KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2017 onwards)

DEPARTMENT OF COMPUTER SCIENCE, COMPUTER APPLICATION & INFORMATION TECHNOLOGY

SUBJECT : RELATIONAL DATABASE MANAGEMENT SYSTEM
SEMESTER : III
SUBJECT CODE: 17TTU303 **CLASS : II B.Sc.IT**

SCOPE

It is an entry-level course for creating database systems. This course is designed for students who have an interest in developing applications. It is used to understand the role and nature of relational database management systems (RDBMS) in today's IT environment.

OBJECTIVES

- To understand the application development environment.
- To gain programming Skills and Database Creation in RDBMS.
- Ability to use SQL for storing and retrieving data from the RDBMS.
- Ability to arrive at a normalized design of tables and other database objects in RDBMS.
- Ability to use PL/SQL

UNIT-I

Origin of database – database elements – design concepts – components ofDBMS – Advantages and disadvantages of DBMS. Database Models: flat file – hierarchical model – network model – relational model – object oriented model – Features of Object oriented Database Management system – Features of distributed DBMS – Comparison of DBMS & DDBMS – Object relational model. ER-model: entities – relationships - ERD symbols – cardinalities – sample ERD.

UNIT-II

Relational model: Introduction – Relational database: attributes and domain – Tuples – Relation and their schemes – Relation representation – keys – relationships – relational operations – Integrity rules. Relational algebra: Basic operations – Additional relational algebraic operations – some relational algebra queries. Functional Dependency: Reasoning about FD's – closure of set of FD's – Attribute closure.

UNIT-III

Relational database manipulation: Introduction – SQL: Data definition – Data manipulation: Basic data retrieval – condition specification – Arithmetic and aggregate operations. SQL joins – set manipulation – categorization – updates – views – index. Data Control language : grant, revoke – simple privileges.

UNIT-IV

Declaration section – executable command section : conditional logic, loops, CASE statements – exception handling section: predefined and user defined exceptions. Triggers : definition – types : row level, statement level, before and after, instead of – syntax – enabling and disabling triggers - replacing and dropping triggers. Cursors – definition – open – fetch – close – cursor attributes- select for update – types : implicit, explicit. Procedures, Functions: Local and global – procedures vs functions – stored procedures, functions – create procedure syntax - create function syntax – calling procedures, functions. Replacing and dropping procedures, functions.

UNIT-V

Package header – package body – calling package members - Replacing and dropping package. Overview of Normalization : advantages - disadvantages. Normal forms: first normal form – second normal form – third normal form – boyce codd normal form – Introduction to fourth, fifth and sixth normal forms – denormalization. Parallel Databases: Introduction – Design of Parallel Databases – Advantages and Disadvantages of Parallel Database.

Suggested Readings

1. Bipin C. Desai.(2013). An Introduction to Database Systems, New Delhi: Galgotia Publications.
2. Rajiv chopra (2013). Database Management systems (3rd ed.). S.Chand publications.
3. Steven Feurstein, Bill Pribyl (2014). Oracle PL/SQL Programming (6th ed.). O ' Reilly Media.
4. Shio Kumar Singh (2011). Database Management Systems – Concepts, design and Applications (2nd ed.). New Delhi: Pearson Education.
5. Ivan Byross (2010). SQL, Pl/SQL the Programming Language of Oracle Paperback. BPB Publications.
6. Rajeeb C. Chatterjee (2012). Learning Oracle SQL and Pl/SQL: A simplified Guide. Prentice Hall of India.

Web Sites

1. http://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm
2. www.databasedir.com
3. http://plsql-tutorial.com/

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2017 onwards)

DEPARTMENT OF COMPUTER SCIENCE, COMPUTER APPLICATION & INFORMATION TECHNOLOGY

SUBJECT : **RELATIONAL DATABASE MANAGEMENT SYSTEM**
SEMESTER : **III**
SUBJECT CODE: **17ITU303** **CLASS** : **II B.Sc.IT**

LECTURE PLAN

S. No	Lecture Duration (Hr)	Topics to be Covered	Support Materials
		UNIT – I	
1.	1	Origin of Database, Database Fundamentals	W1, W2
2.	1	Database Elements, Design concepts	S1:2-8
3.	1	Components of DBMS – Classification of DBMS Users, DBMS Facilities, Structure of DBMS Advantages and Disadvantages of DBMS	S1:20-23 S1:23-30
4.	1	Database Models: Flat-file, Hierarchical model, Network model Relational model	S2:59-61 S2: 62-65 S2:66-69
5.	1	Object oriented model – Features of Object oriented DBMS, Approaches of Object Oriented DBMS	S2: 755-758
6.	1	Comparison of DBMS & DDBMS	S1:661-663
7.	1	Object relational model, ER model – entities, types of attributes	S1:46-49
8.	1	Relationships, ERD Symbols, sample ERD	S1:46-49
9.	1	Recapitulation and discussion of important	

		questions	
	Total no. of Hours planned for Unit – I		9

Suggested Readings

S1: Bipin C. Desai. 2013. An Introduction to Database Systems, Galgotia Publications, New Delhi

S2: Database Management system, 3rd revised edition 2013, Rajiv chopra, S. Chand publication

Web Sites

W1: www.databasedir.com

W2: www.rdbms.org

UNIT II			
1.	1	Relational Model: Introduction, Relational database – attributes and domains, Tuples	S1: 153-157
2.	1	Relation and their schemes – Relation representation keys	S1: 157-161
3.	1	Relationships – relational operations	S1: 157-161
		Integrity rules	S1: 162-165
4.	1	Relational Algebra: Basic operations	S1: 167-169
			S1: 167-169
5.	1	Additional relational algebraic operations	S1: 179-184
6.	1	Some Relational algebra queries	S1: 179-184
			S1: 179-184
7.	1	Functional dependency: Reasoning about FD's	S1: 353-354
8.	1	Closure of set of FD's - Attribute closure	S1: 297-298
9.	1	Recapitulation and discussion of important questions	
	Total no. of Hours planned for Unit-II		9

Suggested Readings

S1: Bipin C. Desai. 2013. An Introduction to Database Systems, Galgotia Publications, New Delhi

		UNIT – III	
1.	1	Relational database manipulation: Introduction SQL: Data definition	S1: 208-212
2.	1	Data manipulation: Basic data retrieval - Condition specification	S1: 214-219
3.	1	Arithmetic operations	S1: 219-220
		Aggregate operations	S1: 220-221
4.	1	SQL joins	S1: 221-228
5.	1	Set manipulation	S1: 228-233
6.	1	Categorization - Updates	S1: 233-236
7.	1	Views	S1: 237-249
		Index	S1: 240-242
8.	1	Data Control language: grant-revoke	S3: 362-363
		Simple privileges	S3: 480-482
9.	1	Recapitulation and discussion of important questions	
	Total no. of Hours planned for Unit – III		9

Suggested Readings

S1: Bipin C. Desai. 2013. An Introduction to Database Systems, Galgotia Publications, New Delhi

S3: Kevin Loney and George Koch. 2002. Oracle 9i The Complete Reference, 1st Edition, Tata Mcgraw-Hill, New Delhi

		UNIT – IV	
1.	1	PL/SQL – Overview, Declarations and Executable command section	S3: 490-496, W5
2.	1	Conditional and CASE statements, Iterative	S3: 496-502, W5

		statements	
3.	1	Exception handling section - Predefined exceptions	S3: 503-506
4.	1	User defined exception, Sample Exception program	S3: 506-509
5.	1	Triggers – types, Row level and statement level, before and after	S3: 511-514
		Instead of – syntax – enabling and disabling triggers	S3: 511-514
		Replacing and dropping triggers	S3: 511-514
6.	1	Cursor – definition, open, fetch and close, Cursor attributes, Implicit and Explicit attributes	S1: 189-191
			S1: 189-191
7.	1	Local and global, procedure and functions Creating stored procedure – Stored procedures	S3: 532-542
8.	1	Replacing and dropping – Procedures and function	S3: 547
9.	1	Recapitulation and discussion of important questions	
	Total no. of Hours planned for Unit – IV		9

Suggested Readings

S1: Bipin C. Desai. 2013. An Introduction to Database Systems, Galgotia Publications, New Delhi

S3: Kevin Loney and George Koch. 2002. Oracle 9i The Complete Reference, 1st Edition, Tata Mcgraw-Hill, New Delhi

Web Sites

W5: http://mobile.dudasite.com/site/plsql-tutorial-1/default.htm

UNIT - V			
1.	1	Package – package header	S3: 542-547

		- Package body, Creating package - Calling package members	
2.	1	- Replacing and dropping package	S3: 547
3.	1	Overview of Normalization, Advantages and disadvantages of Normalization	S5: 615
4.	1	Normal Form - First normal form - Second normal form - Third normal form	S5: 615-619
5.	1	Boyce codd normal form, Demoralization	S5: 615-617, 652
6.	1	Introduction to fourth, fifth and sixth NF	S1: 357, 356-364
7.	1	Parallel Databases: Introduction - Design of parallel db	S2: 460-461, W3
8.	1	Advantages and Disadvantages of Parallel Database	S2: 461-463, W3
9.	1	Recapitulation and discussion of important questions	
10.	1	Previous ESE Question Paper Discussion	
11.	1	Previous ESE Question Paper Discussion	
12.	1	Previous ESE Question Paper Discussion	
	Total no. of Hours planned for Unit – V		12

Suggested Readings

S1: Bipin C. Desai. 2013. An Introduction to Database Systems, Galgotia Publications, New Delhi

S2: Database Management system, 3rd revised edition 2013, Rajiv chopra, S. Chand publication

S3: Kevin Loney and George Koch. 2002. Oracle 9i The Complete Reference, 1st Edition, Tata Mcgraw-Hill, New Delhi

S5: Ragu Ramakrishnan and Gehrke. 2003. Database Management Systems Third Edition McGraw-Hill, New Delhi

Web Sites

W3: http://docs.oracle.com/cd/A58617_01/server.804/a58238/ch1_unde.htm

Faculty HOD

UNIT-I

SYLLABUS

Origin of database – database elements – design concepts – components of DBMS – Advantages and disadvantages of DBMS. Database Models: flat file – hierarchical model – network model – relational model – object oriented model – Features of Object oriented Database Management system – Features of distributed DBMS – Comparison of DBMS & DDBMS – Object relational model. ER-model: entities – relationships - ERD symbols – cardinalities – sample ERD.

1.1 Origin of the database

Business modeling is the process of evaluating and capturing the daily tasks performed by a business, and the origination of any database. The following subsections explain some basic concepts involved in business modeling. These concepts include business rules, business processes, data, analysis, entities, attributes, and re-engineering.

Business Rules
Businesses have rules. These rules affect how the business operates in many different ways. From the perspective of designing a database, business rules are important because they tell us how data is created, modified, and deleted within an organization. Rules place restrictions and limitations on data, and ultimately help determine the structure of the database, as well as the application used to access the database.

Two broad categories of business rules associated with the design of a database system are as follows:

- Database-oriented
- Application-oriented

Database-oriented rules are those that affect the logical design of the database. These rules affect how the data is grouped and how tables within the database are related to one another. Application-oriented rules deal with the operation of an application through which a user interfaces with the database. Data edits can be built into the application interface as a check and balance against the constraints that reside in the database.

Business Processes
Business processes deal with the daily activities that take place. Business processes are conducted either manually, by individuals within the organization, or they are automated. Businesses function through business processes. Data is entered into the database through some business process. Business rules affect how the data can be entered.

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 .

There are basically two types of date that reside in any database:

• Static, or historic
• Dynamic, or transactional

Static or historic data is seldom or never modified once stored in the database. Historic data never changes.
Dynamic or transactional data is data that is frequently modified once stored in the database. At a minimum, most companies have dynamic data.

Requirements Analysis
Requirements analysis is the process of analyzing the needs of a business and gathering system requirements from the end user that will eventually become the building blocks for the new database. During requirements analysis, business rules and processes are taken into consideration.

