Q**

ID	Name
101	Jones
103	Smith
104	Lalonde
106	Byron
107	Evan
110	Drew
112	Smith

### DIFFERENCE (-)

The difference operation removes common tuples from the first relation.

**P - Q**

ID	Name
101	Jones
107	Evan
112	Smith

### INTERSECTION ( $\cap$ )

The intersection operation selects the common tuples from two relations.

**P  $\cap$  Q**

ID	Name
103	Smith
104	Lalonde
110	Drew



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit II Batch: 2017-2020

### CARTESIAN PRODUCT (X)

The extended Cartesian or simply the cartesian product of two relations is the concatenation of tuples belonging to the two relations. A new resultant relation scheme is created consisting of all possible combinations of tuples.

$$R = P \times Q$$

Example

Personnel (P)

ID	Name
101	Jones
103	Smith
104	Lalonde
106	Byron
107	Evan
110	Drew
112	Smith

Software Packages (S)

S
J1
J2

P X S

ID	Name	S
101	Jones	J1
101	Jones	J2
103	Smith	J1
103	Smith	J2
104	Lalonde	J1
104	Lalonde	J2
106	Byron	J1
106	Byron	J2
107	Evan	J1
107	Evan	J2
110	Drew	J1
110	Drew	J2
112	Smith	J1
112	Smith	J2

The union and intersection operations are associative and commutative.

Example:

From the given 3 relations R(R), S(S) and T(T)

$$R \cup (S \cap T) = (R \cup S) \cap T$$

$$R \cap (S \cup T) = (R \cap S) \cup T$$

The difference operation is non-commutative and non associative.

$$R - S \neq S - R$$

$$R - (S - T) \neq (R - S) - T$$

### 2.3.2 Additional Relational Algebraic operations

The basic set operations which provide a very limited data manipulation facility have been supplemented by the definition of the following operations: projection, selection, join and division. projection and selection are unary operations join and division are binary operations.

Projection ( $\pi$ )

The projection of a relation is defined as a projection of all its tuples over some set of attributes that is it yields a vertical subset of the relation. The projection operation is used to either reduce the number of attributes in the resultant relation or to reorder attributes. In the first case the arity or degree of relation is reduced.

Personnel

ID	Name
101	Jones
103	Smith
104	Lalonde
106	Byron
107	Evan
110	Drew
112	Smith

↓

Name
Jones
Smith
Lalonde
Byron
Evan
Drew
Smith



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit II Batch: 2017-2020

### Selection( $\sigma$ )

This is an operation that selects only some of the tuples of the relation. Such an operation is called selection operation. The projection operation yields a vertical subset of a relation. The action is defined over a subset of the attribute names but over all the tuples in the relation. The selection operation yields a horizontal subset of the given relation that is the action defined is over the complete set of attribute names only a subset of the tuples are included in the result. To have a tuple included in the result relation, the specified selection conditions or predicates must be satisfied by it. The selection operation is represented by the symbol  $\sigma$  and it is sometimes known as restriction operation.

Consider the selection operation

$\sigma \text{ id} < 105 (\text{Personnel})$

The result is

Personnel

ID	Name
101	Jones
103	Smith
104	Lalonde

### Join ()

The join operator allows the combining of two relations to form a single new relation. The tuples from the operand relations that participate in the operation and contribute to the result are related. The join operation allows the processing of relationships existing between the operand relations

Consider the following relations

Assignment(Emp#, Prod#, Job#)

Job\_Function(Job#, title)

Temp = (Assignment  $\bowtie$  Job\_Function)

Two common and very useful variant of join are the equi-join and natural join. In equi-join and natural join the comparison operator is always equality operator(=). But only one of the two sets of domain compatible attributes is retained in the result relation of the natural join.

### Division ( $\div$ )

The division operation is useful when a query involves the phrase “for all objects having all the specified properties”. Both P-Q and Q represent a set of attributes.

Example:



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit II Batch: 2017-2020

Product(Prod#, Prod\_Name, Prod\_details)

Developed\_By(Prod#,Emp#)

Temp=Product÷ Developed\_By

### 2.3.3 Some relational algebra queries

Sample database

Employee

Emp#	Name
101	Jones
103	Smith
104	Lalonde
106	Byron
107	Evan
110	Drew
112	Smith

Assigned To

Proj#	Emp#
COMP453	101
COMP354	103
COMP343	104
COMP354	104
COMP231	106
COMP278	106
COMP353	106
COMP354	106
COMP453	106
COMP231	107
COMP353	107
COMP278	110
COMP353	112
COMP354	112

Project

Proj#	Project_name	Chief_Architect
COMP231	Pascal	101
COMP278	Pascal/Object	103
COMP353	Database	104
COMP354	Operating System	104



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit II Batch: 2017-2020

COMP453	Database	106
---------	----------	-----

### Queries

1. Get Emp# of employees working on project COMP353

Emp#
106
107
112

2. Get details of employees working on project COMP353

Emp#	Name
106	Byron
107	Evan
112	Smith

2. Obtain details of employees working on Database project

Emp#	Name
101	Jones
106	Byron
107	Evan
112	Smith

3. Gather details of employees working on both COMP353 and COMP354

Emp#	Name
106	Byron
112	Smith

4. Find the employee numbers of employees who do not work on project COMP453.

Emp#
106



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402      Unit II      Batch: 2017-2020

**POSSIBLE QUESTIONS**

**UNIT 1**

**PART – A (20 MARKS)**

**(Q.NO 1 TO 20 Online Examinations)**

**Part - B (5X 6 =30 Marks)**  
**(Answer ALL the Questions)**

1. Explain on the concept of relations and their schemas.
2. List out the integrity rules formulated by Codd for the relational model.
3. Discuss the basic operations of relational algebra.
4. What are the additional relational algebraic operations used in relational model.
5. Discuss the characteristics of relations.
6. What are the three main categories of constraints in databases?
7. Write short notes on relational database and its schema.
8. Define Data Definition Language in SQL and explain the different statements used in DD2.
9. Different DML in SQL. Explain the different statements used in DML.
10. Discuss the basic constraints that are specified in SQL.
11. Write a short note on the basic retrieval Queries in SQL.

**Part - C (10X1 =10 Marks)**  
**(Compulsory Question)**





## **KARPAGAM ACADEMY OF HIGHER EDUCATION**

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402      Unit II      Batch: 2017-2020

1. Define the term locking. Explain the different types of locking used in concurrency control.
2. Explain in detail the 13 object oriented database management system Commandments.
3. Compare and contrast the traditional and object oriented DBMS.
4. Elaborate different Data Manipulation Language in SQL. Explain the different statements used in DML.
5. Discuss on functional dependencies of relational database design in detail with a suitable example.

Subject : ]

S.No
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

51
52
53
54
55
56
57
58
59
60
61
62

# KARPAGAM ACADEMY OF HIGHER EDUCATION

(Established Under Section 3 of UGC Act 1956)

COIMBATORE – 641 021

## DEPARTMENT OF COMPUTER APPLICATIONS

Batch 2017– 2020

### Database Management Systems

### UNIT II

Questions
_____expresses basic semantic properties inherent to a model
A data base state is said to be consistent if the database satisfies a set of constraints called _____
_____regulates the application behaviour
Integrity constraints should be manipulated by the database administrator using _____ language
how many basic methods permits the rejection of in consistent updates
_____assertion to be expressed in tuple relational calculus
single relation and single variable assertions is called
multirelation and multivariable constraints such as _____
_____ is a special processing because of the cost of evaluating the aggregates
The user from query optimization a time consuming task is best handled by_____
calculus query must be decomposed into a sequence of relational operations called_____
The data accessed by the query must be_____ so that the operations on relations are translated to bear on local data
high level query to lowlevel query is the main function of_____
A good measure of resource consumption is the _____ that will be incurred in processing the query
Which directly affects their execution time, dictates some principles useful to a query processor
_____command is used to remove the table definition information
_____is used to modify the structure of an existing table.

View can be dropped using _____command
_____columns are not allowed to contain null values
_____are valuable to give the security to our original table
_____clause is used to modify a particular row.
_____is a condition specified on a database schema & restricts the data that can be stored in an instance of the database.
A set of fields that uniquely identifies a tuple according to a key constraint is called_____ for the relation
_____is used to refer the primary key in another entity
_____value is unknown or not applicable
_____statement is used to define a new table.
Rows are inserted using the _____command
rows are deleted using the _____command
Modify the column values in an existing row using_____ command
_____command is used to remove the table definition information
_____is used to modify the structure of an existing table.
View can be dropped using _____command
Duplicates are eliminated by using _____ keyword.
Group function is otherwise known as _____
Pattern matching has done through_____operator.
Which keyword is used to check if an element is in a given set?
Any two tables that are Union-Compatible that is, have the same number of _____
_____keyword is used to eliminates the duplicates
_____keyword is used to retain the duplicates
_____is a query that has another query embedded within it.
_____is used to calculate the number of values in the Column.
_____is used to calculate the sum of all values in the column
_____is used to calculate the average of all values in the column
Which function is used to extract the maximum values in the relations?
Which function is used to extract the maximum values in the relations?
Which clause is used, when we are using Group by clause, instead of where clause?
_____is used when the column value is either unknown or inapplicable.
_____keyword specifies that the join condition is equality on all common attributes and the where clause is not required.
_____constraints for a single table.
_____keyword is used to assign a default value with a domain.

Which constraint is used to check the ranges in the column values?
_____ operator is another set comparison operator such as IN.
Which keyword is similar to NOT IN?
An _____ consists of files, both data files and Oracle system files.
A _____ consists of a number of bytes of disk space in the operating system's storage system
An _____ is two or more contiguous Oracle data blocks
A _____ is a set of extents that you allocate to a logical structure like a table or an index
A _____ is a set of one or more data files, and usually consists of related segments.
The smallest logical component of an Oracle database is the _____
The _____ of data blocks contains the data stored in the tables or their indexes.
The _____ is a purely logical construct and is the primary logical storage structure of an Oracle database.
_____ tablespaces are the default in Oracle Database 10g.

**Subject Code: 16CAU402**

<b>opt1</b>	<b>opt2</b>	<b>opt3</b>	<b>opt4</b>	<b>Answer</b>
behavioural constraints	structural constraints	constraints	concepts	structural constraints
distributed authorization	database consis	semantic integrity	security	semantic integrity
behavioural constraints	structural constraints	constaints	concepts	behavioural constraints
high level	lowlevel	structured	conceptual	high level
1	2	3	4	2
Integrity	individual	set oriented	privacy	Integrity
individual	set oriented	assertion	taxonomy	individua
primary key	foreign key	secondary	unique key	primary key
assertion involving	setorinted	individual	group	assertion involving
query processor	optmization	fragments	processor	query processor
algebraic query	query located	fragments	process	algebraic query
algebraic query	query localized	localized	fragments	localized
calculus query	decomposed	algebraic query	query processor	query processor
cost	total cost	I/o and cpu cost	cpu cost	total cost
complexiy relational algebra	relational algebra	binary	implementation	complexiy relational algebra
Delete	Remove	Destroy	Drop	Drop
Modify	Alter	Change	Recreate	Alter



Delete View	Remove View	Replace View	Drop View	Drop View
Primary Key	Candidate key	foreign Key	Unique Key	Primary Key
Original table	Duplicate table	Views	Tables	Views
Update	Modify	Alter	Where	Where
integrity Constraint	restriction	key	check	integrity Constraint
primary key	Candidate key	foreign key	Super key	Candidate key
Candidate key	referential key	foreign key	Primary key	foreign key
not null	zero	unknown	null	null
Create	Produce	Insert	Add	Create
Create	Insert	Add	Make	Insert
Delete	drop	remove	alter	Delete
Modify	Alter	Update	Change	Update
Delete	Remove	Destroy	Drop	Drop
Modify	Alter	Change	Recreate	Alter
Delete View	Remove View	Replace View	Drop View	Drop View
Remove	Distinct	RM	Redundant	Distinct
Collection	Aggregate	function	Count	Aggregate
Comparison	arithmetic	Logical	Aggregate	Aggregate
not	in	not exist	except	in
Columns	rows	Columns and rows	null values	Columns
Union	Union all	Intersect all	Except all	Union
Union	Union all	Intersect	Except	Union
Query	Subquery	QBE	QUEL	Subquery
Count	aggregate	Cal	Calculate	Count
Total	Sum	Count	Collection	Sum
Total	Sum	Average	avg	avg
max	maximum	excess	large	max
Small	minimum	lower	min	min
where	Having	Distinct	Group	Having
zero	all	Empty	null	null
full outerjoin	left outer join	Natural	Right outerjoin	Natural
Assertion	table constraints	default	union	table constraint
Static	Default	Permanent	distinct	Default

range	verify	check	condition	check
Exists	IN	avail	present	Exists
Exist	IN	Not EXIST	Except	Except
production database	test databases	Oracle database	execution database	Oracle dat
extends	data block	segments	tablespaces	data block
extends	data block	segments	tablespaces	extends
extends	data block	segments	tablespaces	segments
extends	data block	segments	tablespaces	tablespace
extends	data block	segments	tablespaces	data block
free space section	row data section	database section	segment section	free space
extends	data block	segments	tablespaces	tablespace
locally managed	dictionary managed	database managed	uniform managed	locally ma





abase

s

section  
s

naged

### UNIT III

Conventional Data Models & Systems - Network, Data Model & IDMS Systems - Membership types & options in a set - DML for the network model - Navigation within a network database - Hierarchical Data Model & IMS System - Hierarchical Database structure - HSAM, HISAM, HDAM & HIDAM organization - DML for hierarchical model - Overview of IMS

### OVERVIEW OF CONVENTIONAL DATA MODELS:

#### **Hierarchical Data Model:**

One of the most important applications for the earliest database management systems was production planning for manufacturing companies. If an automobile manufacturer decided to produce 10,000 units of one car model and 5,000 units of another model, it needed to know how many parts to order from its suppliers. To answer the question, the product (a car) had to be decomposed into assemblies (engine, body, chassis), which were decomposed into subassemblies (valves, cylinders, spark plugs), and then into sub-subassemblies, and so on. Handling this list of parts, known as a bill of materials, was a job tailor-made for computers. The bill of materials for a product has a natural hierarchical structure. In this model, each record in the database represented a specific part. The records had parent/child relationships, linking each part to its subpart, and so on.

To access the data in the database, a program could:

- find a particular part by number (such as the left door),
- move "down" to the first child (the door handle),
- move "up" to its parent (the body), or
- move "sideways" to the next child (the right door).

Retrieving the data in a hierarchical database thus required navigating through the records, moving up, down, and sideways one record at a time.

One of the most popular hierarchical database management systems was IBM's Information Management System (IMS), first introduced in 1968. The advantages of

IMS and its hierarchical model are as follows:

- **Simple structure:** The organization of an IMS database was easy to understand. The database hierarchy paralleled that of a company organization chart or a family tree.

- **Parent/child organization:** An IMS database was excellent for representing parent/child relationships, such as "A is a part of B" or "A is owned by B."
- **Performance:** IMS stored parent/child relationships as physical pointers from one data record to another, so that movement through the database was rapid. Because the structure was simple, IMS could place parent and child records close to one another on the disk, minimizing disk input/output.

IMS is still a very widely used DBMS on IBM mainframes. Its raw performance makes it the database of choice in high-volume transaction processing applications such as processing bank ATM transactions, verifying credit card numbers, and tracking the delivery of overnight packages. Although relational database performance has improved dramatically over the last decade, the performance requirements of applications such as these have also increased, insuring a continued role for IMS.

### **Network Data Model:**

The simple structure of a hierarchical database became a disadvantage when the data had a more complex structure. In an order-processing database, for example, a single order might participate in three different parent/child relationships, linking the order to the customer who placed it, the salesperson who took it, and the product ordered. The structure of this type of data simply didn't fit the strict hierarchy of IMS.

To deal with applications such as order processing, a new network data model was developed. The network data model extended the hierarchical model by allowing a record to participate in multiple parent/child relationships.

For a programmer, accessing a network database was very similar to accessing a hierarchical database. An application program could:

- find a specific parent record by key (such as a customer number),
- move down to the first child in a particular set (the first order placed by this customer),
- move sideways from one child to the next in the set (the next order placed by the same customer), or
- move up from a child to its parent in another set (the salesperson who took the order).

Once again the programmer had to navigate the database record-by-record, this time specifying which relationship to navigate as well as the direction.

### **Network databases had several advantages:**

- **Flexibility:** Multiple parent/child relationships allowed a network database to represent

data that did not have a simple hierarchical structure.

- **Standardization:** The CODASYL standard boosted the popularity of the network model, and minicomputer vendors such as Digital Equipment Corporation and Data General implemented network databases.
- **Performance:** Despite their greater complexity, network databases boasted performance approaching that of hierarchical databases. Sets were represented by pointers to physical data records, and on some systems, the database administrator could specify data clustering based on a set relationship.

Network databases had their disadvantages, too. Like hierarchical databases, they were very rigid. The set relationships and the structure of the records had to be specified in advance. Changing the database structure typically required rebuilding the entire database.

## Network Model

In the relational model, the data and the relationships among data are represented by a collection of tables. The network model differs from the relational model in that data are represented by collections of *records*, and relationships among data are represented by *links*.

## Basic Concepts

A network database consists of a collection of records connected to one another through links. A record is in many respects similar to an entity in the E-R model. Each record is a collection of fields (attributes), each of which contains only one data value. A link is an association between precisely two records. Thus, a link can be viewed as a restricted (binary) form of relationship in the sense of the E-R model.

As an illustration, consider a database representing a *customer-account* relationship in a banking system. There are two record types, *customer* and *account*. As we saw earlier, we can define the *customer* record type, using Pascal-like notation:

```
type customer = record  
customer name: string;  
customer street: string;  
customer city: string;  
end
```

The *account* record type can be defined as



```
type account = record
  account number: string;
  balance: integer;
end
```

The sample database in Figure D.1 shows that Hayes has account A-102, Johnson has accounts A-101 and A-201, and Turner has account A-305.

## Data-Structure Diagrams

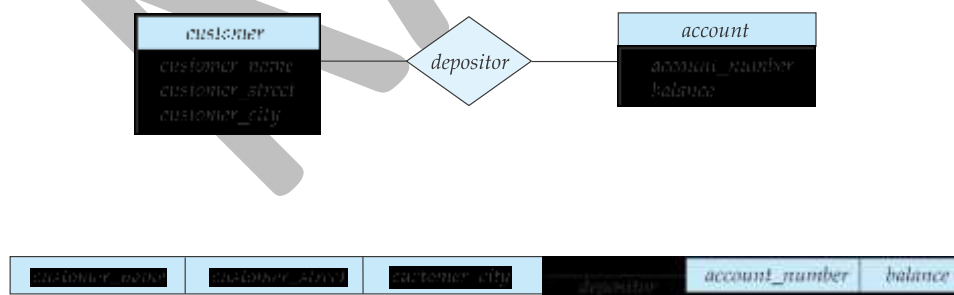
A *data-structure diagram* is a schema representing the design of a network database. Such a diagram consists of two basic components:

1. **Boxes**, which correspond to record types
2. **Lines**, which correspond to links

A data-structure diagram serves the same purpose as an E-R diagram; namely, it specifies the overall logical structure of the database. So that you will understand how such diagrams are structured, we shall show how to transform E-R diagrams into their corresponding data-structure diagrams.

## Binary Relationship

Consider the E-R diagram of Figure D.2a, consisting of two entity sets, *customer* and *account*, related through a binary, many-to-many relationship *depositor*, with no descriptive attributes. This diagram specifies that a customer may have several accounts, and that an account may belong to several different customers. The corresponding data-structure diagram appears in Figure D.2b. The record type *customer* corresponds to the entity set *customer*. It includes three fields —



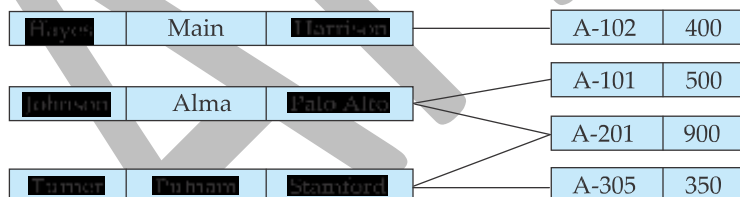
**Figure D.2** E-R diagram and its corresponding data-structure diagram.

*customer-name*, *customer street*, and *customer city* — as defined in Section D.1. Similarly, *account* is the record type corresponding to the entity set *account*. It includes the two fields

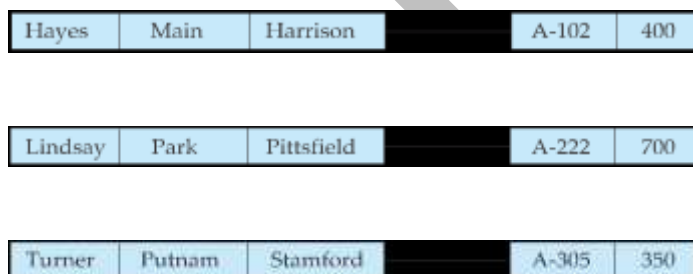
*account number* and *balance*. Finally, the relationship *depositor* has been replaced with the link *depositor*.

The relationship *depositor* is many to many. If the relationship *depositor* were one to many from *customer* to *account*, then the link *depositor* would have an arrow pointing to *customer* record type (Figure D.3a). Similarly, if the relationship *depositor* were one to one, then the link *depositor* would have two arrows: one pointing to *account* record type and one pointing to *customer* record type (Figure D.3b). Since, in the E-R diagram of Figure D.2a, the *depositor* relationship is many to many, we draw no arrows on the link *depositor* in Figure D.2b.

A database corresponding to the described schema may thus contain a number of *customer* records linked to a number of *account* records. A sample database corresponding to the data-structure diagram of Figure D.2 appears in Figure D.4. Since the relationship is many to many, we show that Johnson has accounts A-101 and A-201 and that account A-201 is owned by both Johnson and Smith. A sample database corresponding to the data-structure diagram of Figure D.3a is depicted in Figure D.1. Since the relationship is one to many from *customer* to *account*, a customer may have more than one account, as Johnson does — she owns both A-101 and A-201. An *account*, however, cannot belong to more than one customer, and the database observes this restriction. Finally, a sample database corresponding to the data-structure diagram of Figure D.3b is shown in Figure D.5. Since the relationship is one to one, an account can be owned by precisely one customer, and a customer can have only one account; the sample database follows those rules.



**Figure D.4** Sample database corresponding to diagram of Figure D.2b.

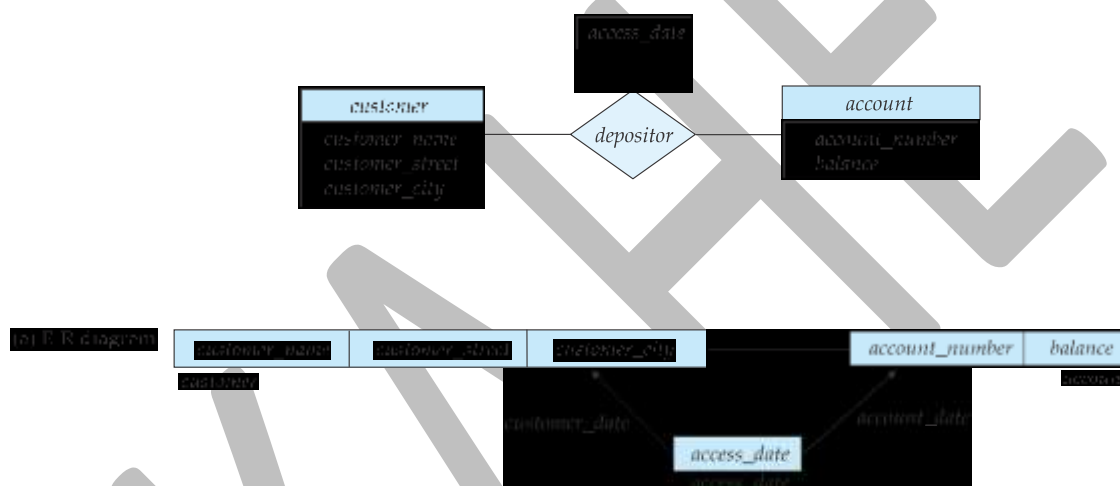


**Figure D.5** Sample database corresponding to diagram of Figure D.3b.

If a relationship includes descriptive attributes, the transformation from an E-R diagram to a data-structure diagram is more complicated. A link cannot contain any data value, so a new record type needs to be created and links need to be established.

Consider the E-R diagram of Figure D.2a. Suppose that we add the attribute *access date* to the relationship *depositor*, to denote the most recent time that a customer accessed the account. This newly derived E-R diagram appears in Figure D.6a. To transform this diagram to a data-structure diagram, we must

1. Replace entities *customer* and *account* with record types *customer* and *account*, respectively.
2. Create a new record type *access date* with a single field to represent the date.
3. Create the following many-to-one links:



**Figure D.6** E-R diagram and its corresponding network diagram.

Hayes	Main	Harrison	10 June 2009	A-102	400
Johnson	Alma	Palo Alto	24 May 2009	A-201	900
			17 June 2009		
Turner	Putnam	Stamford	28 May 2009	A-305	350

**Figure D.7** Sample database corresponding to diagram of Figure D.6b.

- *customer date* from the *access date* record type to the *customer* record type

- *account date* from the *access date* record type to the *account* record type

The resulting data-structure diagram appears in Figure D.6b.

An instance of a database corresponding to the described schema appears in Figure D.7. It shows that:

- Account A-201 is held by Johnson alone, and was last accessed by her on 17 June.
- Account A-305 is held by Turner alone, and was last accessed by him on 28 May.
- Account A-102 is held by both Hayes and Johnson. Hayes accessed it last on 10 June, and Johnson accessed it last on 24 May.

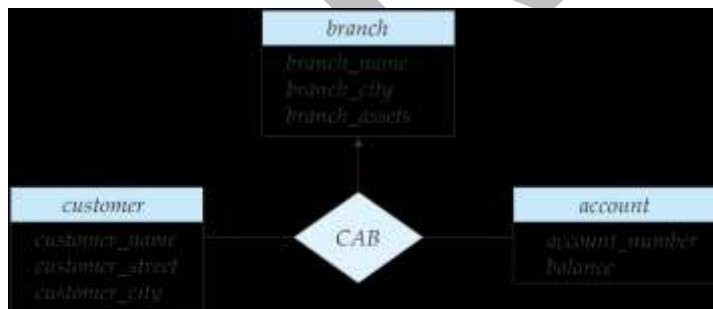
## D.2.2 General Relationships

Consider the E-R diagram of Figure D.8a, which consists of three entity sets — *account*, *customer*, and *branch* — related through the general relationship *CAB* with no descriptive attribute.

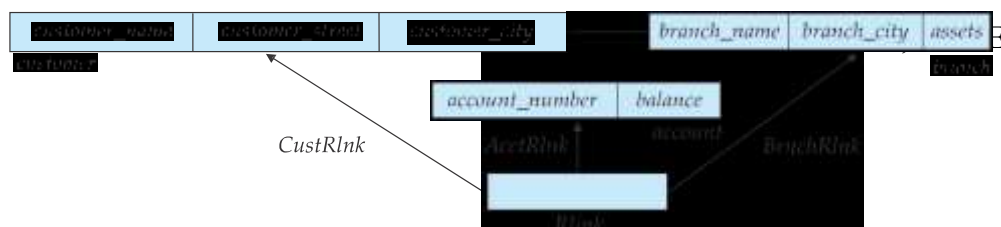
Since a link can connect precisely two different record types, we need to connect these three record types through a new record type that is linked to each of them directly.