In the analysis phase of a system, a requirements document should be established that outlines the following basic information:
• Objectives and goals of the business as it pertains to the proposed system
• A list of proposed requirements for the system
• A list of business processes and rules
• Documentation for current business processes or documentation from the legacy system

Entities
An entity is a business object that represents a group, or category of data. For example, a category of information associated with an online bookstore is book titles. Another category is authors because an author might have written many books. Entities are objects that are used to logically separate data.

Attributes
An attribute is a sub-group of information within an entity. For example, suppose you have an entity for book titles. Within the book titles' entity, several attributes are found, such as the actual title of the book, the publisher of the book, the author, the date the book was published, and so on. Attributes are used to organize specific data within an entity.

Business Process Re-engineering
Business process re-engineering (BPR) is the task of reworking business processes in order to streamline the operations of an organization. BPR may involve redesigning an existing system in order to improve methods for storing and accessing the data in conjunction with the business

processes that have been refined.

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 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.
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.

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).

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

1.7 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

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.

Fig 1.9 Class and Object

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 throughout 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.

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 safe 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 refusing 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 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 to 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 system. The usage of both system defined types and user-defined types is same.

Property of Computational Completeness
This feature of OODBMS implies that does 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.

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

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.

Possible Questions
Part-A
(Online – Multiple choice questions)
Included in Excel file – File name Unit-I(MCQ).xls
(Each questions carries one mark each)

PART – B (2 Marks)
1. What are attribute and its types?
2. Write a short note on DBA?
3. Write the advantages and disadvantages of relational model.
4. What is object oriented database model?
5. Write a short note on cardinality with example

PART – C (8 Marks)
1. Discuss in detail about distributed DBMS.
2. Differentiate between Hierarchical and Network database models with a neat sketch. Write the pros and cons of each model.
3. Define a DBMS. Classify the types of users of a DBMS based on the degree their expertise.
4. Compare DBMS and DDBMS.
5. Elaborate in detail the various design concepts of a database
6. Discuss in detail about any three database models and compare them with their
7. benefits and drawbacks.
8. Enumerate in detail various types of Relationships with examples for each.
9. Enumerate in detail the major components involved in the structure of a DBMS.
10. Describe ER Medel in detail with its attributes classification.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

Subject: Relational Database Management System Sub.code: 17ITU303

S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____is a Collection of data	DBMS	Database	Relationships	Entities	**Database**
2	_____is a software designed to assist in maintaining and utilizing large collections of data.	Database	DBMS	Entities	attributes.	**DBMS**
3	_____ data is never modified once stored in the database	Dynamic	Static	Physical	Logical	**Static**
4	_____is the task of reworking business processes in order to streamline the operations of an organization	Business Rules	Business Modelling	Business Process Reengineering	Requirement Analysis	**Business Process Reengineering**
5	_____ model is also called Application interface	External	Internal	Conceptual	Logical	**External**
6	_____was adopted by the ANSI and ISO.	PSQL	SQL	R-SQL	Sequel	**SQL**
7	_____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**

8	Database management System that exhibits related tables are based on _____	Network model	Hierarchical model	Relational kmodel	Object-based model	**Relational model**
9	A model that pictorially represents a schema of a table is called _____	Network model	ER Model	Object-based model	Hierarchical model	**ER Model**
10	The internal structure of a database is represented by _____ model	ER model	Semantic data model	Logical data Model	Physical data model	**Semantic data model**
11	_____model is used to represent parent-child relationship.	Physical data model	ER model	Hierarchical model	structure chart	**ER model**
12	A structure of data in a data model is called_____	Schema	relation	record	entities	**Schema**
13	Field is otherwise known as _____	Column	Entity	Relationship	Relation	**Column**
14	Column is otherwise known as _____	Entity	Relationship	Relation	attribute	**attribute**
15	_____is used to define schemas of a table.	DDL	DML	DCL	TCL	**DDL**
16	Information about the conceptual,external and physical schemas is stored in _____	Directory	System Catalogs	IMS	Information System	**System Catalogs**
17	Which is not a commonly used data type in SQL	StringBuffer	Alphanumeric	Number	Date	**StringBuffer**
18	Conceptual schema otherwise called as _____	Physical Schema	Internal Schema	Logical Schema	relations	**Logical Schema**
19	Physical Schema specifies _____details	Information	data	Storage	relationships	**Storage**

20	_____is used to speed up data retrieval operations.	DML Operations	Select Operation	Indexes	select operation with where condition	**Indexes**
21	The process of arriving at a good physical schema is called_____	Conceptual database design	Physical database design	Logical Schema	schema	**Physical database design**
22	A DBMS enable users to create, modify and query data through a _____	DDL	DML	TCL	DCL	**DML**
23	A DBMS provides a specialized language called_____	Grammar	English-like language	Object Oriented Language	Structured Query Language	**Structured Query Language**
24	_____is a formal query language based on mathematical logic	Relational calculus	Relational Model	Algebra	SQL	**Relational calculus**
25	DML and DDL are collectively referred to as the _____	Connectivity	Common language	Data sub language	Data Manipulation language	**Data sub language**
26	_____is any one execution of a user program in a DBMS	Compiled program	transaction	user output	concurrent execution	**transaction**
27	_____is a mechanism used to control access to database objects	stop	terminate	Authorization	Restriction	**Authorization**
28	Referential Integrity is implemented using _____	Candidate key	Super Key	Primary key	Foreign key	**Foreign key**
29	In _____ relationship one record in a table can be related to many records in another table	one-many	one-one	many-many	many-one	**one-many**

30	_____is a collection of pages or a collection of records.	Indexes	record blocks	files	Query	**files**
31	_____manager is responsible for maintaining a log and restoring the system to a consistent state after a crash.	lock manger	recovery manager	buffer manager	transaction manager	**recovery manager**
32	_____is an object in the real world	Entity	Attribute	Relationship	Property	**Entity**
33	Collection of similar entities are called_____	Attributes	Entity	Entity Set	Relationship	**Entity Set**
34	An Entity is described using a set of _____	Entity	Entity Set	Attributes	Relationship	**Attributes**
35	_____is used to uniquely identify an entity in the set.	Key	Lock	Attributes	Entity	**Key**
36	_____is used to uniquely identify a particular row	Candidate Key	Primary Key	foreign Key	Referential key	**Primary Key**
37	A _____is an association among two or more entities	Attributes	Entity Sets	Key	Relationships	**Relationships**
38	Indicated by using arrow from entities to relationships in the ER diagram.	Arrow	Thick line	Dotted line	Shaded line	**Arrow**
39	Derived attribute is indicated by _____in ER diagram	Solid line	Thick line	Thin line	Dotted line	**Dotted line**
40	_____is a set of associated values	Entity	Attribute	Relationships	Domain	**Domain**
41	_____consists of a relation schema and a relation instance.	relation	table	domain	entity	**relation**

42	An instance of a relation is a set of _____	tuple	domain	attribute	relationships	**tuple**
43	Each tuple is a _____	Column	row	table	instance	**row**
44	Which among the following is not used to develop Application interface?	Power Builder	Visual Basic	Oracle	Developer 2000	**Oracle**
45	Centralized control of the database is exerted by _____ user	Naïve	Online	DBA	Application Programmer	**DBA**
46	_____ users communicate with the database directly via an online terminal	Naïve	Online	DBA	Application Programmer	**Online**
47	_____ is the central component of DBMS also called as database control system	File Manager	Disk Manager	Query Processor	Data Manager	**Data Manager**
48	Data Dictionary is also called _____	Data files	Meta Data	Data compiler	Data manager	**Meta Data**
49	_____ performs all physical I/O operations	Disk manager	Data dictionary	Data Manager	File Manager	**Disk manager**
50	The property of Inheritance is a main feature of _____	DBMS	RDBMS	OODBMS	DDBMS	**OODBMS**
51	_____ is the process of reconstructing the data in case of failure	Recovery	Backup	Integrity	Consistency	**Recovery**
52	_____ is a logically interrelated collection of shared data physically distributed over a computer network.	Database	Relational Database	Object oriented database	Distributed Database	**Distributed Database**
53	_____ relationship uses the keyword 'must'	Optional	Mandatory	Recursive	Many-many	**Mandatory**
54	_____ relationship uses the keyword 'may'	Optional	Mandatory	Recursive	Many-many	**Optional**

55	_____ is a circular relationship that exists between two attributes in the same entity.	Optional	Mandatory	Recursive	Many-many	**Recursive**
56	The degree of a relation is also called _____	arity	instance	relation	Schema	**arity**
57	_____is the count of the number of entities involved in that relationship	degree	cardinality	entity set	entity type	**cardinality**
58	If two entities are involved in a relation it is called _____	Unary	Binary	Ternary	None	**Binary**
59	If three entities are involved in a relation it is called _____	Unary	Binary	Ternary	None	**Ternary**
60	If only one entity is involved in a relation it is called _____	Unary	Binary	Ternary	None	**Unary**

UNIT-II
SYLLABUS

Relational model: Introduction – Relational database: attributes and domain – Tuples – Relation and their schemes – Relation representation – keys – relationships – relational operations – Integrity rules. Relational algebra: Basic operations – Additional relational algebraic operations – some relational algebra queries. Functional Dependency: Reasoning about FD's – closure of set of FD's – Attribute closure.

.

Relational Model

2.1. Introduction

In modeling, we represent an entity set in the database by a set of its properties. Only those properties of the entity type of interest to the application are used in the model. A data model allows capturing of these properties using its data structures.

The relational model like all data models, consists of three basic components:

- A set of domains and a set of relations
- Operations on relations
- Integrity rules

Eg. A sample relation **Student**

Reg_No	Name
14ITU001	Ramesh
14CSU001	Sunitha
14CAU004	Arun
14ITU002	Sudha
14CSU002	Ravi

A Relation with its attributes are specified as a relation scheme with the general format

Relation_Scheme_Name(Attribute_name1,Attribute_name2,…..).

For the above example it is represented as

Student(Reg_No, Name)

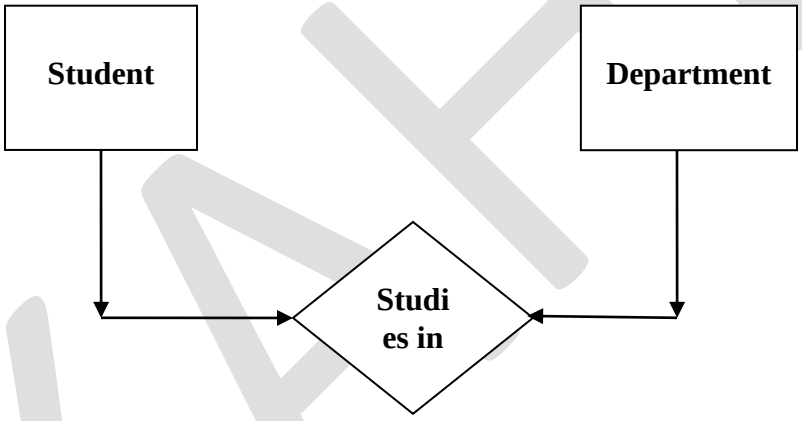
- The "Student" relation consists of 6 tuples. So its cardinality is 6.
- The number of attributes is called degree or arity. The degree of Student relation is 2.

- Here the attribute Reg_No can be used to uniquely identify a student.

Consider there is another relation called Department with the following attribute domain

Course_code	Course_name
111	IT
112	CS
113	CA

A relationship can be exhibited between these two relations as follows



2.1.1 A Brief Review of Set Theory

A set is well-defined collection of objects. It is commonly represented by a list of its elements (called members). Operations on sets include union, intersection, Cartesian product and difference.

Example 1

G = {BCD}

H = {WXY, BCD, ABC}

The union of two sets G and H is the set that contains all elements belonging either to set G or to set H.

G U H = {BCD, WXY, ABC}

The intersection of the sets G and H is the set composed of all elements belonging to both G and H.