To transform the E-R diagram of Figure D.8a to a network data-structure diagram, we need to do the following:

1. Replace entity sets *account*, *customer*, and *branch* with record types *account*, *customer*, and *branch*, respectively.
2. Create a new record type *Rlink* that may either have no fields or have a single field containing a unique identifier. The system supplies this identifier, and the application program does not use it directly. This new type of record is sometimes referred to as a *dummy* (or *link* or *junction*) record type.



(a) E-R diagram



**Figure D.8** E-R diagram and its corresponding data-structure diagram.

### 3. Create the following many-to-one links:

- *CustRlnk* from *Rlink* record type to *customer* record type
- *AcctRlnk* from *Rlink* record type to *account* record type
- *BrncRlnk* from *Rlink* record type to *branch* record type

## The DBTG CODASYL Model

The first database-standard specification, called the CODASYL DBTG 1971 report, was written in the late 1960s by the Database Task Group. Since then, a number of changes have been proposed many of which are reflected in our discussion concerning the DBTG model.

### D.3.1 Link Restriction

In the DBTG model, only many-to-one links can be used. Many-to-many links are disallowed to simplify the implementation. We represent one-to-one links using a many-to-one link. These restrictions imply that the various algorithms of Section D.2 for transforming an E-R diagram to a data-structure diagram must be revised.

Consider a binary relationship that is either one to many or one to one. In this case, the transformation algorithm defined in Section D.2.1 can be applied directly. Thus, for our customer-account database, if the *depositor* relationship is one to many with no descriptive attributes, then the appropriate data-structure diagram is as shown in Figure D.10a. If the relationship has a descriptive attribute (for example, *access-date*), then the appropriate data-structure diagram is as shown in Figure D.10b.

If the *depositor* relationship, however, is many to many, then our transformation algorithm must be refined; if the relationship has no descriptive attributes (Figure D.11a), then this algorithm must be employed:

1. Replace the entity sets *customer* and *account* with record types *customer* and *account*, respectively.
2. Create a new dummy record type, *Rlink*, that may either have no fields or have a single field containing an externally defined unique identifier.
3. Create the following two many-to-one links:
  - *CustRlnk* from *Rlink* record type to *customer* record type
  - *AcctRlnk* from *Rlink* record type to *account* record type

The corresponding data-structure diagram is as shown in Figure D.11b. An instance of a database corresponding to the described schema appears in Figure D.12. We encourage you to compare this sample database with the one described in Figure D.4.

If the relationship *depositor* is many to many with a descriptive attribute (for example, *access date*), then the transformation algorithm is similar to the one described. The only difference is that the new record type *Rlink* now contains the field *access date*.

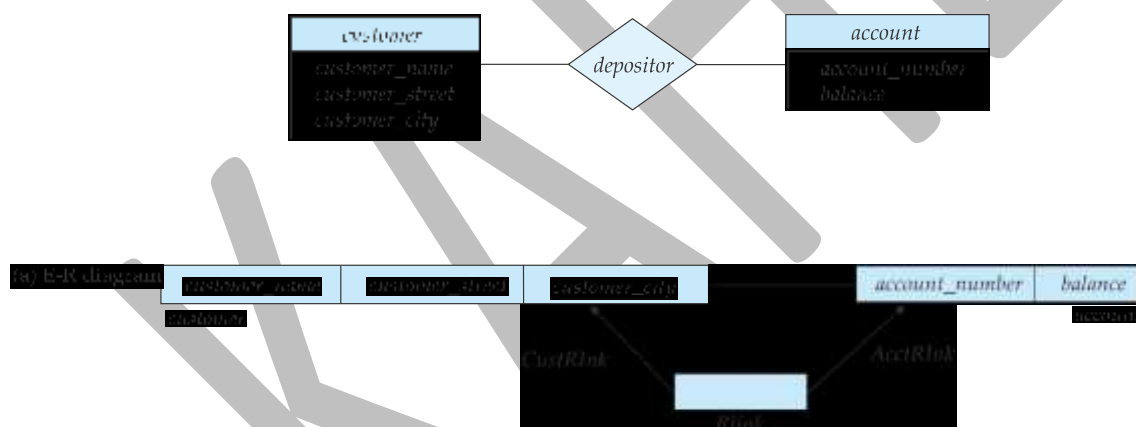


Figure D.11 E-R diagram and its corresponding data-structure diagram.

**Figure D.11** E-R diagram and its corresponding data-structure diagram.

Given that only many-to-one links can be used in the DBTG model, a data-structure diagram consisting of two record types that are linked together has the general form of Figure D.13. This structure is referred to in the DBTG model as a DBTG set. The name of the set is usually chosen to be the same as the name of the link connecting the two record types. In each such DBTG set, the record type A is designated as the owner (or parent) of the set, and the record type B is designated as the member (or child) of the set. Each DBTG set can have any number of set occurrences — that is, actual instances of linked records. For example, in Since many-to-many links are disallowed, each set occurrence has precisely one owner, and has zero or more member records. In addition, no member record of a set can participate in more than one occurrence of the set at any point. A member record, however, can participate simultaneously in several set occurrences of different DBTG sets. There are two DBTG sets:

1. *depositor*, which has *customer* as the owner of the DBTG set, and *account* as the member of the DBTG set
2. *account branch*, which has *branch* as the owner of the DBTG set, and *account* as the member of the DBTG set

The set *depositor* can be defined as follows:

**set name is depositor owner is customer member is account**

The set *account branch* can be defined similarly:

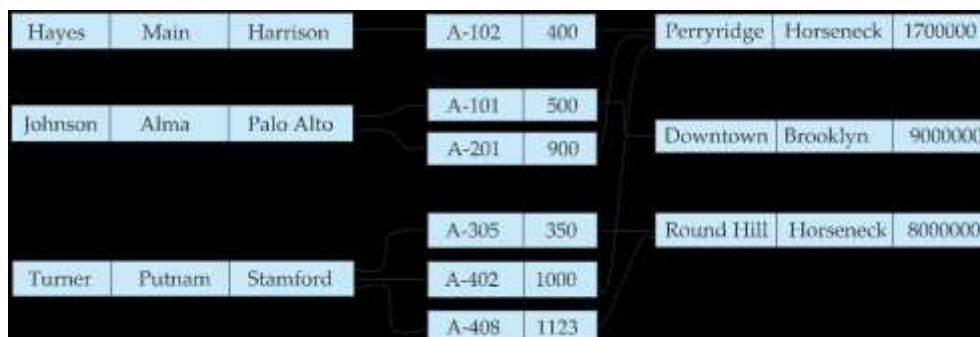
**set name is account branch owner is branch member is account**

An instance of the database appears in Figure D.16. There are six set occurrences listed next: three of set *depositor* (sets 1, 2, and 3), and three of set *account branch* (sets 4, 5, and 6).

1. Owner is *customer* record Hayes, with a single member *account* record A-102.
2. Owner is *customer* record Johnson, with two member *account* records A-101 and A-201.
3. Owner is *customer* record Turner, with three member *account* records A-305, A-402, and A-408.







**Figure D.16** Six set occurrences.

- Owner is *branch* record Perryridge, with three member *account* records A-102, A-201, and A-402.
- Owner is *branch* record Downtown, with one member *account* record A-101.
- Owner is *branch* record Round Hill, with two member *account* records A-305 and A-408.

Note that an *account* record (which is, in this case, a member of both DBTG sets) cannot appear in more than one set occurrence of one individual set type. This restriction exists because an account can belong to exactly one customer, and can be associated with only one bank branch. An account, however, can appear in two set occurrences of different set types. For example, account A-102 is a member of set occurrence 1 of type *depositor*, and is also a member of set occurrence 4 of type *account branch*.

The member records of a set occurrence can be ordered in a variety of ways. We shall discuss this issue in greater detail in Section D.6.6, after we describe the mechanism for inserting and deleting records into a set occurrence.

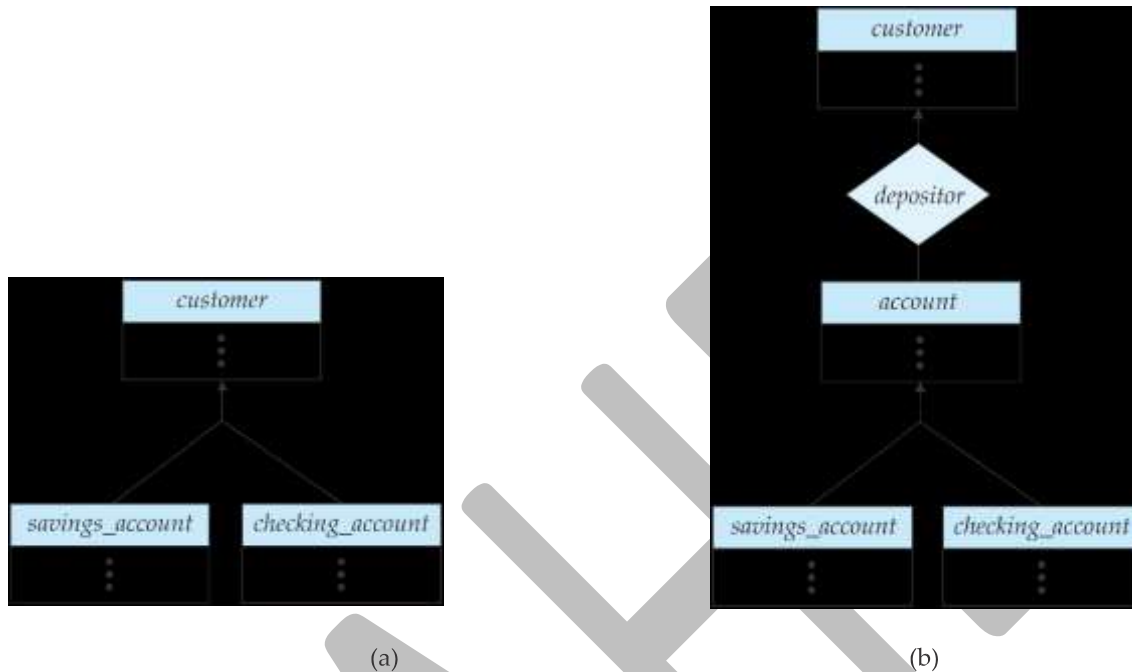
The DBTG model allows more complicated set structures, in which one single owner type and several different member types exist. For example, suppose that we have two types of bank accounts: checking and saving. Then, the data-structure diagram for the customer-account schema is as depicted in Figure D.17a. Such a schema is similar in nature to the E-R diagram of Figure D.17b.

The DBTG model also provides for the definition of a special kind of set, referred to as a *singular set* (or *system set*). In such a set, the owner is a system-defined, unique record type, called *system*, with no fields. Such a set has a *single* set occurrence. This scheme is useful in searching records of one particular type, as we shall discuss in Section D.4.4.

### D.3.3 Repeating Groups

The DBTG model provides a mechanism for a field (or collection of fields) to have a set of values, rather than one single value. For example, suppose that a customer





**Figure D.17** Data-structure and E-R diagram.

has several addresses. In this case, the *customer* record type will have the (*street*, *city*) pair of fields defined as a repeating group. Thus, the *customer* record for Turner may be as in Figure D.18.

The repeating-groups construct provides another way to represent the notion of weak entities in the E-R model. As an illustration, let us partition the entity set *customer* into two sets:

1. *customer*, with descriptive attribute *customer name*
2. *customer address*, with descriptive attributes *customer street* and *customer city*

The *customer address* entity set is a weak entity set, since it depends on the strong entity set *customer*.

The E-R diagram describing this schema appears in Figure D.19a. If we do not use the repeating-group construct in the schema, then the corresponding data-

### The Find and Get Commands

The two most frequently used DBTG commands are

- **find**, which locates a record in the database and sets the appropriate currency pointers

- **get**, which copies the record to which the current of run unit points from the database to the appropriate program work area template

Let us illustrate the general effect that the **find** and **get** statements have on the program work area. Consider the sample database of Figure D.16. Suppose that the current state of the program work area of a particular application program is as shown in Figure D.20. Further suppose that a **find** command is issued to locate the customer record belonging to Johnson. This command causes the following changes to occur in the state of the program work area:

- The current of record type *customer* now points to the record of Johnson.
- The current of set type *depositor* now points to the record of Johnson.
- The current of run unit now points to *customer* record Johnson.

If the **get** command is executed, the result is that the information pertaining to Johnson is loaded into the *customer* record template.

#### D.4.3 Access of Individual Records

The **find** command has a number of forms. We shall present only a few of these commands in this appendix. There are two different **find** commands for locating individual records in the database. The simplest command has the form

**find any** <record type> **using** <record-field>

This command locates a record of type <record type> whose <record-field> value is the same as the value of <record-field> in the <record type> template in the program work area. Once the system finds such a record, it sets the following currency pointers to point to that record:

- The current of run-unit pointer
- The record-type currency pointer for <record type>
- The set currency pointer for every set in which <record type> is either the owner type or member type.

As an illustration, let us construct the DBTG query that prints the street address of Hayes:

```
customer.customer name := "Hayes";  
find any customer using customer name;
```



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit III

Batch: 2017-2020

```
get customer;  
print (customer.customer street);
```

There may be several records with the specified value. The **find** command locates the first of these in some prespecified ordering (see Section D.6.6). To locate other database records that match the <record-field>, we use the command

**find duplicate** <record type> **using** <record-field>

which locates (according to a system-dependent ordering) the next record that matches the <record-field>. The currency pointers noted previously are affected.

As an example, let us construct the DBTG query that prints the names of all the customers who live in Harrison:

```
customer.customer city := "Harrison"; find any customer using customer city; while DB-status  
=0 do  
begin  
get customer;  
print (customer.customer name);  
find duplicate customer using customer city;  
end;
```

We have enclosed part of the query in a **while** loop, because we do not know in advance how many such customers exist. We exit from the loop when DB-status = 0. This action indicates that the most recent **find duplicate** operation failed, implying that we have exhausted all customers residing in Harrison.

### D.4.4 Access of Records within a Set

The previous **find** commands located *any* database record of type <record type>. In this subsection, we concentrate on **find** commands that locate records in a particular DBTG set. The set in question is the one that is pointed to by the <set- type> currency pointer. There are three different types of commands. The basic **find** command is

**find first** <record type> **within** <set-type>

which locates the first member record of type <record type> belonging to the current occurrence of <set-type>. The various ways in which a set can be ordered are discussed in Section D.6.6. To step through the other members of type <record type> belonging to the set occurrence, we repeatedly execute the following command:

**find next** <record type> **within** <set-type>

The **find first** and **find next** commands need to specify the record type since a DBTG set can have members of different record types.

As an illustration of how these commands execute, let us construct the DBTG query that prints the total balance of all accounts belonging to Hayes.

```
sum := 0;
customer.customer name := "Hayes";
find any customer using customer name;
find first account within depositor;
while DB-status = 0 do
begin
  get account;
  sum := sum + account.balance;
  find next account within depositor;
end
print (sum);
```

Note that we exit from the **while** loop and print out the value of *sum* only when the DB-status is set to a value not equal to zero. Such a nonzero value results after the **find next** operation fails, indicating that we have exhausted all the members of a set occurrence of type *depositor*, whose owner is the record of customer Hayes.

The previous **find** commands locate member records within a particular DBTG set occurrence. There are many circumstances, however, under which it may be necessary to locate the owner of a particular DBTG set occurrence. We can do so through the following command:

**find owner within** <set-type>



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit III

Batch: 2017-2020

The set in question is  $\langle \text{set-type} \rangle$ . Note that, for each set occurrence, there exists precisely one single owner.

As an illustration, consider the DBTG query that prints all the customers of the Perryridge branch:

```
branch.branch name := "Perryridge";  
find any branch using branch name;  
find first account within account branch;  
while DB-status =0 do  
begin  
  find owner within depositor;  
  get customer;  
  print (customer.customer name);  
  find next account within account branch;  
end
```

Note that, if a customer has several accounts in the Perryridge branch, then his name will be printed several times.

As a final example, consider the DBTG query that prints the names of all the customers of the bank. Such a query cannot be formed easily with the mechanism that we have described thus far, since no one single set has all the customer records.

as its members. The remedy is to define a singular set (Section D.3.2) consisting of members of type *customer*. This set is defined as follows:

**set name is** *AllCust* **owner is** system **member is** *customer*

Once such a set has been defined, we can form our query as follows:

```
find first customer within AllCust;  
while DB-status =0 do  
begin  
  get customer;  
  print (customer.customer name);  
  find next customer within AllCust;  
end
```

#### D.4.5 Predicates

The **find** statements that we have described allow the value of a field in one of the record templates to be matched with the corresponding field in the appropriate database records. Although, with this technique, we can formulate a variety of DBTG queries in a convenient and concise way, there are many queries in which a field value must be matched with a specified range of values, rather than to only one. To accomplish this match, we need to **get** the appropriate records into memory, to examine each one separately for a match, and thus to determine whether each is the target of our **find** statement.

As an illustration, consider the DBTG query to print the total number of accounts in the Perryridge branch with a balance greater than \$10,000:

```
count := 0;
branch.branch name := "Perryridge";
find any branch using branch name;
find first account within account branch;
while DB-status = 0 do
begin
  get account;
  if account.balance > 10000 then count := count + 1;
  find next account within account branch;
end
print (count);
```

## D.5 DBTG Update Facility

In Section D.4, we described the various DBTG commands for querying the database. In this section, we describe the mechanisms available for updating information in the database. They include the creation of new records and deletion of old records, as well as the modification of the content of existing records.

### D.5.1 Creation of New Records

To create a new record of type *<record type>*, we insert the appropriate values in the corresponding *<record type>* template. We then add this new record to the database by executing

**store** *<record type>*

Note that this technique allows us to create and add new records only one at a time.

As an illustration, consider the DBTG program for adding a new customer, Jackson, to the database:

```
customer.customer name := "Jackson"; customer.customer street := "Old Road";  
customer.customer city := "Richardson"; store customer;
```

Note that, if a new record is created that must belong to a particular DBTG set occurrence (for example, a new *account*), then, in addition to the **store** operation, we need a mechanism for inserting records into set occurrences. This mechanism is described in Section D.6.

### D.5.2 Modification of an Existing Record

To modify an existing record of type *<record type>*, we must find that record in the database, get that record into memory, and then change the desired fields in the template of *<record type>*. Then, we reflect the changes to the record to which the currency pointer of *<record type>* points by executing

**modify** *<record type>*

The DBTG model requires that the **find** command executed prior to modification of a record must have the additional clause **for update**, so that the system is aware that a record is to be modified. We are not required to update a record that we “find for update.” However, we cannot update a record unless it is found for update.

As an example, consider the DBTG program to change the street address of



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit III

Batch: 2017-2020

Turner to North Loop.

KAHE



```
customer.customer name := "Turner";  
find for update any customer using customer name;  
get customer;  
customer.customer street := "North Loop";  
modify customer;
```

### D.5.3 Deletion of a Record

To delete an existing record of type <record type>, we must make the currency pointer of that type point to the record in the database to be deleted. Then, we can delete that record by executing

```
erase <record type>
```

Note that, as in the case of record modification, the **find** command must have the attribute **for update** attached to it.

As an illustration, consider the DBTG program to delete account A-402 belonging to Turner:

```
finish := false;  
customer.customer name := "Turner";  
find any customer using customer name;  
find for update first account within depositor;  
while DB-status = 0 and not finish do  
begin  
  get account;  
  if account.account number = "A-402" then  
    begin  
      erase account;  
      finish := true;  
    end  
  else find for update next account within depositor;  
end
```

We can delete an entire set occurrence by finding the owner of the set — say, a record of type <record type> — and executing



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit III

Batch: 2017-2020

**erase all** <record type>

This command will delete the owner of the set, as well as all the set's members. If a member of the set is an owner of another set, the members of that second set also will be deleted. Thus, the **erase all** operation is recursive.

Consider the DBTG program to delete customer "Johnson" and all her ac- counts:



```
customer.customer name := "Johnson";  
find for update any customer using customer name;  
erase all customer;
```

A natural question is what happens when we wish to delete a record that is an owner of a set, but we do not specify **all** in the erase statement. In this case, several possibilities exist:

- Delete only that record.
- Delete the record and all its members.
- Do not delete any records.

It turns out that each of these options can be specified in the DBTG model. We discuss them in Section D.6.

## D.6 DBTG Set-Processing Facility

We saw in Section D.5 that the **store** and **erase** statements are closely tied to the set-processing facility. In particular, a mechanism must be provided for inserting records into and removing records from a particular set occurrence. In the case of deletion, we have a number of different options to consider if the record to be deleted is the owner of a set.

### D.6.1 The connect Statement

To insert a new record of type <record type> into a particular occurrence of <set-type>, we must first insert the record into the database (if it is not already there). Then, we need to set the currency pointers of <record type> and <set-type> to point to the appropriate record and set occurrence. Then, we can insert the new record into the set by executing

```
connect <record type> to <set-type>
```

A new record can be inserted as follows:

1. Create a new record of type <record type> (see Section D.5.1). This action sets the appropriate <record type> currency pointer.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit III

Batch: 2017-2020

2. Find the appropriate owner of the set <set-type>. This automatically sets the appropriate currency pointer of <set-type>.
3. Insert the new record into the set occurrence by executing the **connect** statement.

KAHE

As an illustration, consider the DBTG query for creating new account A-267, which belongs to Jackson:

```
account.account number := "A-267";  
account.balance := 0;  
store account;  
customer.customer name := "Jackson";  
find any customer using customer name;  
connect account to depositor;
```

### D.6.2 The disconnect Statement

To remove a record of type <record type> from a set occurrence of type <set-type>, we need to set the currency pointer of <record type> and <set-type> to point to the appropriate record and set occurrence. Then, we can remove the record from the set by executing

**disconnect** <record type> **from** <set-type>

Note that this operation only removes a record from a set; it does not delete that record from the database. If deletion is desired, we can delete the record by executing **erase** <record type>.

Assume that we wish to close account A-201. To do so, we need to delete the relationship between account A-201 and its customer. However, we need to keep the record of account A-201 in the database for the bank's internal archives. The following program shows how to perform these two actions within the DBTG model. This program will remove account A-201 from the set occurrence of type *depositor*. The account will still be accessible in the database for record-keeping purposes.

```
account.account number := "A-201";  
find for update any account using account number;  
find owner within depositor;  
disconnect account from depositor;
```

### D.6.3 The reconnect Statement

To move a record of type <record type> from one set occurrence to another set occurrence of type <set-type>, we need to find the appropriate record and the owner of the set occurrences to which that record is to be moved. Then, we can move the record by executing

**reconnect** <record type> **to** <set-type>



## **KARPAGAM ACADEMY OF HIGHER EDUCATION**

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit III

Batch: 2017-2020

Consider the DBTG program to move all accounts of Hayes that are currently at the Perryridge branch to the Downtown branch:

KAHE

```
customer.customer name := "Hayes";  
find any customer using customer name;  
find first account within depositor;  
while DB-status = 0 do  
begin  
  find owner within account branch;  
  get branch;  
  if branch.branch name = "Perryridge" then  
    begin  
      branch.branch name := "Downtown";  
      find any branch using branch name;  
      reconnect account to account branch;  
    end  
  find next account within depositor;  
end
```

#### D.6.4 Insertion and Retention of Records

When a new set is defined, we must specify how member records are to be inserted. In addition, we must specify the conditions under which a record must be retained in the set occurrence in which it was initially inserted.

##### D.6.4.1 Set Insertion

A newly created member record of type <record type> of a set type <set-type> can be added to a set occurrence either explicitly (manually) or implicitly (auto- matically). This distinction is specified at set-definition time via

**insertion is** <insert mode>

where <insert mode> can take one of two forms:

- **Manual.** We can insert the new record into the set manually (explicitly) by executing

**connect** <record type> **to** <set-type>

- **Automatic.** The new record is inserted into the set automatically (implicitly) when it is created — that is, when we execute

**store** <record type>

In either case, just prior to insertion, the <set-type> currency pointer must point to the set occurrence into which the insertion is to be made.

As an illustration, consider the creation of account A-535 that belongs to Hayes and is at the Downtown branch. Suppose that set insertion is **manual** for set type *depositor* and is **automatic** for set type *account branch*. The appropriate DBTG program is

```
branch.branch name := "Downtown"; find any branch using branch name; account.account  
number := "A-535"; account.balance := 0;  
store account;  
customer.customer name := "Hayes";  
find any customer using customer name;  
connect account to depositor;
```

#### D.6.4.2 Set Retention

There are various restrictions on how and when a member record can be removed from a set occurrence into which it has been inserted previously. These restrictions are specified at set-definition time via

**retention is** <retention-mode>

where <retention-mode> can take one of the three forms:

1. **Fixed.** Once a member record has been inserted into a particular set occurrence, it cannot be removed from that set. If retention is fixed, then, to reconnect a record to another set, we must erase that record, re-create it, and then insert it into the new set occurrence.
2. **Mandatory.** Once a member record has been inserted into a particular set occurrence, it can be reconnected to another set occurrence of only type <set-type>. It can neither be disconnected nor be reconnected to a set of another type.
3. **Optional.** No restrictions are placed on how and when a member record can be removed from a set occurrence. A member record can be reconnected, disconnected, and connected at will.

The decision of which option to choose depends on the application. For example, in our banking database, the **optional** retention mode is appropriate for the *depositor* set because we may have defunct accounts not owned by anybody. On the other hand, the **mandatory** retention mode is appropriate for the *account branch* set, since an account *has* to belong to some branch.

#### D.6.5 Deletion



When a record is deleted (erased) and that record is the owner of set occurrence of type `<set-type>`, the best way of handling this deletion depends on the specification of the set retention of `<set-type>`.

- If the retention status is **optional**, then the record will be deleted and every member of the set that it owns will be disconnected. These records, however, will remain in the database.
- If the retention status is **fixed**, then the record and all its owned members will be deleted. This action occurs because the fixed status means that a member record cannot be removed from the set occurrence without being deleted.
- If the retention status is **mandatory**, then the record cannot be erased, because the mandatory status indicates that a member record must belong to a set occurrence. The record cannot be disconnected from that set.

#### D.6.6 Set Ordering

The members of a set occurrence of type `<set-type>` can be ordered in a variety of ways. These orders are specified by a programmer when the set is defined via

**order** is `<order-mode>`

where `<order-mode>` can be any of the following:

- **first**. When a new record is added to a set, it is inserted in the first position. Thus, the set is in reverse chronological order.
- **last**. When a new record is added to a set, it is inserted in the final position. Thus, the set is in chronological order.
- **next**. Suppose that the currency pointer of `<set-type>` points to record *X*. If *X* is a member type, then, when a new record is added to the set, that record is inserted in the next position following *X*. If *X* is an owner type, then, when a new record is added, that record is inserted in the first position.
- **prior**. Suppose that the currency pointer of `<set-type>` points to record *X*. If *X* is a member type, then, when a new record is added to the set, that record is inserted in the position just prior to *X*. If *X* is an owner type, then, when a new record is added, that record is inserted in the last position.
- **system default**. When a new record is added to a set, it is inserted in an arbitrary position determined by the system.
- **sorted**. When a new record is added to a set, it is inserted in a position that ensures that the set will remain sorted. The sorting order is specified by a particular key value when a programmer



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit III

Batch: 2017-2020

defines the set. The programmer must specify whether members are ordered in ascending or descending order relative to that key.