G ∩ H = {BCD}

The difference of two sets G and H is the set that contains all the elements that are member of G but not of H.

G – H = { φ } or {null set}

H – G = {WXY, ABC}

Example 2

The Cartesian product of two sets J and K is defined in terms of ordered pairs or 2-tuples.

J = {BCD, ABC}

K = {Hierarchical, Relational}

J X K = {(BCD, Hierarchical), (BCD, Relational), (ABC, Hierarchical),(ABC, Relational)}

2.2 Relational Database

2.2.1 Attributes and Domains

An object or entity is characterized by its properties also called as attributes. An attribute may only be allowed to take a value from a set of permissible values. The set of allowable values for the attribute is called domain.

Example

If persons can only be between 0 and 255 years of age, then attribute is defined as

P_Age: {x | x is a positive integer and $0 \leq x \leq 255$}

A domain D_i id defined as a set of values of the same data type. Domain D_i is said to be simple if all its elements are non-decomposable. Atomic domains are also referred to as **application-independent domains** because these general sets are not dependent on a particular application.

Structured or composite domains can be referred to as application dependent domains that consists of non-atomic values. For example Address field specifies street number, street name, city, state, pincode etc.

2.2.2 Tuples

An entity type having n attributes can be represented by an ordered set of theses attributes called an n-tuple. The representation of the entity must be a member of the set D_1 X D_2 ..... D_n as the resulting set of this Cartesian product contains all possible ordered n-tuples.

Tuples are generally denoted by lowercase letters r,s,t, …. of the alphabet. An n-tuple can be specified as

$$T= (a_1 \ldots..,a_n)$$

Where each a_i for $1 \leq i \leq n$ is a value in the domain D_i and is the value of the attribute A_i in the tuple t.

Example:

An applicant, John Doe, who is 55 years old and is an analyst, may be represented as a 3-tuple: John Doe, 55, analyst".

Let t = (John Doe, 55, analyst). Then we have the following projections

t [Name,Profession] = (John Doe, analyst)

t [Name, Age] = (John Doe, 55)

t [Age, Profession]=(55,analyst)

t [Name]=(John Doe)

A projection that re-orders the attributes can also be used

t [Name, Profession, Age]= (John Doe, analyst, 55)

2.2.3 Relations and their Schemes

A relation consists of homogeneous set of tuples. A relation can also be viewed as a Cartesian product of the domain. The set of tuples in the relation are not static but can vary with time. The set of attributes on which the tuples are defined is the invariant. This is called the scheme of the

relation or the relation scheme.

The relation has two parts: a relation scheme (or header) and a time varying set of tuples (or body). The ordering of the attributes in the scheme is immaterial, however, the tuple layout should match this ordering.

Example for relation schemes:

1. Applicant (Name, Age, Profession)
2. Student (Reg_no, Name)
3. Department(Dept_code, Dept_name)

A relation scheme can be represented as $R(A_1, \ldots, A_{n)}$, domain of each attribute A_i by D_i for $1 \leq i \leq n$, and define the relation R over the set of attributes R, denoted by R(R), as a set of n-tuples such that:

$$R(R) \subseteq D_1 \times D_2 \ldots\ldots D_n$$

The value n (the number of attributes in the relation) is known as the degree of relation or arity of the relation. A relation of degree one is called an unary relation, of degree two is called binary relation and of degree n is called an n-ary relation.

2.2.4 Relation representation

Conceptually a relation can be represented as a table. Each column of the table represents an attribute and each row represents a tuple of the relation. A tuple may be represented as a labeled n-tuple or as an ordered n-tuple. The labeled n-tuples are represented using distinct attribute names $A_1, \ldots, A_n$ and the values $a_1, \ldots, a_n$ from the corresponding domains.

Example

Applicant

Name	Age	Profession
John, Doe	55	Analyst
Mirian Taylor	31	Programmer
Abe Malcolm	28	Receptionist
Adrian Cook	33	Programmer
Liz Smith	32	Manager

2.2.5 Keys

In the instance of an Employee relation, values of an attribute such as Emp# may be sufficient to distinguish between employee tuples. Such a subset of attributes with the following time-independent properties is called the key of the relation:

a) Unique identification: In each tuple of R, the values of X uniquely identify that tuple.
b) Non-redundancy: No proper subset of X has the unique identification property, that is no attribute can be discarded without violating the unique identification property.

There may be more than one key in a relation; all such keys are known as candidate keys. One of the candidate keys is chosen as the primary key; the others are known as alternate keys. An attribute that forms part of a candidate key of a relation is called a prime attribute.

2.2.6 Relationship

The key property and the fact that every tuple must have a key are used to capture relationships between entities.

Example

An employee may perform different roles in the software development teams. John Doe may be an analyst for product “Super file system” and manager for team for product “B1”.

Assignment is a relationship between entities Employee, Product and Job_Function. A possible representation of this relationship is by using the entities involved in the relationship:

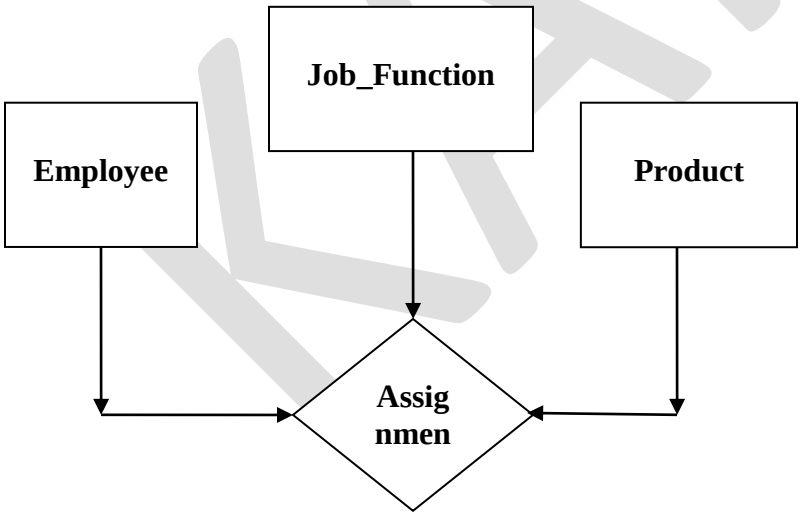
Assignment (Employee, Product, Job_Function)

Using the unique identification properties of keys we can replace Employee, Product and Job_Function entities in the Assignment with their keys that is,

Assignment (Emp#, Prod#, Job#)

Assignment is a relation that establishes a relationship between three "owner" relations. Such a relation may be thought of as an **associative relation**. The key of associative relation is always the key attributes of the owner relations.

The attributes Emp#, Prod# and Job# in the relation Assignment are known as foreign keys. A foreign key is an attribute or set of attribute of a relation such that the value of each attribute in this set is that of a primary key of relation.



Employee

Emp#	Name	Profession

101	Jones	Analyst
103	Smith	Programmer
104	Lalonde	Receptionist
106	Byron	Receptionist
107	Evan	VP R& D
110	Drew	VP Operation
112	Smith	Manager

Product

Prod#	Prod_Name	Prod_Details
HEAP1	Heap_Sort	ISS module
BINS9	Binary_Search	ISS/R module
FM6	File_Manager	ISS/R – PC subsys
B1	B++ Tree	ISS/R turbo sys
B2	B++ Tree	ISS/R – PC turbo

Job_Function

Job#	Title
1000	CEO
900	President
800	Manager
700	Chief Programmer
600	Analyst

Assignment

Emp#	Prod#	Job#

107	HEAP1	800
101	HEAP1	600
110	BINS9	800
103	HEAP1	700
101	BINS9	700
110	FM6	800
107	B1	800

2.2.7 Relational Operations

A number of operations are defined in tuple calculus and domain calculus approaches that is define by Codd. These approaches are used to manipulate the relations. Relations can be derived from other relations (by taking a subset of the set of attributes) or a number of relations can be combined to define a new relation (by joining the relations). The transformation of relations is useful in obtaining relation from the database.

2.2.8 Integrity Rules

The relational model includes two general integrity rules. These integrity rules implicitly or explicitly define the set of consistent database states or changes of state or both.

Integrity Rule 1 (Entity Integrity)

It is concerned with primary key values. A null value for an attribute is a value that is either not known at the time or does not apply at a given instance of the object. If any attribute of a primary key were permitted to have null values, then, because the attributes in the key must be non-redundant, the key cannot be used for unique identification of tuples.

Example

P1:

ID	Name

101	Jones
103	Smith
104	Lalonde
107	Evan
110	Drew
112	Smith

P2:

ID	Name
101	Jones
@	Smith
104	Lalonde
107	Evan
110	Drew
@	Lalonde
@	Smith

In the second table null values are permitted that are represented as @. Here the two tuples <@, smith> are indistinguishable, even though they may represent two different instances of the entity type employee. Integrity Rule 1 specifies that instances of the entities are distinguishable and thus no prime attribute value may be null. This rule is also called as Entity Rule.

Definition: Integrity Rule 1 (Entity Integrity)

If attribute A of Relation R(R) is a prime attribute, then A cannot accept null values.

Integrity Rule 2 (Referential Integrity)

It is concerned with Foreign keys that is with attributes of a relation having domains that are those of the primary key of another relation.

Relation (R) may contain references to another relation (S). They need not be distinct. Suppose

reference in R is via a set of attributes that forms a primary key of relation S. this set of attributes in R is a foreign key. Reference from a tuple of relation R is made unambiguously to an existing tuple in the S relation. The referencing attribute in relation R can have null values if it is not referencing any tuple in S relation.

Example:

P:

Emp#	Name	Manager
101	Jones	@
103	Smith	110
104	Lalonde	107
107	Evan	110
110	Drew	112
112	Smith	112

In this relation, each employee has a manager and manager are also employees. So we represent managers by their employee numbers. Manager is a foreign key that refers to primary key of same relation. CEO can have null values. Some employees may also be temporarily without managers, and this can be represented by the manager taking null values.

Definition: Integrity Rule 2(Referential Integrity)

Given two relations R and S, suppose R refers to relation S via a set of attributes that forms the primary key of S and this set of attributes forms a foreign key in R. Then the value of the foreign key in a tuple in R must either be equal to the primary key of a tuple of S or be entirely null.

If we delete a tuple that is a target of a foreign key reference, then three explicit possibilities exist to maintain database integrity:

- All tuples that contain references to the deleted tuple also should be deleted. It is called domino or cascading deletion.
- Only tuples that are not referenced by any other tuple can be deleted.

- The tuple is deleted. However, to avoid the domino effect, the pertinent foreign key attributes of all referencing tuples are set to null.

The choice of the option to use during a tuple deletion depends on the application. For example, in most cases it would be inappropriate to delete all employees under a given manager on the manager's departure; it would be more appropriate to replace it by null.

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 parity 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

Q

ID	Name
103	Smith
104	Lalonde
106	Byron
110	Drew

UNION (∪)

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 ∪ 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 ∪ 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 (∩)

The intersection operation selects the common tuples from two relations.

P∩Q

ID	Name
103	Smith
104	Lalonde
110	Drew

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 X 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:

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

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

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

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 nave 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 (π)

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

Selection(σ)

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 σ and it is sometimes known as restriction operation.

Consider the selection operation

σ id<105 (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 ∞ 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 (÷)

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:

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
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

3. Obtain details of employees working on Database project

Emp#	Name
101	Jones
106	Byron
107	Evan
112	Smith

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

Emp#	Name
106	Byron
112	Smith

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

Emp#
106

2.4 Functional Dependencies (FD)

Definition

Let R be a relation scheme and X, Y be sets of attributes in R. A functional dependency from X to Y exists if and only if:

- For every instance of |R| of R, if two tuples in |R| agree on the values of the attributes in X, then they agree on the values of the attributes in Y

We write X → Y and say that X determines Y

Example on Student (sid, name, supervisor_id, specialization):

- {supervisor_id} → {specialization} means
- If two student records have the same supervisor (e.g., Dimitris), then their specialization (e.g., Databases) must be the same
- On the other hand, if the supervisors of 2 students are different, we do not care about their specializations (they may be the same or different).