Consider again Figure D.16, where the set occurrence of type *depositor* with the owner-record customer Turner and member-record accounts A-305, A-402, and A-408 are ordered as indicated. Suppose that we add a new account A-125 to that set. For each <order-mode> option, the new set ordering is as follows:

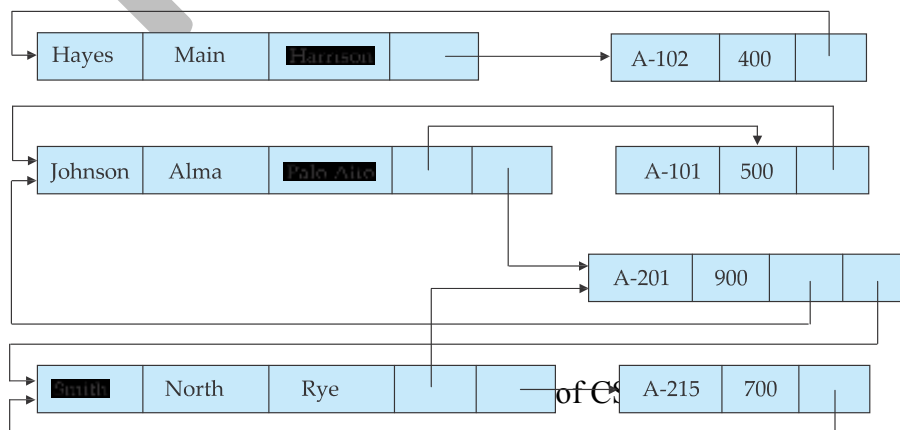


- **first:** {A-125, A-305, A-402, A-408}
- **last:** {A-305, A-402, A-408, A-125}
- **next:** Suppose that the currency pointer points to record “Turner”; then the new set order is {A-125, A-305, A-402, A-408}
- **prior:** Suppose that the currency pointer points to record A-402; then the new set order is {A-305, A-125, A-402, A-408}
- **system default:** Any arbitrary order is acceptable; thus, {A-305, A-402, A-125, A-408} is a valid set ordering
- **sorted:** The set must be ordered in ascending order with account number being the key; thus, the ordering must be {A-125, A-305, A-402, A-408}

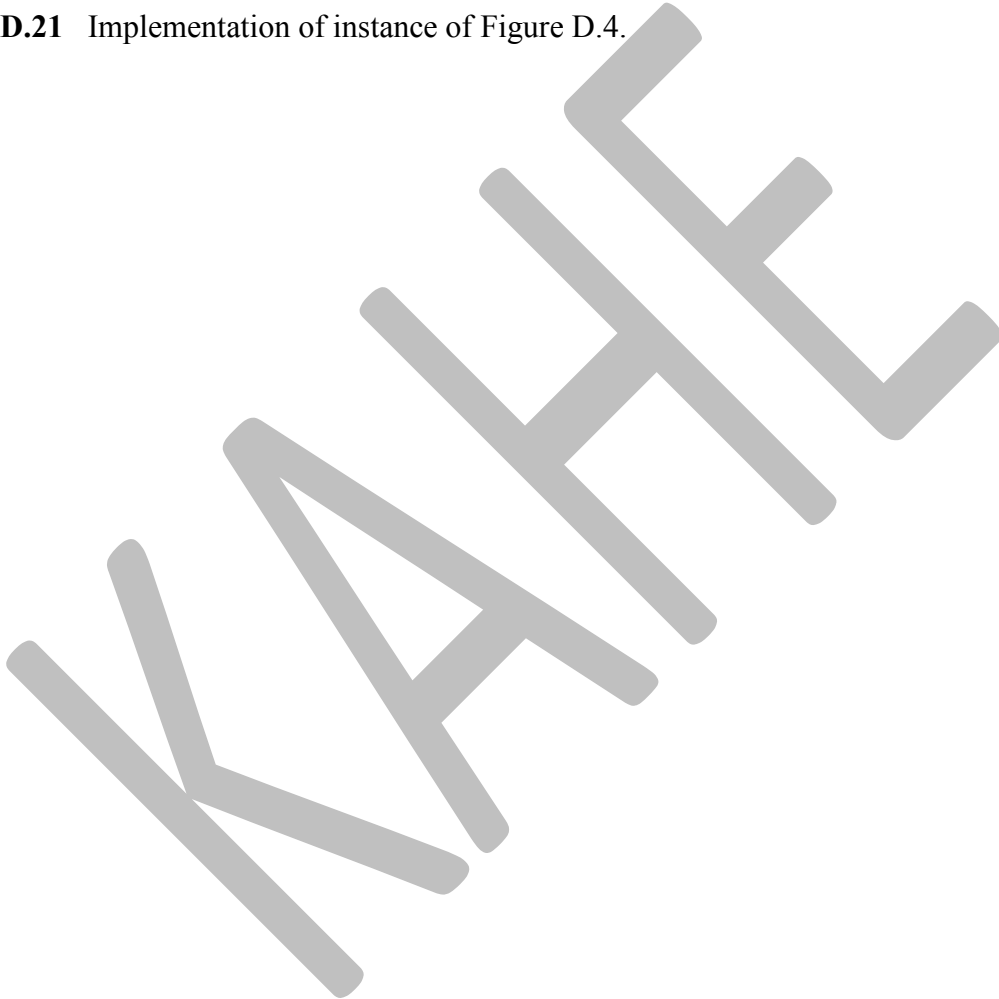
## D.7 Mapping of Networks to Files

A network database consists of records and links. We implement links by adding *pointer fields* to records that are associated via a link. Each record must have one pointer field for each link with which it is associated. As an illustration, return to the data-structure diagram of Figure D.2b, and to the sample database corresponding to it in Figure D.4. Figure D.21 shows the sample instance with pointer fields to represent the links. Each line in Figure D.4 is replaced in Figure D.21 by two pointers.

Since the *depositor* link is many to many, each record can be associated with an arbitrary number of records. Thus, it is not possible to limit the number of pointer fields in a record. Therefore, even if a record itself is of fixed length, the actual record used in the physical implementation is a variable-length record.



**Figure D.21** Implementation of instance of Figure D.4.



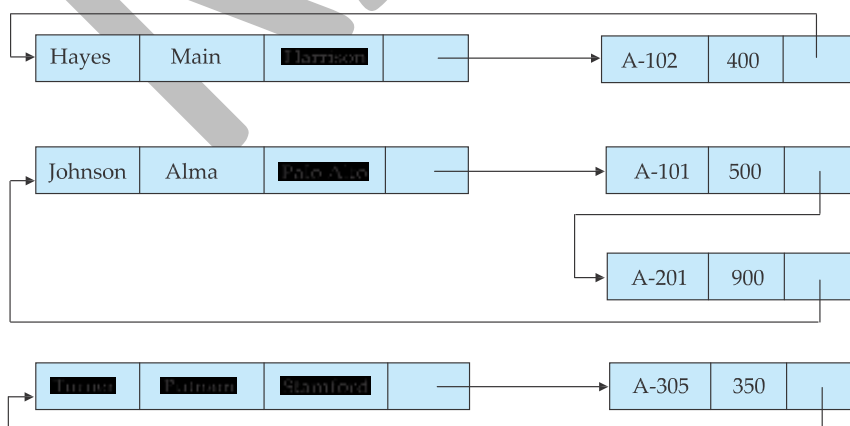
These complications led the architects of the DBTG model to restrict links to be either one to one or one to many. We shall see that, under this restriction, the number of pointers needed is reduced, and it is possible to retain fixed-length records. To illustrate the implementation of the DBTG model, we assume that the  *depositor* link is one to many and is represented by the DBTG set  *depositor* as defined here:

**set name is depositor owner is customer member is account**

A sample database corresponding to this schema is in Figure D.1.

An  *account* record can be associated with only one  *customer* record. Thus, we need only one pointer in the  *account* record to represent the  *depositor* relationship. However, a  *customer* record can be associated with many  *account* records. Instead of using multiple pointers in the  *customer* record, we can use a  *ring structure* to represent the entire occurrence of the DBTG set  *depositor*. In a ring structure, the records of both the owner and member types for a set occurrence are organized into a circular list. There is one circular list for each set occurrence (that is, for each record of the owner type).

Figure D.22 shows the ring structure for the example of Figure D.1. Let us examine the DBTG-set occurrence owned by the “Johnson” record. There are two member-type ( *account*) records. Instead of containing one pointer to each member record, the owner (Johnson) record contains a pointer to only the first member record (account A-101). This member record contains a pointer to the next member record (account A-201). Since the record for account A-201 is the final member record, it contains a pointer to the owner record.



**Figure D.22** Ring structure for instance of Figure D.1.



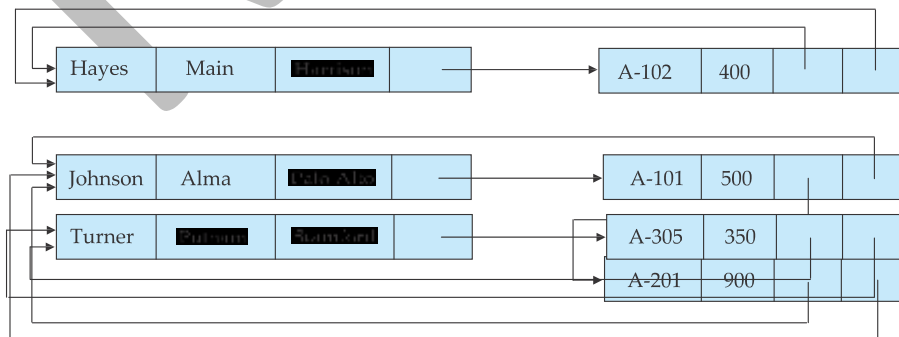
If we represent DBTG sets by using the ring structure, a record contains exactly one pointer for each DBTG set in which it is involved, regardless of whether it is of the owner type or member type. Thus, we can represent fixed-length records within a ring structure without resorting to variable-length records. This structural simplicity is offset by added complexity in accessing records within a set. To find a particular member record of a set occurrence, we must traverse the pointer chain to navigate from the owner record to the desired member record.

The ring-structure implementation strategy for the DBTG model provided the basis for the DBTG data retrieval facility. Recall these statements:

- **find first** <record type> **within** <set type>
- **find next** <record type> **within** <set type>

The terms **first** and **next** in these statements refer to the ordering of records given by the ring-structure pointers. Thus, once the owner has been found, it is easy to do a **find first**, since all the system must do is to follow a pointer. Similarly, all the system must do in response to a **find next** is to follow the ring-structure pointer.

The **find owner** statement of the DBTG query language can be supported efficiently by a modified form of the ring structure in which every member-type record contains a second pointer, which points to the owner record. This structure appears in Figure D.23. Under this implementation strategy, a record has one pointer for each DBTG set for which it is of the owner type, and two pointers (a *next-member* pointer and an *owner* pointer) for each DBTG set for which it is of the member type. This strategy allows efficient execution of a **find owner** statement. Under our earlier strategy, it is necessary to traverse the ring structure until we find the owner.



**Figure D.23** Ring structure of Figure D.22 with owner pointers.





Block 1

Block 1

Hayes	Main	Turner	
Johnson	Alma	Turner	
Turner	Turner	Turner	

A-102	400		
-------	-----	--	--

A-101	500		
A-201	900		

A-305	350		
-------	-----	--	--

Block 2

**Figure D.24** Clustered record placement for instance of Figure D.1.

The physical placement of records is important for an efficient implementation of a network database, as it is for a relational database.

The statements **find first**, **find next**, and **find owner** are designed for processing a sequence of records within a particular DBTG-set occurrence. Since these statements are the ones most frequently used in a DBTG query, it is desirable to store records of a DBTG-set occurrence physically close to one another on disk. To specify the strategy that the system is to use to store a DBTG set, we add a **placement** clause to the definition of the member record type.

Consider the DBTG set *depositor* and the example shown in Figure D.1. If we add the clause

**placement clustered via depositor**

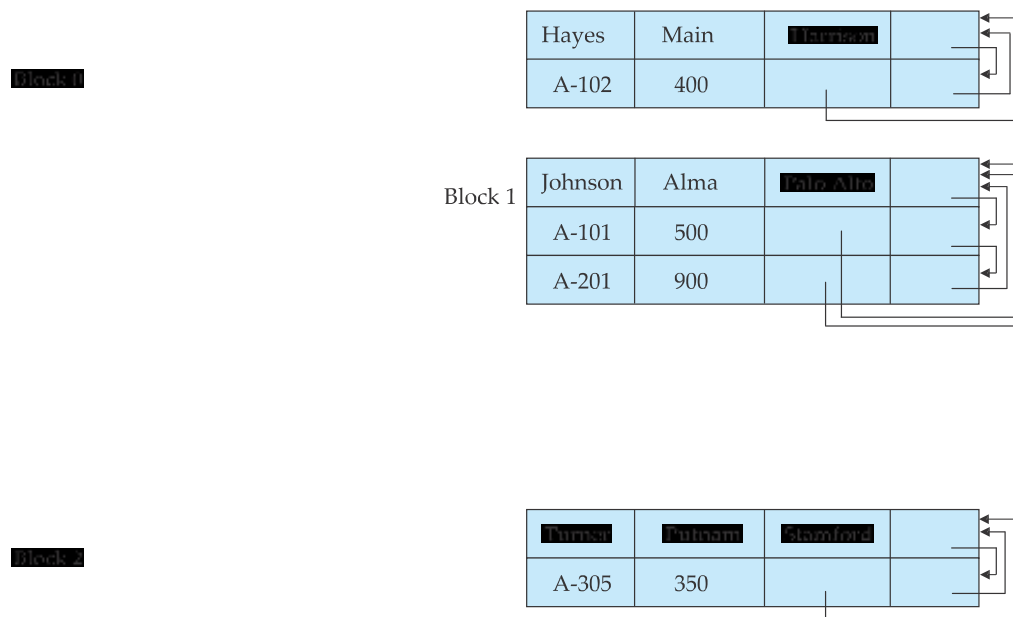
to the definition of record type *account* (the member-record type of the *depositor* DBTG set), the system will store members of each set occurrence close to one another physically on disk. To the

extent possible, members of a set occurrence will be stored in the same block. Figure D.24 illustrates this storage strategy for the instance of Figure D.1.