Sometimes, omit the brackets for simplicity:

- supervisor_id → specialization

Trivial FDs

A functional dependency X → Y is trivial if Y is a subset of X

- {name, supervisor_id} → {name}
- If two records have the same values on both the name and supervisor_id attributes, then they obviously have the same supervisor_id.
- Trivial dependencies hold for all relation instances

A functional dependency X → Y is non-trivial if Y∩X = ∅

- {supervisor_id} → {specialization}
- Non-trivial FDs are given implicitly in the form of constraints when designing a database.
- For instance, the specialization of a students must be the same as that of the supervisor.

- They constrain the set of legal relation instances. For instance, if I try to insert two students under the same supervisor with different specializations, the insertion will be rejected by the DBMS

Functional Dependencies and Keys

A FD is a generalization of the notion of a *key*.

For Student (sid, name, supervisor_id, specialization),

we write:

{sid} → {name, supervisor_id, specialization}

- The sid determines all attributes (i.e., the entire record)
- If two tuples in the relation student have the same sid, then they must have the same values on all attributes.
- In other words they must be the same tuple (since the relational modes does not allow duplicate records)

Superkeys and Candidate Keys

A set of attributes that determine the entire tuple is a **superkey**

- {sid, name} is a superkey for the student table.
- Also {sid, name, supervisor_id} etc.

A minimal set of attributes that determines the entire tuple is a **candidate key**

- {sid, name} is not a candidate key because I can remove the name.
- sid is a candidate key – so is HKID (provided that it is stored in the table).

If there are multiple candidate keys, the DB designer chooses designates one as the **primary key.**

2.4.1 Reasoning about Functional Dependencies

A functional dependency from X to Y exists if and only if:

- For every instance of |R| of R, if two tuples in |R| agree on the values of the attributes in X, then they agree on the values of the attributes in Y

We write X → Y and say that X determines Y

Example: From {sid} → {first_name} and {sid} →{last_name} We can infer {sid} → {first_name, last_name}

Armstrong's Axioms

Be X, Y, Z be subset of the relation scheme of a relation R

Reflexivity:

If Y⊆X, then X→Y (trivial FDs)

- {name, supervisor_id}→{name}

Augmentation:

If X→Y , then X∪Z→Y∪Z

- if {supervisor_id} →{spesialization} , then {supervisor_id,name}→{spesialization,name}

Transitivity:

If X→Y and Y→Z, then X→Z

- if {supervisor_id} →{spesialization} and {spesialization} →{lab}, then {supervisor_id}→{lab}

Properties of Armstrong's Axioms

Armstrong's axioms are sound (i.e., correct) and complete (i.e., they can produce all possible FDs)

Example: Transitivity

Let X, Y, Z be subsets of the relation R

If X→Y and Y→Z, then X→Z

Proof of soundness:

Assume two tuples T1 and T2 of |R| are such that, for all attributes in X, T1.X = T2.X,

- since X→Y then, T1.Y = T2.Y
- since Y→Z and T1.Y = T2.Y then T1.Z = T2.Z

Additional Rules based on Armstrong's axioms

Armstrong's axioms can be used to produce additional rules that are not basic, but useful:

Weak Augmentation rule: Let X, Y, Z be subsets of the relation R If X→Y,then X∪Z→Y

Proof of soundness for Weak Augmentation

If X→Y

(1) Then by Augmentation X∪Z→Y∪Z

(2) And by Reflexivity Y∪Z →Y because Y ⊂ Y∪Z

(3) Then by Transitivity of (1) and (2) we have X∪Z → Y

Other useful rules:

If X → Y and X → Z, then X → Y∪Z **(union)**

If X → Y∪Z, then X → Y and X → Z **(decomposition)**

If X → Y and ZY → W, then ZX → W **(pseudotransitivity)**

2.4.2 Closure of a Set of Functional Dependencies

For a set F of functional dependencies, we call the closure of F, noted F+, the set of all the functional dependencies that can be derived from F (by the application of Armstrong's axioms).

- Intuitively, F+ is equivalent to F, but it contains some additional FDs that are only implicit in F.

Consider the relation scheme R(A,B,C,D) with

F = {{A} →{B},{B,C} →{D}}

F+ = { {A} →{A}, {B}→{B}, {C}→{C}, {D}→{D}, {A,B}→{A,B}, […],

{A}→{B}, {A,B}→{B}, {A,D}→{B,D}, **{A,C}→{B,C}**, {A,C,D}→{B,C,D}, {A} →{A,B}, {A,D}→{A,B,D}, {A,C}→{A,B,C}, {A,C,D}→{A,B,C,D}, **{B,C} →{D}**, […], **{A,C} →{D}**, […]}

2.4.3 Closure of a Set of Attributes

For a set X of attributes, we call the **closure** of X *(with respect to a set of functional dependencies F)*, noted X+, the maximum set of attributes such that X→X+ *(as a consequence of F)*

Consider the relation scheme R(A,B,C,D) with functional dependencies {A}→{C} and {B}→{D}.

- {A}+ = {A,C}
- {B}+ = {B,D}
- {C}+={C}
- {D}+={D}
- {A,B}+ = {A,B,C,D}

{A,B} is a superkey because:

- It determines all attributes
- Is minimal - neither {A} nor {B} alone are candidate keys

Algorithm for Computing the Closure of a Set of Attributes

Input:

- R a relation scheme
- F a set of functional dependencies
- $X \subset R$ (the set of attributes for which we want to compute the closure)

Output:

- X+ the closure of X w.r.t. F

$X^{(0)} := X$

Repeat $X^{(i+1)} := X^{(i)} \cup Z$, where Z is the set of attributes such that there exists $Y \rightarrow Z$ in F, and $Y \subset X^{(i)}$

Until $X^{(i+1)} := X^{(i)}$

Return $X^{(i+1)}$

Closure of a Set of Attributes: Example

R = {A,B,C,D,E,G}

F = { {A,B}→{C}, {C}→{A}, {B,C}→{D}, {A,C,D}→{B}, {D}→{E,G}, {B,E}→{C}, {C,G}→{B,D}, {C,E}→{A,G}}

X = {B,D}

$X^{(0)}$ = {B,D}

- {D}→{E,G},

$X^{(1)}$ = {B,D,E,G},

- {B,E}→{C}

$X^{(2)}$ = {B,C,D,E,G},

- {C,E}→{A,G}

$X^{(3)}$ = {A,B,C,D,E,G}

$X^{(4)} = X^{(3)}$

Redundancy of FDs

Sets of functional dependencies may have redundant dependencies that can be inferred from the

others

- {A}→{C} is redundant in: {{A}→{B}, {B}→{C},{A}→ {C}}

Parts of a functional dependency may be redundant

- Example of extraneous/redundant attribute on RHS:

{{A}→{B}, {B}→{C}, {A}→{C,D}} can be simplified to

{{A}→{B}, {B}→{C}, {A}→{D}}

(because {A}→{C} is inferred from {A} → {B}, {B}→{C})

- Example of extraneous/redundant attribute on LHS:

{{A}→{B}, {B}→{C}, {A,C}→{D}} can be simplified to

{{A}→{B}, {B}→{C}, {A}→{D}}

(because of {A}→{C})

Canonical Cover

A *canonical cover* for *F* is a set of dependencies F_c such that

- *F* and F_c, are equivalent
- F_c contains no redundancy
- Each left side of functional dependency in F_c is unique.
- For instance, if we have two FD X→Y, X→Z, we convert them to X→Y∪Z.

Algorithm for canonical cover of *F*:

repeat

Use the union rule to replace any dependencies in *F*

$X_1 \rightarrow Y_1$ and $X_1 \rightarrow Y_2$ with $X_1 \rightarrow Y_1 Y_2$

Find a functional dependency $X \rightarrow Y$ with an

extraneous attribute either in X or in Y

If an extraneous attribute is found, delete it from $X \rightarrow Y$

until *F* does not change

Note: Union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied

Example of Computing a Canonical Cover

$R = (A, B, C)$

$F = \{A \rightarrow BC$

$B \rightarrow C$

$A \rightarrow B$

$AB \rightarrow C\}$

Combine $A \rightarrow BC$ and $A \rightarrow B$ into $A \rightarrow BC$

- Set is now $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$

A is extraneous in $AB \rightarrow C$ because of $B \rightarrow C$.

- Set is now $\{A \rightarrow BC, B \rightarrow C\}$

C is extraneous in $A \rightarrow BC$ because of $A \rightarrow B$ and $B \rightarrow$ C.

The canonical cover is:

$A \rightarrow B, \qquad B \rightarrow C$

Possible Questions

Part-A

(Online – Multiple choice questions)

(Each questions carries one mark each)

PART – B (2 Marks)

1. Define set and operations on its.
2. Define key and give its types.
3. Define functional dependency
4. Write a short note on integrity rules.

PART – C (8 Marks)

1. Explain in detail about various types of keys with example.
2. What is canonical cover? With a neat algorithm and example sketch its computation
3. List the basic operation used in relational algebra. Explain them in detail with examples for each.
4. Discuss in detail about the various DML with examples for each commands.
5. Define a set. Describe the operations performed on a Set in detail with examples.
6. Discuss the two types of integrity rules that define the set of consistent database states
7. Write short notes on
   i) Attributes and Domains ii) Tuples iii) Relations and their schemes
8. Explain in detail about the set of Functional Dependencies FD
9. What terms describe a Relational model? Sketch it with a neat example.
10. On what circumstance a functional dependency occurs in a database. Explain in detail with examples.

Part – A - Included in Excel file – **File name** Unit-II(MCQ).xls

- Objective type / Multiple choice questions. Each question carries one mark
- It is for online examination

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

Subject: Relational Database Management System Sub.code: 17ITU303

S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	There are _____ basic components in relational model	1	2	3	4	**3**
2	There are _____ types of Integrity Rules	1	2	3	4	**2**
3	_____ key is used to implement Entity integrity	Primary key	Foreign key	Candidate key	Composite key	**Primary key**
4	Which key is used to implement Referential integrity	Primary key	Foreign key	Candidate key	Composite key	**Foreign key**
5	A _____ is a well defined collection of objects.	Entity	Relation	Table	Set	**Set**
6	Which symbol is used to represent Selection Operation	σ	π	X	÷	**σ**
7	For attributes X and Y in a functional dependency, "X determines Y" is represented as	X -> Y	X È Y	X Ç Y	X – Y	**X -> Y**
8	_____ operation is both associative and commutative	Difference	Union	Selection	Projection	**Union**
9	A minimal set of attributes that determines the entire tuple is _____	Super key	Foreign key	Candidate key	Primary key	**Candidate key**
10	Graphical representation of an entity is called	Data Flow Diagram	Structure Diagram	Use case Diagram	ER Diagram	**ER Diagram**
11	_____ is a single instance in an entity collection	Entity	Entity set	Entity instance	Entity collection	**Entity instance**