The clustered placement strategy does not require the owner record of a DBTG set to be stored near the set's members. Thus, each record type can be stored in a distinct file. If we are willing to store more than one record type in a file, we can specify that owner and member records are to be stored close to one another physically on disk. We do so by adding the clause **near owner** to the **placement** clause. For our example of the *depositor* set, we add the clause

**placement clustered via *depositor* near owner**

to the definition of the record type *account*. Figure D.25 illustrates this storage strategy. By storing member records in the same block as the owner, we reduce



**Figure D.25** Record placement using clustering with the **near owner** option.

the number of block accesses required to read an entire set occurrence. This form of storage is analogous to the clustering file structure that we proposed earlier for the relational model. This similarity is not surprising, since queries that require traversal of DBTG-set occurrences under the network model require natural joins under the relational model.

## D.8 Summary

A network database consists of a collection of *records* that are connected to each other through *links*. A link is an association between precisely two records. Records are organized in the form of an arbitrary graph.

A *data-structure diagram* is a schema for a network database. Such a diagram consists of two basic components: boxes, which correspond to record types, and lines, which correspond to links. A data-structure diagram serves the same purpose as an E-R diagram; namely, it specifies the overall logical structure of the database. For every E-R diagram, there is a corresponding data-structure diagram.

In the late 1960s, several commercial database systems based on the network model emerged. These systems were studied extensively by the Database Task Group (DBTG) within the CODASYL group. In the DBTG model, only many-to-one links can be used. Many-to-many links are disallowed to simplify the implementation. One-to-one links are represented as many-to-one links. A data-structure diagram consisting of two record types that are linked together is referred to, in the DBTG model, as a *DBTG set*. Each DBTG set has one record type designated as the *owner* of the set, and another record type designated as a *member* of the set. A

DBTG set can have any number of *set occurrences*.

The data-manipulation language of the DBTG model consists of a number of commands embedded in a host language. These commands access and manipulate database records and links, as well as locally declared variables. For each such application program, the system maintains a *program work area*, which contains *record templates*, *currency pointers*, and *status flags*.

The two most frequently used DBTG commands are **find** and **get**. There are many different formats for the **find** command. The main distinction among them is whether any records in the database, or records within a particular set occurrence, are to be located.

There are various mechanisms available in the DBTG model for updating information in the database. They allow the creation and deletion of new records (via the **store** and **erase** operations), as well as the modification (via the **modify** operation) of the content of existing records. The **connect**, **disconnect**, and **reconnect** operations provide for inserting records into and removing records from a particular set occurrence.

When a new set is defined, we must specify how member records are to be inserted, and under what conditions they can be moved from one set occurrence

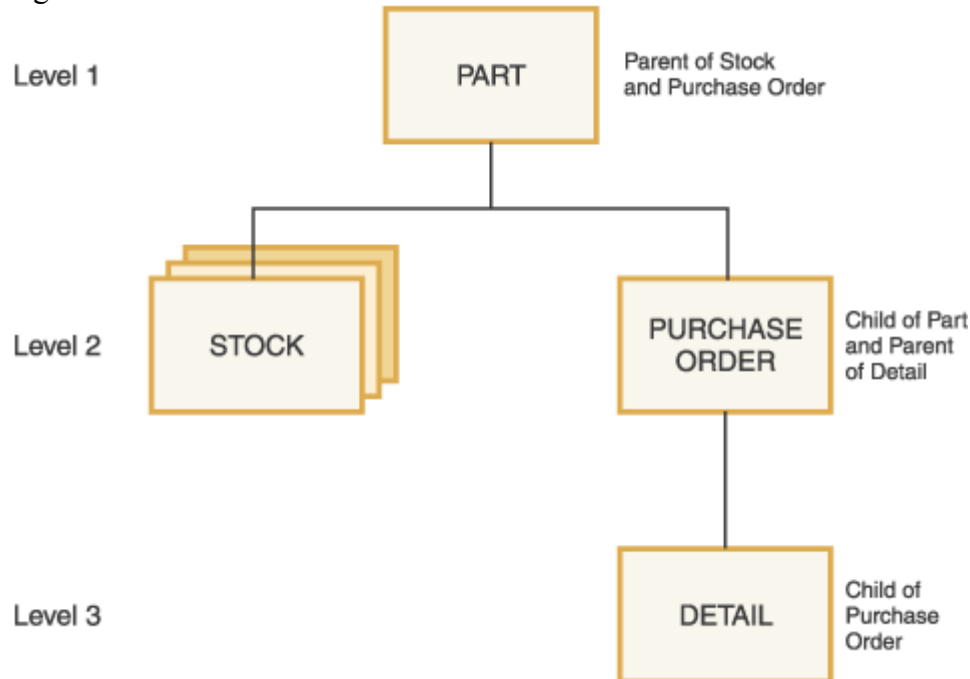
to another. A newly created member record can be added to a set occurrence either explicitly or implicitly. This distinction is specified at set-definition time via the **insertion is** statement with the **manual** and **automatic** insert-mode options. There are various restrictions on how and when a member record can be removed from a set occurrence into which it has been inserted previously. These restrictions are specified at set-definition time via the **retention is** statement with the **fixed**, **mandatory**, and **optional** retention-mode options. Implementation techniques for the DBTG model exploit the restrictions of the model to allow the physical representation of DBTG sets without the need for variable-length records. A DBTG set is represented by one ring structure for each occurrence.

## IBM

IBM is a database management system (DBMS) that helps you organize business data with both program and device independence.

Hierarchical databases and data manipulation language (DL/I calls) are the heart of IMS DB. Data within the database is arranged in a tree structure, with data at each level of the hierarchy related to, and in some way dependent on, data at a higher level of the hierarchy. The following figure shows the hierarchical database model. In a hierarchical database, data is stored in the database only once, but it might be able to be retrieved by several paths.

Figure 1. Hierarchical database model



With IMS DB, you can do the following:

- Maintain data integrity. The data in each database is consistent and remains in the database even when IMS DB is not running.
- Define the database structure and the relationships between the database segments. IMS segments in one database can have relationships with segments in a different IMS database (logical relationships). IMS databases also can be accessed by one or more secondary indexes.
- Provide a central point of control and access for the IMS data that is processed by IMS applications.
- Perform queries against the data in the database.
- Perform database transactions (inserts, updates, and deletes) as a single unit of work so that the entire transaction either completes or does not complete.
- Perform multiple database transactions concurrently, with the results of each transaction kept isolated from the others.
- Maintain the databases. IMS DB provides facilities for backing up and recovering IMS databases, as well as facilities for tuning the databases by reorganizing and restructuring them.

In addition, IMS DB enables you to adapt IMS databases to the requirements of varied applications. Application programs can access common (and therefore consistent) data, thereby reducing the need to maintain the same data in multiple ways in separate files for different applications.

IMS databases are accessed internally using a number of IMS database organization access methods. The database data is stored on disk storage using z/OS® access methods.

IMS DB provides access to these databases from applications running under the following:

- IMS Transaction Manager (IMS TM)

- CICS® Transaction Server for z/OS
- z/OS batch jobs
- WebSphere® Application Server for z/OS
- DB2® for z/OS stored procedures

In addition, IMS Version 11 and later includes the IMS Open Database Manager (ODBM). ODBM enables access to IMS databases from Java programs anywhere in the enterprise using distributed relational database architecture (DRDA®) and distributed database management (DDM).

**POSSIBLE QUESTIONS****UNIT 1****PART – A (20 MARKS)****(Q.NO 1 TO 20 Online Examinations)****Part - B (5X 6 =30 Marks)  
(Answer ALL the Questions)**

1. Illustrate the Binary relation concept of network model in detail.
2. Explain the general relations – hips concepts of network model in detail.
3. Explain the DDL commands used in network model in detail.
4. What are the DML commands used in network model?
5. How are the DML commands of n/w model interpreted in the application environment.
6. Write short notes on IBMS IMS data base system.
7. Write a note on the several record access scheme available in IMS.
8. Explain the basic concepts of Hierarchical model with a suitable example.\
9. Write short notes on tree structure diagrams in hierarchical data base.
10. Describe the DML command statements in Hierarchical Data Model.

**Part - C (10X1 =10 Marks)  
(Compulsory Question)**

1. Define the term locking. Explain the different types of locking used in concurrency control.
2. Explain in detail the 13 object oriented database management system Commandments.
3. Compare and contrast the traditional and object oriented DBMS.
4. Elaborate different Data Manipulation Language in SQL. Explain the different statements used in DML.
5. Discuss on functional dependencies of relational database design in detail with a suitable example.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Established Under Section 3 of UGC Act 1956)  
COMBATOORE – 641 021  
**DEPARTMENT OF COMPUTER APPLICATIONS**  
Batch 2017 – 2020

Subject : Database Management Systems

Subject Code: 17CAU402

DEPARTMENT OF COMPUTER APPLICATIONS					
S.N	Questions	out1	out2	out3	out4
1	The execution cost is expressed as a weighted combination of _____ and _____ cost.	SMPs	HTTP	CPU	CMPT
2	How many components involved in query optimization ?	1	2	3	4
3	_____ is the set of alternative execution plans to represent the input query.	cost model	search strategy	query	search space
4	_____ predict the cost of a given execution plan.	search strategy	cost model	query	search space
5	_____ strategy explores the search space and selects the best plan using the cost model.	cost model	query	search space	search strategy
6	_____ is a tree such that at least one operand of each operator node is a base relation.	linear tree	busy tree	join tree	tree view
7	_____ is more general and may have operators with no base relations as operands.	busy tree	linear tree	join tree	tree view
8	Deterministic strategies proceed by _____ plan.	dynamic	deterministic	building	search strategy
9	_____ strategies allow the optimizer to trade optimization time for execution time.	normalized	transformation	optimization	randomized
10	query optimization techniques are often extensions of the techniques for _____ system.	INGRES	Quel	centralized	decentralized
11	The transaction can complete its task successfully is called ?	commits	aborts	update	null
12	The transaction stays without completing its task is called ?	commit	aborts	update	null
13	Actions are updates by restoring the database to the state before their execution is called?	commit	aborts	rollback	deadlock
14	The data items that a transaction reads are said to constitute is called ?	write set	read set	write and read set	base set
15	The data item that a transaction writes are said to constitute is called ?	write set	read set	write and read set	base set
16	The union of the read set and write set of transaction is known as _____.	write set	read set	write and read set	base set
17	dynamic database have to deal with the problem of _____.	phantoms	static	dynamic	tuples
18	Hierarchical architecture is a combination of _____.	shared nothing	shared disk	shared memory	shared nothing and memory
19	Hierarchical architecture is also called _____.	cluster	shared partition	disk	memory
20	The term declustering is also used to mean _____.	partitioning	hash	range	partitioning
21	_____ is the simplest strategy. it ensures uniform data distribution.	round robin partitioning	hash partitioning	range partitioning	hash indexing
22	_____ applies a hash function to some attributes which yields the partition number.	round robin partitioning	hash partitioning	range partitioning	hash indexing
23	_____ has distributed tuples based on the value intervals of some attributes.	round robin partitioning	hash partitioning	range partitioning	hash indexing
24	_____ enables the parallel execution of multiple queries generated by concurrent transactions.	inter query	inter operator	intra operation	parallelism
25	_____ are used to decrease response time.	inter query	inter operator	intra operation	parallelism
26	Intra_operator parallelism is based on the decomposition of one operator in a set of independent sub operators called ?	operator instance	inter operator	intra operation	parallelism
27	_____ has several operators with a producer_consumer link are executed in parallel.	materialized	pipeline parallelism	independent parallelism	processors
28	_____ is the set of alternative execution plan to represent the input query.	cost model	search strategy	annotation	search space
29	Operator trees are enriched with _____.	cost model	search strategy	annotation	search space
30	Load balancing problem can appear with intra_operator parallelism namely _____.	data skew	selectivity skew	redistribution skew	data skew
31	_____ explores the search space and selects the best plan.	cost model	search strategy	annotation	search space
32	_____ is the smallest unit of sequential processing that cannot be further partitioned.	activation	redistribution skew	selectivity skew	product skew
33	object represents a _____ in the system that is being modeled.	real entity	object identity	identical value	real entity
34	An _____ database is a template for all objects of that type.	primitive	secondary	primitive	abstract
35	The composite object relationship between types can be represented by _____.	class	collection	composition	subtyping
36	collection is similar to a _____, in that it groups object.	collection	composition	subtyping	class
37	subtyping is based on the _____ relationship among types.	specialization	collection	composition	subtyping
38	The data allocation problem for object database involves allocation of both _____.	methods	classes	methods and classes	functions
39	How many types of client /server architectures have been proposed ?	2	1	4	5
40	OID represents the _____.	POID	AID	object identifier management	LOID
41	The process of converting a disk version of the pointer to a methods version of a pointer is known as _____.	pointer Swizzling	Load mapping	Load generation	surrogates
42	A common way of tracking objects is to leave _____.	pointer Swizzling	Load mapping	Load generation	surrogates
43	_____ object are currently involved in an activity in response to an invocation or a message.	ready	active	waiting	suspended
44	_____ object are not currently involved.	ready	active	waiting	suspended
45	_____ object are temporarily unavailable for invocation.	ready	active	waiting	suspended
46	Relational query language operate on very simple type systems consisting of a single type _____.	relation	physical storage	path expression	query processing
47	_____ is an alternative general technique to represent and compute path expressions.	access support relation	set matching	path indexes	access support relation
48	_____ relation that has been defined to determine serializable histories.	access support relation	set matching	path indexes	access support relation
49	The architecture of a hierarchical database is based on _____ the concepts of relationships.	set structure	tree node	parent/child set	parent/child set

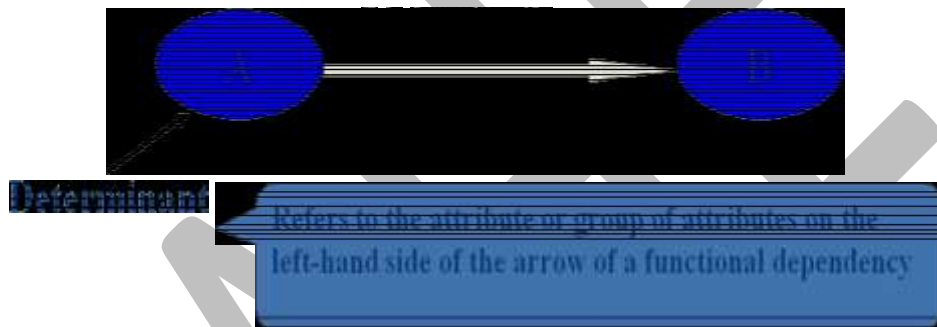


#### **UNIT IV**

Relational Data Base Design - Function Dependencies & Normalization for Relational Databases - Functional Dependencies - Normal forms based on primary keys - (1NF, 2NF, 3NF & BCNF) - Lossless join & Dependency preserving decomposition

#### **FUNCTIONAL DEPENDENCIES:**

**Functional dependency** describes the relationship between attributes in a relation. For example, if A and B are attributes of relation R, and B is functionally dependent on A (denoted  $A \twoheadrightarrow B$ ), if each value of A is associated with exactly one value of B. (A and B may each consist of one or more attributes.)



**Trivial functional dependency** means that the right-hand side is a subset (not necessarily a proper subset) of the left-hand side. They do not provide any additional information about possible integrity constraints on the values held by these attributes.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit IV      Batch: 2017-2020

We are normally more interested in **nontrivial dependencies** because they represent integrity constraints for the relation.

### Main characteristics of functional dependencies in normalization

- Have a one-to-one relationship between attribute(s) on the left- and right- hand side of a dependency;
- hold for all time;
- are nontrivial.

**Functional dependency** is a property of the meaning or semantics of the attributes in a relation. When a functional dependency is present, the dependency is specified as a **constraint** between the attributes.

An important integrity constraint to consider first is **the identification of candidate keys, one of which is selected to be the primary key** for the relation using functional dependency.

### FIRST NORMAL FORM:

A basic objective of the first normal form defined by Codd in 1970 was to permit data to be queried and manipulated using a "universal data sub-language" grounded in first-order logic. SQL is an example of such a data sub-language, albeit one that Codd regarded as seriously flawed.). Querying and manipulating the data within an unnormalized data structure, such as the following non-1NF representation of customers' credit card transactions, involves more complexity than is really necessary:

One of Codd's important insights was that this structural complexity could always be removed completely, leading to much greater power and flexibility in the way queries could be formulated (by users and applications) and evaluated (by the DBMS).

Now each row represents an individual credit card transaction, and the DBMS can obtain the answer of interest, simply by finding all rows with a Date falling in October, and summing their Amounts. All of the values in the data structure are on an equal footing: they are all exposed to the DBMS directly, and can directly participate in queries, whereas in the previous situation some values were embedded in lower-level structures that had to be handled specially. Accordingly, the normalized design lends itself to general-purpose query processing, whereas the unnormalized design does not.

The objectives of normalization beyond 1NF were stated as follows by Codd:



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit IV      Batch: 2017-2020

1. To free the collection of relations from undesirable insertion, update and deletion dependencies;
2. To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the life span of application programs;
3. To make the relational model more informative to users;
4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.

—E.F. Codd, "Further Normalization of the Data Base Relational Model.

### SECOND NORMAL FORM

**Second normal form (2NF)** is a normal form used in database normalization. 2NF was originally defined by E.F. Codd in 1971. A table that is in first normal form (1NF) must meet additional criteria if it is to qualify for second normal form. Specifically: a 1NF table is in 2NF if and only if, given any candidate key K and any attribute A that is not a constituent of a candidate key, A depends upon the whole of K rather than just a part of it.

In slightly more formal terms: a 1NF table is in 2NF if and only if all its non-prime attributes are functionally dependent on the whole of a candidate key. (A non-prime attribute is one that does not belong to any candidate key.)

Note that when a 1NF table has no composite candidate keys (candidate keys consisting of more than one attribute), the table is automatically in 2NF.

A 2NF alternative to this design would represent the same information in two tables: an "Employees" table with candidate key {Employee}, and an "Employees' Skills" table with candidate key {Employee, Skill}:

Neither of these tables can suffer from update anomalies.

Not all 2NF tables are free from update anomalies, however. An example of a 2NF table which suffers from update anomalies is:



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit IV      Batch: 2017-2020

The underlying problem is the transitive dependency to which the Winner Date of Birth attribute is subject. Winner Date of Birth actually depends on Winner, which in turn depends on the key Tournament / Year.

This problem is addressed by third normal form (3NF).

### THIRD NORMAL FORM

The **third normal form (3NF)** is a normal form used in database normalization. 3NF was originally defined by E.F. Codd in 1971. Codd's definition states that a table is in 3NF if and only if both of the following conditions hold:

- The relation R (table) is in second normal form (2NF)
- Every non-prime attribute of R is non-transitively dependent (i.e. directly dependent) on every key of R.
- 

A **non-prime attribute** of R is an attribute that does not belong to any candidate key of R. A transitive dependency is a functional dependency in which  $X \rightarrow Z$  ( $X$  determines  $Z$ ) indirectly, by virtue of  $X \rightarrow Y$  and  $Y \rightarrow Z$  (where it is not the case that  $Y \rightarrow X$ ).

A 3NF definition that is equivalent to Codd's, but expressed differently was given by Carlo Zaniolo in 1982. This definition states that a table is in 3NF if and only if, for each of its functional dependencies  $X \rightarrow A$ , **at least one** of the following conditions holds:

- $X$  contains  $A$  (that is,  $X \rightarrow A$  is trivial functional dependency), or
- $X$  is a superkey, or
- $A$  is a **prime attribute** (i.e.,  $A$  is contained within a candidate key)
- an example of a 2NF table that fails to meet the requirements of 3NF is:

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA      Course Name: DATA BASE MANAGEMENT SYSTEMS  
Course Code : 17CAU402      Unit IV      Batch: 2017-2020

- Because each row in the table needs to tell us who won a particular Tournament in a particular Year, the composite key {Tournament, Year} is a minimal set of attributes guaranteed to uniquely identify a row. That is, {Tournament, Year} is a candidate key for the table.
- The breach of 3NF occurs because the non-prime attribute Winner Date of Birth is transitively dependent on the candidate key {Tournament, Year} via the non-prime attribute Winner. The fact that Winner Date of Birth is functionally dependent on Winner makes the table vulnerable to logical inconsistencies, as there is nothing to stop the same person from being shown with different dates of birth on different records.
- In order to express the same facts without violating 3NF, it is necessary to split the table into two:
- Update anomalies cannot occur in these tables, which are both in 3NF.

### BCNF

**Boyce-Codd normal form** (or **BCNF** or **3.5NF**) is a normal form used in database normalization. It is a slightly stronger version of the third normal form (3NF). A table is in Boyce-Codd normal form if and only if for every one of its non-trivial [dependencies]  $X \rightarrow Y$ ,  $X$  is a superkey—that is,  $X$  is either a candidate key or a superset thereof.

Only in rare cases does a 3NF table not meet the requirements of BCNF. A 3NF table which does not have multiple overlapping candidate keys is guaranteed to be in BCNF.[4] Depending on what its functional dependencies are, a 3NF table with two or more overlapping candidate keys may or may not be in BCNF.

An example of a 3NF table that does not meet BCNF is: Today's Court Bookings

#### Court Start Time End Time Rate Type

1	09:30	10:30	SAVER
1	11:00	12:00	SAVER
1	14:00	15:30	STANDARD



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit IV

Batch: 2017-2020

2	10:00	11:30	PREMIUM-B
2	11:30	13:30	PREMIUM-B
2	15:00	16:30	PREMIUM-A

- Each row in the table represents a court booking at a tennis club that has one hard court (Court 1) and one grass court (Court 2)
- A booking is defined by its Court and the period for which the Court is reserved
- Additionally, each booking has a Rate Type associated with it. There are four distinct rate types:
  - SAVER, for Court 1 bookings made by members
  - STANDARD, for Court 1 bookings made by non-members
  - PREMIUM-A, for Court 2 bookings made by members
  - PREMIUM-B, for Court 2 bookings made by non-members

The table's candidate keys are:

- {Court, Start Time}
- {Court, End Time}
- {Rate Type, Start Time}
- {Rate Type, End Time}

Recall that 2NF prohibits partial functional dependencies of non-prime attributes on candidate keys, and that 3NF prohibits transitive functional dependencies of non-prime attributes on candidate keys. In the Today's Court Bookings table, there are no non-prime attributes: that is, all attributes belong to candidate keys. Therefore the table adheres to both 2NF and 3NF.

The table does not adhere to BCNF. This is because of the dependency Rate Type  $\rightarrow$  Court, in which the determining attribute (Rate Type) is neither a candidate key nor a superset of a candidate key.

Any table that falls short of BCNF will be vulnerable to logical inconsistencies. In this example, enforcing the candidate keys will not ensure that the dependency Rate Type  $\rightarrow$  Court is respected. There is, for instance, nothing to stop us from assigning a PREMIUM A Rate Type to a Court 1 booking as well as a Court 2 booking—a clear contradiction, as a Rate Type should only ever apply to a single Court.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA

Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit IV

Batch: 2017-2020

The design can be amended so that it meets BCNF: Today's Bookings

Rate Types

**Court Start End Member**

**Rate Type Court Member**

**Time**

**Time Flag**

**Flag**

SAVER 1 Yes STANDARD

1 No PREMIUM-

A 2 Yes

PREMIUM-

B 2 No

1 09:30 10:30 Yes

1 11:00 12:00 Yes

1 14:00 15:30 No

2 10:00 11:30 No

2 11:30 13:30 No

2 15:00 16:30 Yes





## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit IV Batch: 2017-2020

The candidate keys for the Rate Types table are {Rate Type} and {Court, Member Flag}; the candidate keys for the Today's Bookings table are {Court, Start Time} and {Court, End Time}. Both tables are in BCNF. Having one Rate Type associated with two different Courts is now impossible, so the anomaly affecting the original table has been eliminated.

### COMPARISON OF 3NF AND BCNF

Boyce Codd normal form (also known as BCNF) is a normal form –that is a form that provides criteria for determining a table's degree of vulnerability to logical inconsistencies and anomalies. This normal form is used in database normalisation. It is a bit stronger than its predecessor, the third normal form (also known as 3NF). A table is thought to be in BCNF if and only if for every one of its non-trivial functional dependencies –that is a boundary that is set between two sets of attributes in a relation taken from a database– is a superkey (a set of attributes of a relational variable that postulates that in all relations assigned to that specific variable there are no two distinct rows containing the same value for the attributes in that particular set). BCNF postulates that any table that fails to meet the criteria to be attributed as a BCNF is vulnerable to logical inconsistencies.

3NF is a normal form that is also used in database normalisation. It is thought that a table is in 3NF if and only if 1) the table is in second normal form (or 2NF, which is a first normal code, or 1NF, that has met the criteria to become a 2NF), and 2) every non-prime attribute of the table is non-transitively dependent on every key of the table (meaning it is not directly dependent on every key). There is another postulation of 3NF that is also used to define the differences between 3NF and the BCNF.

This theorem was conceived by Carlo Zaniolo in 1982. It states that a table is in 3NF if and only if for each functional dependency where  $X \rightarrow A$ , at least one of three conditions must hold: either  $X \rightarrow A$ ,  $X$  is a superkey, or  $A$  is a prime attribute (which means  $A$  is contained within a candidate key –or a minimal superkey for that relation). This newer definition differs from the theorem of a BCNF in that the latter model would simply eliminate the last condition. Even as it acts as a newer version of the 3NF theorem, there is a derivation of the Zaniolo theorem. It states that  $X \rightarrow A$  is non-trivial. If that is true, let  $A$  be a non-key attribute and also let  $Y$  be a key of  $R$ . If that holds then  $Y \rightarrow X$ . This means that  $A$  is not transitively dependent on  $Y$  if and only if  $X \rightarrow Y$  (or if  $X$  is a superkey).

### LOSSLESS AND DEPENDENCY PRESERVING DECOMPOSITION





## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit IV Batch: 2017-2020

### Lossless-Join Decomposition

Let  $R$  be a relation schema and let  $F$  be  $H$ , set of FDs over  $R$ . A decomposition of  $R$  into two schemas with attribute sets  $X$  and  $Y$  is said to be a lossless-join decomposition with respect to  $F$  if, for every instance  $T$  of  $R$  that satisfies the dependencies in

$F$ ,  $\pi_X(r) \bowtie \pi_Y(r) = r$ . In other words, we can recover the original relation from the decomposed relations.