12	____ attributes cannot be further divided into smaller components	Composite	Simple	Derived	Stored	**Simple**
13	Which attribute have more than one value for an entity instance	Derived	Simple	Composite	Multi-valued	**Multi-valued**
14	Derived attributes are depicted in the E-R diagram with	double-lined ellipse	single-lined ellipse	dotted line	dashed ellipse	**dashed ellipse**
15	___ is a single attribute or a combination of attributes that uniquely identify an individual instance of an entity type	Composite	Simple	Key	Derived	**Key**
16	What symbol is used to represent multi-valued attribute	double-lined ellipse	single-lined ellipse	dotted line	thick line	**double-lined ellipse**
17	The elements of a set is also called	attribute	enitity	relationship	member	**member**
18	_____is defined in terms of ordered pairs or 2-tuples	Union	Intersection	Cartesian Product	Minus	**Cartesian Product**
19	The set of allowable values for the attribute is called	Entity	Domain	Table	Degree	**Domain**
20	Atomic domains are also referred t	application-independent domains	System dependent domain	application dependent domains	System independent domains	**application-independent domains**
21	t [Name, Age] is an example for	Selection	Intersection	Union	Projection	**Projection**
22	A relation scheme can be represent	$R(A_1, \ldots, A_{n)}$	$R(t_1, \ldots, t_{n)}$	$R(D_1, \ldots, D_{n)}$	$R(S_1, \ldots, S_{n)}$	$\mathbf{R(A_1, \ldots, A_{n)}}$
23	A relation with degree n is called as	Unary	Binary	Ternary	n - ary	**n - ary**
24	An attribute that forms a part of a candidate key of a relation is called a	Prime Attribute	Secondary Attribute	Independent Attribute	Dependent Attribute	**Prime Attribute**
25	For which basic operation the relations need not be union compatible	Union	Cartesian Product	Intersection	Minus	**Cartesian Product**

26	For X ® Y, if Y is a subset of X it is called as	Functional Dependency	Transitive Functional Dependency	Trivial Functional Dependency	Augmented Functional Dependency	**Trivial Functional Dependency**
27	If X®Y and Y®Z, then X®Z. this property is called as	Reflexivity	Augmentation	Pseudotransitivity	Transitivity	**Transitivity**
28	If X ® YÈZ, then X ® Y and X ® Z. this property is called as	Decomposition	Reflexivity	Augmentation	Pseudotransitivity	**Decomposition**
29	Closure of F is denoted as	F*	F+	F-	Fc	**F+**
30	A canonical cover for F is denoted	F*	F+	F-	Fc	**Fc**
31	Canonical Cover Fc of F must be such that	F and F_c are redundant	F and F_c are equivalent	F and F_c are union compatible	F and F_c are refelxive	**F and F_c are equivalent**
32	Closure of set of attributes X is denoted by	X*	X-	X+	Xc	**X+**
33	_____ represents an ordered pair of two tuples	Selection	Projection	Cartesian Product	Division	**Cartesian Product**
34	_____ operation selects common tuples from two relations	Intersection	Projection	Cartesian Product	Union	**Intersection**
35	_____ operation removes common tuples from first relations	Intersection	Difference	Cartesian Product	Union	**Difference**
36	____ operation eliminates duplicate values from the result set	Intersection	Difference	Cartesian Product	Union	**Union**
37	In relational model relation with its attributes are represented as____	R(A1, . . . ,An)	R(A1xA2x…..An)	R(A1:A2: . . . :An)	R(A1-An)	**R(A1, . . . ,An)**
38	A null set is represented as ____	{ }	{ φ }	{0}	{$}	**{ φ }**
39	If the degree of a relation is two it is called	Unary	Binary	Ternary	n -ary	**Binary**

40	What symbol is used to represent Intersection operation	È	Ç	X	σ	**Ç**
41	A relation is represented conceptually as a	Matrix	File	Table	Record	**Table**
42	Which one of the following can be a primary key for Employee Database	Address	Age	Date of Birth	Employee ID	**Employee ID**
43	The name for a relationship must always be	a noun	a verb	an adjective	a preposition	**a verb**
44	The name for an entity must always be	a noun	a verb	an adjective	a preposition	**a noun**
45	What symbol is used to represent cartesian product	È	Ç	X	σ	**X**
46	Combining more than one relation is called _____	Decomposition	Union	Join	Intersection	**Join**
47	Selection Operation is used to _____from a relation	Select the columns	Select the rows	Select the table	none.	**Select the rows**
48	Projection Operation is used to _____	Extract the columns	Extract rows	Select the table	none.	**Extract the columns**
49	Two relations have the same number of fields is called_____	Compatible	Union Compatible	Domain	attributes	**Union Compatible**
50	_____is used to combine information from two or more relations.	Concatenation	Combining	Joins	selection	**Joins**
51	Relational Algebra is a _____language	Procedural	non-procedural	declarative	CQL	**Procedural**
52	Selection,Projection,Rename operations are_____	Binary Operations	Cartesian product operations	Unary Operations	joining operation	**Unary Operations**
53	Joins,Set operations, Division operation are_____	Binary Operations	Cartesian product operations	Unary Operations	joining operation	**Binary Operations**
54	Relational calculus is a _____language	Procedural	non-procedural	operator-oriented	CQL	**non-procedural**

55	What symbol is used to represent Union operation	È	Ç	X	σ	**È**
56	An n-tuple can be specified as	T= (t_1 …..,t_n)	T= (d_1 …..,d_n)	T= (r_1 …..,r_n)	T= (a_1 …..,a_n)	**T= (a_1 …..,a_n)**
57	Every relation must have a	Primary key	Foreign key	Reference key	Composite key	**Primary key**
58	The operations and rules for a relation is defined by	Kevin	E.F.Codd	Gerald	George	**E.F.Codd**
59	There are _____ basic operations in relational algebra	1	3	4	2	**4**
60	What symbol is used to represent Intersection operation	Ç	Ç	X	σ	**Ç**

UNIT-III
SYLLABUS

Relational database manipulation: Introduction – SQL: Data definition – Data manipulation: Basic data retrieval – condition specification – Arithmetic and aggregate operations. SQL joins – set manipulation – categorization – updates – views – index. Data Control language : grant, revoke – simple privileges. .

Structured Query Language (SQL)

Structured Query language originated with the System R project in 1974 at IBM's San Jose Research Center. The purpose of this project was to validate the feasibility of the relation model and to implement a DBMS based on this model. The results of this project are well documented in the database literature. The system R project, concluded in 1979, was followed by the release of a number of commercial relational DBMS products from IBM. The first of these was SQL/DS for IBM's mid-range computers. Subsequently, DB2 was released for IBM's mainframe systems. SQL (the original version was called SEQUEL and a predecessor of SEQUEL was named SQUARE) was the data definition and manipulation language for System R. SQL has emerged as the standard query language for relational DBMSs. SQL is both the data definition and data manipulation language of a number of relational database systems. SQL is based on tuple calculus, though not as closely as QUEL. SQL resembles relational algebra in some places and tuple calculus in others.

3.1 Objects

3.1.1 Tables

Creation of tables

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows. Data definition in SQL is via the create statement. The statement can be used to create a table, index, or view (i.e., a virtual table based on existing tables). To create a table, the create statement specifies the name of the table and the names and data types of each column of the table.

Syntax

CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
);

The definition of an existing relation can be altered by using the **alter** statement. This statement allows a new column to be added to an existing relation. The existing tuples of the altered relation are logically considered to be assigned the null value for the added column.

Syntax

alter table existing-table-name
add column-name data-type[,…]

alter table EMPLOYEE
add phone_number decimal(10)

The **create index** statement allows the creation of an index for an already existing relation. The columns to be used in the generation of the index are also specified. The index is named and the ordering for each column used in the index can be specified as either ascending or descending. The **unique** option specifies that only one record could exist at any time with a given value for the column(s) specified in the statement to create the index.

Syntax

Create[unique]index name-of-index
On existing-table-name
(column-name[ascending or descending]

[,column-name[order]…])

Example

Create index empindex

On EMPLOYEE(Name **asc,** Pay_Rate **desc**);

Updating values of a table

Example

SQL> update p2 set price='320' where price='167';

1 row updated.

SQL> update p3 set publisher='tata' where title='cprog';

1 row updated.

Delete records from a table

Example

SQL> delete from p2 where subject='java';

1 row deleted.

3.1.2 Views

Creation of View

A view is a virtual table. In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

Syntax

CREATE VIEW view_name AS SELECT column_name(s)FROM table_name WHERE condition

Example:

CREATE VIEW View_Cust AS SELECT *FROM Customer_Details WHERE CUST_ID IN (101,102,103);

SQL> create view p2 as select subject, price from viewtable;

View created.

SQL> create view p3 as select *from viewtable;

View created.

Insert Statement

Example:

insert into view_cust values(103, 'Langer', 'G.','Justin', 3421, 'Savings', 'Global Commerce Bank', 'Langer_Justin@Yahoo.com');

SQL> insert into p3 values('5','java','kumar','tcs','java','123');

1 row created.

Delete Statement

Example:

delete view_cust where cust_id=103;

Update Statement

Example

Update view_cust set Cust_last_name='Smyth' where cust_id=101;

Drop Statement

You can delete a view with the DROP VIEW command.

Syntax

DROP VIEW view_name

Views with Group By clause

• The query contains a group by clause

Examples

CREATE VIEW View_GroupBY(Dept,NoofEmp) AS SELECT Department, count(Employee_ID)FROM Employee_ManagerGROUP BY Department

3.1.3 Indexes

An index can be created in a table to find data more quickly and efficiently. The users cannot see the indexes, they are just used to speed up searches/queries.

Note: Updating a table with indexes takes more time than updating a table without (because the indexes also need an update). So you should only create indexes on columns (and tables) that will be frequently searched against.

Creation of Index:

It creates an index on a table. Duplicate values are allowed:

Syntax :

CREATE INDEX index_name ON table_name (column_name)

Creation of Unique Index:

It creates a unique index on a table. Duplicate values are not allowed:

Syntax:

CREATE UNIQUE INDEX index_name ON table_name (column_name)

The DROP INDEX Statement

The DROP INDEX statement is used to delete an index in a table.

Syntax:

DROP INDEX index_name

3.1.4 Sequence

Creation of Sequence:

Syntax:

CREATE SEQUENCE seq_person MINVALUE 1 START WITH 1 INCREMENT

BY 1 CACHE 10

The code above creates a sequence object called seq_person that starts with 1 and will increment by 1. It will also cache up to 10 values for performance. The cache option specifies how many sequence values will be stored in memory for faster access. To insert a new record into the "Persons" table, we will have to use the nextval function This function retrieves the next value from seq_person sequence

INSERT INTO Persons (P_Id,FirstName,LastName)
VALUES (seq_person.nextval,'Lars','Monsen')

The SQL statement above would insert a new record into the "Persons" table. The "P_Id" column would be assigned the next number from the seq_person sequence. The "FirstName" column would be set to "Lars" and the "LastName" column would be set to "Monsen".

3.1.5 Synonym

A synonym is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects.

Creating or replacing a synonym

Syntax

create [or replace] [public] synonym [schema .] synonym_name for [schema .] object_name [@ dblink];

- The “or replace” phrase allows you to recreate the synonym (if it already exists) without having to issue a DROP synonym command.
- The “public” phrase means that the synonym is a public synonym and is accessible to all users. Remember though that the user must first have the appropriate privileges to the object to use the synonym.

- The "schema" phrase is the appropriate schema. If this phrase is omitted, Oracle assumes that you are referring to your own schema.
- The "object_name" phrase is the name of the object for which you are creating the synonym. It can be one of the following:

table	**package**
view	**materialized view**
sequence	**java class schema object**
stored procedure	**user-defined object**
function	**synonym**

Example:

create public synonym suppliers for app.suppliers;

This first example demonstrates how to create a synonym called *suppliers*. Now, users of other schemas can reference the table called *suppliers* without having to prefix the table name with the schema named *app*. For example:

select * from suppliers;

If this synonym already exists and we need to redefine it, we can always use the "or replace" phrase as follows:

create or replace public synonym suppliers for app.suppliers;

Dropping a synonym

It is also possible to drop a synonym.

Syntax:

drop [public] synonym [schema .] synonym_name [force];

The public phrase allows you to drop a public synonym. If you have specified public, then you don't specify a schema. The force phrase will force Oracle to drop the synonym even if it has dependencies. It is probably not a good idea to use the force phrase as it can cause invalidation of Oracle objects**.**

Example:

drop public synonym suppliers;

This drop statement would drop the synonym called *suppliers* that we defined earlier.