This definition can easily be extended to cover a decomposition of  $R$  into more than two relations. It is easy to see that  $r \subseteq \pi_X(r) \bowtie \pi_Y(r)$  always holds.

In general, though, the other direction does not hold. If we take projections of a relation and recombine them using natural join, We typically obtain the tuples that were not in the original relation.

### CLOSURE OF DEPENDENCIES

1. We need to consider all functional dependencies that hold. Given a set  $F$  of functional dependencies, we can prove that certain other ones also hold. We say these ones are **logically implied** by  $F$ .

2. Suppose we are given a relation scheme  $R = (A, B, C, G, H, I)$ , and the set of functional dependencies:

$A \rightarrow B$   $A \rightarrow C$   $CG \rightarrow H$   $CG \rightarrow I$   $B \rightarrow H$

Then the functional dependency  $A \rightarrow H$  is logically implied.

3. To see why, let  $t_1$  and  $t_2$  be tuples such that  $t_1[A] = t_2[A]$

As we are given  $A \rightarrow B$ , it follows that we must also have  $t_1[B] = t_2[B]$

Further, since we also have  $B \rightarrow H$ , we must also have  $t_1[H] = t_2[H]$

Thus, whenever two tuples have the same value on  $A$ , they must also have the same value on  $H$ , and we can say that  $A \rightarrow H$ .

4. The **closure** of a set  $F$  of functional dependencies is the set of all functional dependencies logically implied by  $F$ .

5. We denote the closure of  $F$  by  $F^+$ .



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit IV Batch: 2017-2020

6. To compute  $F^+$ , we can use some rules of inference called

### Armstrong's Axioms:

Reflexivity rule: if  $\alpha$  is a set of attributes and  $\beta \in \alpha$ , then  $\alpha \rightarrow \beta$  holds.

- **Augmentation rule:** if  $\alpha \rightarrow \beta$  holds, and  $\gamma$  is a set of attributes, then  $\gamma\alpha \rightarrow \gamma\beta$  holds.
- **Transitivity rule:** if  $\alpha \rightarrow \beta$  holds, and  $\beta \rightarrow \gamma$  holds, then  $\alpha \rightarrow \gamma$  holds.

7. These rules are **sound** because they do not generate any incorrect functional dependencies. They are also **complete** as they generate all of  $F^+$ .

8. To make life easier we can use some additional rules, derivable from Armstrong's Axioms:

- **Union rule:** if  $\alpha \rightarrow \beta$  and  $\alpha \rightarrow \gamma$ , then  $\alpha \rightarrow \beta\gamma$  holds.
- **Decomposition rule:** if  $\alpha \rightarrow \beta$  holds, then  $\alpha \rightarrow \beta$  and  $\alpha \rightarrow \gamma$  both hold.
- **Pseudotransitivity rule:** if  $\alpha \rightarrow \beta$  holds, and  $\gamma\beta \rightarrow \delta$  holds, then  $\alpha\gamma \rightarrow \delta$  holds.
- 

9. Applying these rules to the scheme and set  $F$  mentioned above, we can derive the following:

- $A \rightarrow H$ , as we saw by the transitivity rule.
- $CG \rightarrow HI$  by the union rule.
- $AG \rightarrow I$  by several steps:
  - Note that  $A \rightarrow C$  holds.
  - Then  $AG \rightarrow CG$ , by the augmentation rule.
  - Now by transitivity,  $AG \rightarrow I$ .



## KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: II BCA Course Name: DATA BASE MANAGEMENT SYSTEMS

Course Code : 17CAU402 Unit IV Batch: 2017-2020

### POSSIBLE QUESTIONS

#### UNIT 1

#### PART – A (20 MARKS)

(Q.NO 1 TO 20 Online Examinations)

#### Part - B (5X 6 =30 Marks) (Answer ALL the Questions)

1. Discuss on functional dependencies of relational database design in detail with a suitable example.
2. What is meant by Normalization of relations? Define the keys and attributes of a relation scheme.
3. Write a note on First normal form of relational database design.
4. Define first, second and third normal forms. How do the 2 NF and 3 NF differ from 1 NF.
5. What undesirable dependencies are avoided when a relation is in a) 2 NF and b) 3 NF.
6. Define Boyce-Codd Normal form. How does it differ from 3NF?
7. Write a short note on multivalued dependency.
8. Define fourth normal form. When is it typically applicable and when is it violated?
9. Explain Join dependency and fifth normal form.
10. Illustrate an algorithm to check if decomposition is lossless.
11. Illustrate an algorithm to check if a decomposition is dependency preserving.

#### Part - C (10X1 =10 Marks) (Compulsory Question)

1. Define the term locking. Explain the different types of locking used in concurrency control.
2. Explain in detail the 13 object oriented database management system Commandments.
3. Compare and contrast the traditional and object oriented DBMS.
4. Elaborate different Data Manipulation Language in SQL. Explain the different statements used in DML.
5. Discuss on functional dependencies of relational database design in detail with a suitable example.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Established Under Section 3 of UGC Act 1956)

COIMBATORE - 641 021

DEPARTMENT OF COMPUTER APPLICATIONS

March 2017 – 2020

Subject : Database Management Systems

UNIT V

Subject Code: 17CAU402

S.No	Questions	opt1	opt2	opt3	opt4	Answer
1	concurrency control deals with _____ properties	isolation	consistency	transaction	isolation and consistency	isolation and consistency
2	_____ is probably the most important parameter in distributed system	Level of concurrency	distributed	semantic	consistency	Level of concurrency
3	In serializability how many conflicts are used?	1	2	3	4	2
4	_____ algorithm synchronizes the concurrent execution of transaction early in their execution life cycle	pessimistic	ordering	optimistic	locking based	pessimistic
5	The pessimistic group consist of _____ algorithms	locking based	ordering	optimistic	pessimistic	locking based
6	one of the copies of each lock unit is designated as the primary copy is known to be _____	centralized locking	primary copy locking	decentralized locking	none of these	primary copy locking
7	The lock management duty is shared by all the sites of a network is called + _____	centralized locking	primary	decentralized locking	copy locking	decentralized locking
8	_____ class involves negotiating the execution order of transaction	timestamp order	hybrid	fragmentation	semantic	time stamp ordering
9	_____ is only one of the properties of time stamp generation	simplicity	nonconflict	time stamp	single	simplicity
10	Database function are handled by dedicated computer is called, _____	application server	data base server	parallel database system	client server	data base server
11	_____ is used to run the user interface	application server	data base server	parallel server	client server	client server
12	Distributed database technology can be naturally revised and extended to implement _____	application server	data base server	parallel database system	client server	parallel database system
13	The main computer executing the application programs was termed the _____	Host computer	data base computer	back end computer	server	Host computer
14	The dedicated computer was called, _____	database machine	parallel database computer	back end computer	centralized database machine	database machine
15	Data base server fits naturally in a client_server or distributed environment has an advantages of _____	database server	application server	parallel server	client server	database server
16	Any workstation could access the data at any database server through either the _____	local network	local access	local network and access	global access	local network and access
17	The value of _____ used to create efficient database management	parallel system	client system	server system	remote system	parallel system
18	The parallel database system support the _____ interface	database function	client server	database function and client server	master slave	database function and client server
19	_____ plays the role of a transaction monitor	session manager	request manager	data manager	system manager	session manager
20	Request manager receives client requests related to _____	compilation & query	compilation & query	run & error	data & directory execution	compilation & execution
21	Which one provides the low level functions needed to run compiled queries in parallel?	session manager	request manager	data manager	system manager	data manager
22	Parallel system architectures range between two extremes the _____ architectures	shared memory	shared nothing	shared memory and shared nothing	shared everything	shared memory and shared nothing
23	shared_memory has two strong advantages they are, _____	meta information	control information	load behaviour	simplicity & load behaviour	simplicity & load behaviour
24	shared memory has a problem of _____	cost	limited extensibility	low availability	consistency	limited extensibility
25	shared disk suffer from _____ problems	higher complexity and potential	higher complexity and potential	higher complexity and potential	simplicity	higher complexity and potential
26	In _____ each processor has exclusive access to its main memory and disk units	shared disk	shared memory	shared nothing	shared	shared nothing
27	A _____ is a collection of resource maps and processed related attributes that user can assign to a user	roles	profiles	user	groups	profiles
28	_____ parameters are purely concerned with limiting resource usage	resource parameter	password parameter	system parameter	memory parameter	resource parameters
29	_____ specifies the total time a system may remain connected to the database	SQL TIME	LOGICAL TIME	CONNECT TIME	WHOLE TIME	CONNECT TIME
30	A _____ is used to group together similar users based on their resource needs	group user	resource plan	resource commit	resource cluster	resource commit group
31	The _____ lays out how resource constraint groups are allocated resources	resource group	resource parameter	resource plan	resource cluster	resource plan
32	The _____ privilege has to been granted to enable database resource manager	OTHER RESOURCE	RESOURCE MANAGER	OTHER RESOURCE	RESOURCE MANAGER	RESOURCE MANAGER
33	_____ is used to validate the changes before their implementation	submitting area	validate area	pending area	query area	pending area
34	_____ is used to assign resources to the various resource constraint group	resource plan	resource group	resource plan directory	resource plan	resource plan directory
35	The _____ view shows all currently active resource plan	V\$RESOURCE	V\$RESOURCE_PLAN	V\$CURRENT	V\$ACTIVE	V\$RESOURCE_PLAN
36	A _____ is the right to execute a particular type of SQL statement or to access a database object owned by another user	roles	privilege	grant	accept	privilege
37	The SYSDBA system privilege includes the _____ privilege and has all system privileges with ADMIN OPTION	SYSTEM PRIVILEGE	SYSTEM PRIVILEGE	SYSTEM PRIVILEGE	SYSTEM PRIVILEGE	SYSTEM PRIVILEGE
38	_____ are privileges on the various types of database objects	system privilege	object privileges	query privileges	sysdba privileges	object privileges
39	ALTER and SELECT comes under _____ privileges	view privileges	system privilege	object privileges	sequence privilege	sequence privileges
40	_____ is based on statement, privilege and object level auditing	statement auditing	statement auditing	statement auditing	statement auditing	statement auditing
41	A _____ specifies the auditing of all actions on any type of object	statement level	statement level	statement level	statement level	statement level
42	_____ involves the copying of database files	physical backup	logical backups	server backups	client backups	physical backups
43	A _____ needs a recovery	physical backup	logical backups	inconsistent backup	consistent backup	inconsistent backups
44	An _____ backup is a backup in which the files contain data from different points	physical backup	logical backups	inconsistent backup	consistent backup	inconsistent backups
45	_____ are backup made using the timestamp export utility	physical backup	logical backups	inconsistent backup	consistent backup	logical backups
46	_____ can create and manage backups on disk and on tape devices	RMAN	ASPI	AWR	ACC	RMAN
47	_____ script is kept in the RMAN recovery catalog	text	stored	valid	table	stored
48	_____ script is kept in regular text files	text	stored	valid	table	text
49	The _____ option lets you to specify how many copies of the backups you want	COPY	REDUNDANCY	RETENTION W	BACKUP COPY	REDUNDANCY
50	The _____ option ensures that RMAN doesn't perform a file backup if it has already been backed up	BACKUP OPTI	BACKUP RETE	BACKUP PAR	BACKUP COPY	BACKUP OPTIMIZATION

51	The _____ view displays all the online redo logs for the database	VSLOG	VSBACKUP	VSREDO	VSACTIVE	VSLOG
52	_____ corruption occurs when you have inconsistent data in tables or indexes	repair	data block	block	files	data block
53	_____ tool if used from the operating system to check the structural integrity of the database files for corruption	DBVERIFY	DBATTACH	DBANALYZE	DBREPAIR	DBVERIFY
54	You can use the _____ command to validate backup sets before you use them for recovery.	VALIDATEBACKUP	VALIDATEBACKUPSET	VALIDATEVOLUME	VALIDATECHECKPOINT	VALIDATEBACKUPSET
55	_____ are the means by which RMAN conducts its backup and recovery operations	Channels	volumes	triggers	functions	Channels
56	The default value of <i>degrees of parallelism</i> is _____	2	4	5	1	1
57	The _____ command will restore the recovery catalog	RESTORE CATALOG	RECOVER CATALOG	DROP CATALOG	DISC CATALOG	DROP CATALOG
58	The _____ option tells the RMAN to finish the backup as fast as it can.	PARTIAL	MINIMIZE TIME	MINIMIZE LOAD	DETERMIN	MINIMIZE TIME
59	The _____ file is used to track the physical location of all database block changes	modify tracking	track files	track database	change-tracking	change-tracking
60	_____ contains the Oracle databases that are backed up by Oracle Backup	Media server	Client host serve	Administrative s	client server	Client host server
61	_____ is the frequent writing of the data database buffers in the cache to disk by the database writer	Fast Archiving	Fast Start Checkpointing	Fast Start Checkpointing	Fast Start recovery	Fast Start Checkpointing
62	Bringing the data files up to date using backed-up data files and archived redo log files is called _____	restoring	recovery	backing	backing	recovery
63	The process of applying the contents of both the archived and redo log files to bring the data files up to date is called _____	instreamation reco	open recovery	cache recovery	closed recovery	cache recovery
64	A _____ is a recovery for which you must to shut down the database completely.	instreamation reco	open recovery	cache recovery	closed recovery	closed recovery
65	When you use the _____ command, RMAN restores a data file from an image backup	RESTORE	RECOVER	BACKING	BLOCKING	RESTORE
66						