3.2 Data types

CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data

LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice
date	Store a date only. From January 1, 0001 to December 31, 9999
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647

3.3 SQL Constraints

Constraints are used to limit the type of data that can go into a table. Constraints can be specified when a table is created (with the CREATE TABLE statement) or after the table is created (with the ALTER TABLE statement). The focus will be on the following constraints:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

SQL NOT NULL Constraint

The NOT NULL constraint enforces a column to NOT accept NULL values. The NOT NULL constraint enforces a field to always contain a value. This means that you cannot insert a new record, or update a record without adding a value to this field. The following SQL enforces the "P_Id" column and the "LastName" column to not accept NULL values:

CREATE TABLE Persons (P_Id int NOT NULL,LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255),City varchar(255))

SQL UNIQUE Constraint

The UNIQUE constraint uniquely identifies each record in a database table. The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns. A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it. Note that you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table. The following SQL creates a UNIQUE constraint on the "P_Id" column when the "Persons" table is created

CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255),City varchar(255),UNIQUE (P_Id))

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain unique values. A primary key column cannot contain NULL values. Each table should have a primary key, and each table can have only ONE primary key. The following SQL creates a PRIMARY KEY on the "P_Id" column when the "Persons" table is created:

CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255),City varchar(255),PRIMARY KEY (P_Id))

SQL FOREIGN KEY Constraint

A FOREIGN KEY in one table points to a PRIMARY KEY in another table. Let's illustrate the foreign key with an example. Look at the following two tables:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Note that the "P_Id" column in the "Orders" table points to the "P_Id" column in the "Persons" table. The "P_Id" column in the "Persons" table is the PRIMARY KEY in the "Persons" table. The "P_Id" column in the "Orders" table is a FOREIGN KEY in the "Orders" table. The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables. The FOREIGN KEY constraint also prevents that invalid data form being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

Examples

The following SQL creates a FOREIGN KEY on the "P_Id" column when the "Orders" table is created:

CREATE TABLE Orders (O_Id int NOT NULL, OrderNo int NOT NULL,P_Id int,PRIMARY KEY (O_Id), FOREIGN KEY (P_Id) REFERENCES Persons(P_Id))

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column. If you define a CHECK constraint on a single column it allows only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on

values in other columns in the row. The following SQL creates a CHECK constraint on the "P_Id" column when the "Persons" table is created. The CHECK constraint specifies that the column "P_Id" must only include integers greater than 0.

Syntax

CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL,FirstName varchar(255), Address varchar(255),City varchar(255),CHECK (P_Id>0))

SQL DEFAULT Constraint

The DEFAULT constraint is used to insert a default value into a column. The default value will be added to all new records, if no other value is specified. The following SQL creates a DEFAULT constraint on the "City" column when the "Persons" table is created:

Syntax

CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255),City varchar(255) DEFAULT 'Sandnes')

3.4 E.F.CODD RULES

E.F. Codd, the famous mathematician has introduced 12 rules for the relational model for databases commonly known as Codd's rules. The rules mainly define what is required for a DBMS for it to be considered relational, i.e., an RDBMS. There is also one more rule i.e. Rule00 which specifies the relational model should use the relational way to manage the database. The rules and their description are as follows:-

Rule 000: A RDBMS system should be capable of using its relational facilities (exclusively) to manage the database.

Rule 1: The information rule : All information in the database is to be represented in one and only one way. This is achieved by values in column positions within rows of tables.

Rule 2 : The guaranteed access rule : All data must be accessible with no ambiguity. This is achieved in the RDBMS by using the primary key concept.

Rule 3: Systematic treatment of null values : The DBMS must allow each field to remain null. The null can be stored in any field of any datatype.

Rule 4: Active online catalog based on the relational model : The authorized users can access the database structure by using common language i.e. SQL.

Rule 5: The comprehensive data sublanguage rule : The system must support at least one relational language that has simple syntax and transaction management facilities. It can be used in the application as well as in the RDBMS systems.

Rule 6: The view updating rule : All views must be updatable by the system.

Rule 7: High-level insert, update, and delete : The system is able to insert, update and delete operations fully. It can also perform the operations on multiple rows simultaneously.

Rule 8: Physical data independence : Changes to the physical storage structure must not require a change to an application based on the structure.

Rule 9: Logical data independence : Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure.

Rule 10: Integrity independence: All the Integrity constraints like primary key, unique key etc must be specified separately from application programs and stored in the catalog.

Rule 11: Distribution independence: The distribution of portions of the database to various locations should be invisible to users of the database.

Rule 12: The no subversion rule : If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.

Note:- Any database management system which fulfills 6 or more than 6 rules can be considered as the RDBMS.

3.5 Data Definition Language

- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

Creation of Table

Syntax

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
)
```

Alter Table

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table

Syntax

To add a column in a table, use the following syntax:

ALTER TABLE table_name ADD column_name datatype

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

ALTER TABLE table_name DROP COLUMN column_name

To change the data type of a column in a table, use the following syntax:

ALTER TABLE table_name ALTER COLUMN column_name datatype

SQL ALTER TABLE Example

Look at the "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to add a column named "DateOfBirth" in the "Persons" table. We use the following SQL statement:

ALTER TABLE Persons ADD DateOfBirth date

The "Persons" table will now like this:

P_Id	LastName	FirstName	Address	City	DateOfBirth
1	Hansen	Ola	Timoteivn 10	Sandnes	
2	Svendson	Tove	Borgvn 23	Sandnes	
3	Pettersen	Kari	Storgt 20	Stavanger	

Change Data Type Example

Now we want to change the data type of the column named "DateOfBirth" in the "Persons" table. We use the following SQL statement:

ALTER TABLE Persons ALTER COLUMN DateOfBirth year

Notice that the "DateOfBirth" column is now of type year and is going to hold a year in a two-digit or four-digit format.

Drop Column

If we want to delete the column named "DateOfBirth" in the "Persons" table. We use the following SQL statement:

ALTER TABLE Persons DROP COLUMN DateOfBirth

The "Persons" table will now like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Drop Table

The DROP TABLE statement is used to delete a table.

Syntax:

DROP TABLE table_name

3.6 Data Manipulation Language

- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database

The UPDATE Statement

The UPDATE statement is used to update existing records in a table.

Syntax:

UPDATE table_name SET column1=value, column2=value2,... WHERE some_column=some_value

The DELETE Statement

The DELETE statement is used to delete rows in a table.

Syntax

DELETE FROM table_name WHERE some_column=some_value

SQL DELETE Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

Now we want to delete the person "Tjessem, Jakob" in the "Persons" table. We use the following SQL statement:

DELETE FROM Persons WHERE LastName='Tjessem' AND FirstName='Jakob'

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

Delete All Rows

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM table_name;

(or)

DELETE * FROM table_name;

The INSERT INTO Statement

The INSERT INTO statement is used to insert a new row in a table.

Syntax

It is possible to write the INSERT INTO statement in two forms. The first form doesn't specify the column names where the data will be inserted, only their values:

INSERT INTO table_name VALUES (value1, value2, value3,...)

The second form specifies both the column names and the values to be inserted:

INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)

Example

We have the following "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to insert a new row in the "Persons" table. We use the following SQL statement:

INSERT INTO Persons VALUES (4,'Nilsen', 'Johan', 'Bakken 2', 'Stavanger')

Insert Data Only in Specified Columns

It is also possible to only add data in specific columns. The following SQL statement will add a new row, but only add data in the "P_Id", "LastName" and the "FirstName" columns:

INSERT INTO Persons (P_Id, LastName, FirstName)VALUES (5, 'Tjessem', 'Jakob')

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

SQL - Using GROUP BY

Related rows can be grouped together by **GROUP BY** clause by specifying a column as a grouping column. **GROUP BY** is associated with an aggregate function

To retrieve the total loan-amount of all loans taken by each Customer

Examples

SELECT Cust_ID, SUM(Amount_in_Dollars)FROM Customer_Loan GROUP BY Cust_ID;

Retrieval using HAVING

Used to specify condition on a group

Syntax

SELECT column_name, aggregate_function(column_name)FROM table_name WHERE column_name operator value GROUP BY column_name HAVING aggregate_function(column_name) operator value

List all customers who are having loans greater than 4000

Select Cust_ID,SUM(Amount_in_Dollars)From Customer_Loan Group By Cust_ID Having SUM(Amount_in_Dollars) > 4000.00;

ORDER BY

The ORDER BY keyword is used to sort the result-set by a specified column. The ORDER BY keyword sort the records in ascending order by default. If you want to sort the records in a descending order, you can use the DESC keyword.

Syntax

SELECT column_name(s) FROM table_name ORDER BY column_name(s) ASC/DESC

Example

SELECT * FROM Persons ORDER BY LastName

To select all the persons from the table above, however, we want to sort the persons descending by their last name, We use the following SELECT statement:

Example

SELECT * FROM Persons ORDER BY LastName DESC

3.7 Data Query Language

SELECT - extracts data from a database

The SQL SELECT Statement

The SELECT statement is used to select data from a database. The result is stored in a result table, called the result-set.

Syntax

SELECT column_name(s) FROM table_name

(or)

SELECT * FROM table_name

3.8 Subqueries

Sub query is defined as a query within a query. The inner query will be executed first. The outer and inner queries are connected by any one of the following operators.

The general form is

Outerquery operator(inner query);

1) ANY

Eg. Select * from salesmaster where quantity < any (select stock from itermaster where costprice>150);

When this query is executed, it selects all the stock value from itemmaster whose costprice is grater than 150.

2) IN

Eg. Select * from department where deptid IN (select deptid from employee where salary>10000);

When this query is executed it selects all the deptid column from employe table whose salary value is greater than 10000. then it selects all the rows from department table whose deptid column is in the already selected deptid from employee table.

3) NOT IN

Eg. Select * from department where deptid NOT IN (select deptid from employee where salary>10000);

When this query is executed, it selects all the deptid column from employee table whose salary value is greater than 10000. then it selects all the rows from department table where deptid column is not in the already selected deptid from employee table.

3.9 Operators

An **operator** manipulates data items and returns a result. Syntactically, an operator appears before or after an operand or between two operands.

Arithmetic Operators

Arithmetic operator can be used with one or two arguments to negate, add, subtract, multiply, and divide numeric values. Some of these operators are also used in datetime and interval arithmetic. The arguments to the operator must resolve to numeric datatypes or to any datatype that can be implicitly converted to a numeric datatype. Unary arithmetic operators return the same datatype as the numeric datatype of the argument. For binary arithmetic operators, Oracle determines the argument with the highest numeric precedence, implicitly converts the remaining arguments to that datatype, and returns that datatype.

Operator	Purpose	Example
+ -	When these denote a positive or negative expression, they are unary operators.	SELECT * FROM order_items WHERE quantity = -1; SELECT * FROM employees WHERE -salary < 0;
+ -	When they add or subtract, they are binary operators.	SELECT hire_date FROM employees WHERE SYSDATE - hire_date > 365;
* /	Multiply, divide. These are binary operators.	UPDATE employees SET salary = salary * 1.1;

Do not use two consecutive minus signs (--) in arithmetic expressions to indicate double negation or the subtraction of a negative value. The characters -- are used to begin comments within SQL statements. You should separate consecutive minus signs with a space or parentheses.

Boolean Operator

Used in WHERE clause to join two conditions. And: returns results only when all conditions are true. OR: returns results when any conditions are true. NOT: negates an expression. Order of the operators: NOT first, then AND, then OR. Use parenthesis to change default order of operators

Examples:

Select Product_Description, Product_Finish, Standard_Price
from Product_T Where (Product_Description Like '%Desk'
R Product_Description Like '%Table') And Unit_Price > 300;

3.10 Set Operators

1. **Intersect**
2. **Minus**

3. **Union**
4. **Union all**

3.10.1 Intersect

List all the customer who have both Fixed Deposit and Loan.

Example

SELECT Cust_ID FROM Customer_Fixed_Deposit INTERSECT SELECT Cust_ID FROM Customer_Loan;

3.10.2 Minus

Get All the Customer who have not taken loan

Example

Select Cust_ID from Customer_details MINUS Select Cust_Idfrom Customer_loan;

3.10.3 Union

The UNION operation combines the rows from two sets of query results. By default, the UNION operation eliminates duplicate rows

Example

SELECT Cust_ID FROM Customer_Fixed_Deposit UNION SELECT Cust_ID FROM Customer_Loan;

3.10.4 UnionAll

The UNION ALL operation combines the rows from two sets of query results. It does not eliminate duplicate rows.

Example

SELECT Cust_ID FROM Customer_Fixed_Deposit UNION ALL SELECT Cust_ID FROM Customer_Loan;

3.11 SQL Joins

SQL Joins are used to relate information in different tables. A Join condition is a part of the sql query that retrieves rows from two or more tables. A SQL Join condition is used in the SQL WHERE Clause of select, update, delete statements.

The Syntax for joining two tables is:

```
SELECT col1, col2, col3...FROM table_name1, table_name2
WHERE table_name1.col2 = table_name2.col1;
```

If a sql join condition is omitted or if it is invalid the join operation will result in a Cartesian product. The Cartesian product returns a number of rows equal to the product of all rows in all the tables being joined. For example, if the first table has 20 rows and the second table has 10 rows, the result will be 20 * 10, or 200 rows. This query takes a long time to execute.

Lets use the below two tables to explain the sql join conditions.

database table "product";

product_id	product_name	supplier_name	unit_price
100	Camera	Nikon	300
101	Television	Onida	100
102	Refrigerator	Videocon	150
103	Ipod	Apple	75
104	Mobile	Nokia	50

database table "order_items";

order_id	product_id	total_units	customer
5100	104	30	Infosys
5101	102	5	Satyam

5102	103	25	Wipro
5103	101	10	TCS

SQL Joins can be classified into Equi join and Non Equi join.

1) SQL Equi joins

It is a simple sql join condition which uses the equal sign as the comparison operator. Two types of equi joins are SQL Outer join and SQL Inner join. For example: You can get the information about a customer who purchased a product and the quantity of product.

2) SQL Non equi joins

It is a sql join condition which makes use of some comparison operator other than the equal sign like >, <, >=, <=

1) SQL Equi Joins:

An equi-join is further classified into two categories:

a) SQL Inner Join

b) SQL Outer Join

a) SQL Inner Join:

All the rows returned by the sql query satisfy the sql join condition specified.

For example: If you want to display the product information for each order the query will be as given below. Since you are retrieving the data from two tables, you need to identify the common column between these two tables, which is the product_id.

The query for this type of sql joins would be like,

```
SELECT order_id, product_name, unit_price, supplier_name, total_units FROM
product, order_items WHERE order_items.product_id = product.product_id;
```

The columns must be referenced by the table name in the join condition, because product_id is a column in both the tables and needs a way to be identified. This avoids ambiguity in using the columns in the SQL SELECT statement.

The number of join conditions is (n-1), if there are more than two tables joined in a query where 'n' is the number of tables involved. The rule must be true to avoid Cartesian product.

We can also use aliases to reference the column name, then the above query would be like,

```
SELECT o.order_id, p.product_name, p.unit_price, p.supplier_name, o.total_units FROM product p, order_items o WHERE o.product_id = p.product_id;
```

b) SQL Outer Join:

This sql join condition returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables. The sql outer join operator in Oracle is (+) and is used on one side of the join condition only.

The syntax differs for different RDBMS implementation. Few of them represent the join conditions as "sql left outer join", "sql right outer join".

If you want to display all the product data along with order items data, with null values displayed for order items if a product has no order item, the sql query for outer join would be as shown below:

```
SELECT p.product_id, p.product_name, o.order_id, o.total_units FROM order_items o, product p WHERE o.product_id (+) = p.product_id;
```

The output would be like,

product_id	product_name	order_id	total_units
100	Camera		
101	Television	5103	10
102	Refrigerator	5101	5
103	Ipod	5102	25
104	Mobile	5100	30

SQL Self Join:

A Self Join is a type of sql join which is used to join a table to itself, particularly when the table has a FOREIGN KEY that references its own PRIMARY KEY. It is necessary to ensure that the join statement defines an alias for both copies of the table to avoid column ambiguity.

The below query is an example of a self join,

```
SELECT a.sales_person_id, a.name, a.manager_id, b.sales_person_id, b.name FROM  sales_person a, sales_person b WHERE a.manager_id = b.sales_person_id;
```

2) SQL Non Equi Join:

A Non Equi Join is a SQL Join whose condition is established using all comparison operators except the equal (=) operator. Like >=, <=, <, >

For example: If you want to find the names of students who are not studying either Economics, the sql query would be like,

```
SELECT first_name, last_name, subject FROM student_details WHERE subject != 'Economics'
```

The output would be something like,

first_name	last_name	subject
Anjali	Bhagwat	Maths
Shekar	Gowda	Maths
Rahul	Sharma	Science
Stephen	Fleming	Science

3.12 SQL - Aggregate functions

- Used when information you want to extract from a table has to do with the data in the entire table taken as a set.
- Aggregate functions are used in place of column names in the SELECT statement

- **The aggregate functions in sql are :**

MIN( ), MAX( ), AVG( ), SUM( ) , COUNT( )

Aggregate function – MIN

- Returns the smallest value that occurs in the specified column
- Column need not be numeric type

Syntax

SELECT MIN(column_name) FROM table_name

Example

SELECT MIN(OrderPrice) AS SmallestOrderPrice FROM Orders

Aggregate function – MAX

• Returns the largest value that occurs in the specified column

• Column need not be numeric type

Syntax

SELECT MAX(column_name) FROM table_name

Example

SELECT MAX(OrderPrice) AS LargestOrderPrice FROM Orders

Aggregate function – AVG

• Returns the average of all the values in the specified column

• Column must be numeric data type

Syntax

SELECT AVG(column_name) FROM table_name

Example

SELECT AVG(OrderPrice) AS OrderAverage FROM Orders

Aggregate function – SUM

• Adds up the values in the specified column

• Column must be numeric data type

• Value of the sum must be within the range of that data type

Syntax

SELECT SUM(column_name) FROM table_name

Example

SELECT SUM(OrderPrice) AS OrderTotal FROM Orders

Aggregate function – COUNT

• The COUNT() function returns the number of rows that matches a specified criteria.

Syntax

The COUNT(column_name) function returns the number of values (NULL values will not be counted) of the specified column:

SELECT COUNT(column_name) FROM table_name

SQL COUNT(*) Syntax

The COUNT(*) function returns the number of records in a table:

Syntax

SELECT COUNT(*) FROM table_name

SELECT COUNT(DISTINCT column_name) FROM table_name

This function returns the number of distinct values of the specified column:

3.13 Data Control Languages

Two statement

1. Grant statement
2. Revoke statement

Grant statement

Grant gives user's privileges to data base

The following types of privileges can be granted:

- Delete data from a specific table.
- Insert data into a specific table.
- Create a foreign key reference to the named table or to a subset of columns from a table.
- Select data from a table, view, or a subset of columns in a table.
- Create a trigger on a table.
- Update data in a table or in a subset of columns in a table.
- Run a specified function or procedure.

Syntax

GRANT privilege-type ON [TABLE] { table-Name | view-Name } TO grantees

To grant the SELECT privilege on table `t` to the authorization IDs `Maria` and `Harry`, use the following syntax:

GRANT SELECT ON TABLE t TO Maria, Harry

To grant the SELECT privilege on table `s.v` to all users, use the following syntax:

GRANT SELECT ON TABLE s.v to PUBLIC

Revoke statement

Withdraw access privileges given with the grant command

The following types of privileges can be revoked:

- Delete data from a specific table.
- Insert data into a specific table.
- Create a foreign key reference to the named table or to a subset of columns from a table.
- Select data from a table, view, or a subset of columns in a table.
- Create a trigger on a table.
- Update data in a table or in a subset of columns in a table.
- Run a specified routine (function or procedure).

Syntax

REVOKE privilege-type ON [TABLE] { table-Name | view-Name } FROM grantees

To revoke the SELECT privilege on table t from the authorization IDs Maria and Harry, use the following syntax:

REVOKE SELECT ON TABLE t FROM Maria, Harry

To revoke the SELECT privilege on table s.v from all users, use the following syntax:

REVOKE SELECT ON TABLE s.v FROM PUBLIC

Possible Questions

Part-A

(Online – Multiple choice questions)

(Each questions carries one mark each)

PART – B (2 Marks)

1. What is DML?
2. What is data definition language?
3. Write a short note on SQL join
4. Write a short note on indexes and lists its operation.
5. What is view?

PART – C (8 Marks)

1. Discuss in detail about the various DDL with examples for each commands.
2. What are sub-queries? Write queries to implement sub-queries for any two joining operator.
3. List the basic operation used in relational algebra. Explain them in detail with for each.
4. Discuss in detail about the various DML with examples for each commands
5. Write the steps with example to perform the following operations in a table
   a. i) Creating a table ii) Updating values
   b. iii) Deleting values iv) Modifying table structure
6. Discuss the various SET operators used in SQL with examples for each.
7. Explain in detail about operator to manipulate data items.
8. Illustrate with examples the use of GROUP BY, HAVING and ORDER BYclause in SQL with its various forms.
9. Enumerate in detail about various types of constraints used in SQL database. Give example queries representing each constraint.
10. What is the significance of creating a view for a table? Write the various Data definition and Data manipulation commands used in views

Part – A - Included in Excel file – **File name** Unit-III(MCQ).xls

- Objective type / Multiple choice questions. Each question carries one mark
- It is for online examination

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

Subject: Relational Database Management System Sub.code: 17ITU303

S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____is the collection of schemas for the relations in the database.	Physical Schema	Conceptual Schema	Logical Schema	relational database	**relational database**
2	_____statement is used to define a new table.	Create	Produce	Insert	Add	**Create**
3	Tuples are inserted using the _____command	Create	Insert	Add	Make	**Insert**
4	Tuples are deleted using the _____command	Delete	drop	remove	alter	**Delete**
5	Modify the column values in an existing row using_____ command	Modify	Alter	Update	Change	**Update**
6	_____clause is used to modify a particular row.	Update	Modify	Alter	Where	**Update**
7	_____is a condition specified on a database schema & restricts the data that can be stored in an instance of the database.	integrity Constraint	restriction	key	none	**integrity Constraint**
8	A set of fields that uniquely identifies a tuple according to a key constraint is called_____ for the relation	primary key	Candidate key	foreign key	Super key	**Candidate key**

9	_____is used to refer the primary key in another entity	Candidate key	referential key	foreign key	Primary key	**foreign key**
10	_____value is unknown or not applicable	not null	zero	unknown	null	**null**
11	_____is a specialized language for writing queries	Sub language	Object Oriented Language	Query language	Programming language	**Query language**
12	_____retain all fields of selected tuples in the result.	Star	*	X	%	*****
13	The set of non descriptive attributes is a _____for the relation	Sub key	Foreign key	Super Key	Referential key	**Super Key**
14	_____is a table whose rows are not explicitly stored in the database	View	Virtual	Temporary table	Physical Schema	**View**
15	_____is the collection of Schemas of the relation stored in the database.	Physical Schema	Conceptual Schema	Internal Schema	External Schema	**Conceptual Schema**
16	_____command is used to remove the table definition information	Delete	Remove	Destroy	Drop	**Drop**
17	_____is used to modify the structure of an existing table.	Modify	Alter	Change	Recreate	**Alter**
18	View can be dropped using _____command	Delete View	Remove View	Replace View	Drop View	**Drop View**
19	_____columns are not allowed to contain null values	Primary Key	Candidate key	foreign Key	Unique Key	**Primary Key**
20	_____are valuable to give the security to our original table	Original table	Duplicate table	Views	Tables	**Views**
21	_____ allows to update operation on views.	Updatable views	Modifying views	Altering views	Changeable views	**Updatable views**

22	Primary keys otherwise known as _____	Super key	Sub key	Minimal key	Foreign key	**Super key**
23	Create, Alter, Drop commands are_____ language commands	DML	TCL	DCL	DDL	**DDL**
24	Insert,Select,Update,delete commands are _____language commands	DML	TCL	DCL	DDL	**DML**
25	Queries in algebra are composed using a collection of _____	relations	tuples	attributes	operators	**operators**
26	_____clause is used to group the values under particular characteristics	Order By	Group by	Count	Sum	**Group by**
27	Group function is otherwise known as_____	Collection	Aggregate	function	Count	**Aggregate**
28	_____keyword is used to assign a default value with a domain.	Static	Default	Permanent	distinct	**Default**
29	Which constraint is used to check the ranges in the column values?	range	verify	check	condition	**check**
30	_____is used to calculate the number of values in the Column.	Count	aggregate	Cal	Calculate	**Count**
31	_____is used to calculate the sum of all values in the column	Total	Sum	Count	Collection	**Sum**
32	_____is used to calculate the average of all values in the column	Total	Sum	Average	avg	**avg**

33	Which function is used to extract the maximum value in the relations?	max	maximum	excess	large	**max**
34	Which function is used to extract the minimum value in the relations?	Small	minimum	lower	min	**min**
35	Which clause is used in conjuction with Group by clause to check a condition?	where	Having	Distinct	Group	**Having**
36	_____keyword is used to eliminates the duplicates	Union	Union all	Intersect all	Except all	**Union**
37	_____keyword is used to retain the duplicates	Union	Union all	Intersect	Except	**Union all**
38	_____is a query that has another query embedded within it.	Query	nested query	QBE	QUEL	**nested query**
39	Embedded query is called_____	host language	embedded language	subquery	QBE	**subquery**
40	_____ is an object used to speed up the searching process	Table	View	Sequence	Index	**Index**
41	_____ is called as a virtual table	Table	View	Sequence	Index	**View**
42	Duplicates are eliminated by using _____ keyword.	Remove	Distinct	RM	Redundant	**Distinct**
43	Which keyword is used to check if an element is present a given set?	not	in	not exist	except	**in**
44	Any two tables that are Union-Compatible that is, have the same number of _____	Columns	rows	Columns and rows	null values	**Columns**
45	_____is used when the column value is either unknown or inapplicable.	zero	all	Empty	null	**null**

46	___ are used to relate information in different tables	Join	Selection	Projection	Intersection	**Join**
47	_____ join condition makes use of comparison operators other than the equal sign	Inner join	Outer join	Equi join	Non equi join	**Non equi join**
48	The + symbol is used on one side o	Inner join	Outer join	Equi join	Non equi join	**Outer join**
49	___ is a type of sql join which is used to join a table to itself	Inner join	Outer join	Equi join	Self join	**Self join**
50	____ keyword is used to define an alias name	in	from	as	to	**as**
51	A _____ is the person who accepts the privileges provided to him	Granter	Grantee	Administrator	Revoker	**Grantee**
52	_____ keyword is used to provide a specific privilege to all the users of database.	Public	All	All Users	Public users	**Public**
53	____ the command used to withdraw the access privilege given already	Withdraw	Revoke	Remove	Extract	**Revoke**
54	____ the command used to provide access privilege to the user.	Withdraw	Revoke	Grant	Allow	**Grant**
55	How many forms of Insert statement is provided by SQL?	2	3	4	5	**3**
56	Indexes can be created on _____	Columns	Rows	Tuples	Triggers	**Columns**
57	_____ command is used to delete an index	Delete	Remove	Drop	Delete all	**Drop**
58	____ holds a string with a maximum length of 65,535 characters	LONGTEXT	MEDIUMTEXT	TINYTEXT	TEXT	**TEXT**

59	In _____ datatype if more than 255 characters are stored it will be converted to TEXT type	CHAR	VARCHAR	TINYTEXT	TEXT	**VARCHAR**
60	The value of a Date datatype ranges in	Jan 1, 1000 to December 31, 9999	Jan 1, 0001 to December 31, 1999	Jan 1, 0001 to December 31, 9999	Jan 1, 1000 to December 31, 1999	**Jan 1, 0001 to December 31, 9999**

UNIT-IV
SYLLABUS

Overview of PL/SQL Declaration section – executable command section : conditional logic, loops, CASE statements – exception handling section: predefined and user defined exceptions. Triggers : definition – types : row level, statement level, before and after, instead of – syntax – enabling and disabling triggers - replacing and dropping triggers. Cursors – definition – open – fetch – close – cursor attributes- select for update – types : implicit, explicit. Procedures, Functions: Local and global – procedures vs functions – stored procedures, functions – create procedure syntax - create function syntax – calling procedures, functions. Replacing and dropping procedures, functions.

PL/SQL

PL/SQL is Oracle's procedural language (PL) superset of the Structured Query Language

4.1 PL/SQL Overview

PL/SQL code is grouped into structures called blocks. If you create a stored procedure or package, you give the block of PL/SQL code a name; if the block of PL/SQL code is not given a name, then it is called an anonymous block. A block of PL/SQL code contains three sections:

Section	Description	inclusion
Declarative	Contains all variables, constants, cursors, and user-defined exceptions that are referenced in the executable and declarative sections	Optional
Executable	Contains SQL statements to manipulate data in the database and PL/SQL statements to manipulate data in the block	Mandatory
Exception handling	Specifies the actions to perform when errors and abnormal conditions arise in the executable section	Optional

The Declarations section starts with the keyword **declare** and ends when the Executable Commands section starts (as indicated by the keyword **begin**). The Executable Commands section is followed by the Exception Handling section; the **exception** keyword signals the start of the Exception Handling section. The PL/SQL block is terminated by the **end** keyword.

The structure of a typical PL/SQL block is shown in the following listing:

declare

<declarations section>

begin

<executable commands>

exception

<exception handling>

end;

- Any block may contain sub-block. Sub-blocks may appear anywhere an executable statement may legally appear.
- Statement end with a ;
- Comments are preceded by --or surrounded by /* */ Declared objects exist within a certain scope (addressed later in this course).

4.2 Declarations Section

The Declarations section begins a PL/SQL block. The Declarations section starts with the declare keyword, followed by a list of variable and cursor definitions. You can define variables to have constant values, and variables can inherit data types from existing columns and query results.

Types of Variables

All PL/SQL variables have a data type, which specifies a storage format, constraints, and valid range of values. PL/SQL supports four data type categories—scalar, composite, reference, and LOB (large object)— that you can use for declaring variables, constants, and pointers.

- Scalar data types hold a single value. The main data types are those that correspond to column types in Oracle server tables; PL/SQL also supports Boolean variables.
- Composite data types, such as records, allow groups of fields to be defined and manipulated in PL/SQL blocks.
- Reference data types hold values, called pointers that designate other program items.
- LOB data types hold values, called locators, that specify the location of large objects (for example graphic images) that are stored out of line.

- Non-PL/SQL variables include host language variables declared in precompiler programs, screen fields in Forms applications, and 1SQL*Plus host variables.

Data types

The PL/SQL variables, constants and parameters must have a valid data type, which specifies a storage format, constraints, and a valid range of values.

Numeric

Numeric values on which arithmetic operations are performed.

Character

Alphanumeric values that represent single characters or strings of characters

Boolean

Logical values on which logical operations are performed

Datetime Dates and times.

NUMBER(prec, scale)

Fixed-point or floating-point number with absolute value in range 1E-130 to (but not including) 1.0E126. A NUMBER variable can also represent 0

CHAR

Fixed-length character string with maximum size of 32,767 bytes

VARCHAR2

Variable-length character string with maximum size of 32,767 bytes

Example

```
DECLARE
  num1 INTEGER;
  num2 REAL;
  num3 DOUBLE PRECISION;
BEGIN
  null;
END;
/
```

The %TYPE and %ROWTYPE Attribute

The %TYPE is used attribute to declare a variable according to another previously declared

variable or database column. To use the attribute in place of the data type that is required in the variable declaration, prefix it with the database table and column name. if referring to a previously declared variable, prefix the variable name to the attribute. Using the %ROWTYPE declaration, the variable inherits the column and datatype information for all the columns in the cursor's result set.

Identifier Table.columnname%TYPE;

Example:

```
cursor rad_cursor is
select * from RADIUS_VALS;
rad_val rad_cursor%ROWTYPE;
rad_val_radius rad_val.Radius%TYPE;
```

Scoping Variables and Constants.

SCOPE refers to the visibility of identifiers at different points in the PL /SQL block.

1. An identifier is visible in the block in which it is declared and all its sub-blocks unless rule #2 applies.
2. If an identifier in an enclosing block is redeclared in a sub-block, the original identifier declared in the enclosing block is no longer visible in the sub-block .However, the newly declared identifier has the rules of scope defined in rule #1.

4.3 Executable Commands Section

In the Executable Commands section, you manipulate the variables and cursors declared in the Declarations section of your PL/SQL block. The Executable Commands section always starts with the keyword **begin**

Example:

```
declare
pi constant NUMBER(9,7) := 3.1415927;
radius INTEGER(5);
area NUMBER(14,2);
begin
radius := 3;
```

```
area := pi*power(radius,2);
insert into AREAS values (radius, area);
end;
```

The Executable Commands section may contain conditional logic, such as **if** commands and loops

4.4 Conditional Logic

Within PL/SQL, you can use **if**, **else**, **elsif**, and **case** commands to control the flow of commands within the Executable Commands section. The formats of the **if** clauses are shown in the following listing:

```
if <some condition>
then <some command>
elsif <some condition>
then <some command>
else <some command>
end if;
```

You can nest **if** conditions within each other, as shown in the following listing:

```
if <some condition>
then
if <some condition>
then <some command>
end if;
else <some command>
end if
```

Example

```
declare
pi constant NUMBER(9,7) := 3.1415927;
area NUMBER(14,2);
cursor rad_cursor is
select * from RADIUS_VALS;
```

```
rad_val rad_cursor%ROWTYPE;
begin
open rad_cursor;
fetch rad_cursor into rad_val;
area := pi*power(rad_val.radius,2);
if area >30
then
insert into AREAS values (rad_val.radius, area);
end if;
close rad_cursor;
end;
```

4.5 Loops

You can use loops to process multiple records within a single PL/SQL block. PL/SQL supports three types of loops:

1. Simple loops: A loop that keeps repeating until an **exit** or **exit when** statement is reached within the loop
2. FOR loops: A loop that repeats a specified number of times
3. WHILE loops: A loop that repeats while a condition is met

Simple loops

The simple loop is started by the **loop** keyword and the **exit when** clause determines when the loop should be exited. An **end loop** clause identifies the end of the loop.

Example:

```
loop
area := pi*power(radius,2);
insert into AREAS values (radius, area);
radius := radius+1;
exit when area >100;
end loop;
```

Simple Cursor Loops

You can use the attributes of a cursor—such as whether or not any rows are left to be fetched—as the exit criteria for a loop. To determine the status of the cursor, the cursor's attributes are checked. Cursors have four attributes you can use in your program:

- %FOUND A record can be fetched from the cursor
- %NOTFOUND No more records can be fetched from the cursor
- %ISOPEN The cursor has been opened
- %ROWCOUNT The number of rows fetched from the cursor so far

The %FOUND, %NOTFOUND, and %ISOPEN cursor attributes are Booleans; they are set to either TRUE or FALSE.

Example:

```
declare
pi constant NUMBER(9,7) := 3.1415927;
area NUMBER(14,2);
cursor rad_cursor is
select * from RADIUS_VALS;
rad_val rad_cursor%ROWTYPE;
begin
open rad_cursor;
loop
fetch rad_cursor into rad_val;
exit when rad_cursor%NOTFOUND;
area := pi*power(rad_val.radius,2);
insert into AREAS values (rad_val.radius, area);
end loop;
close rad_cursor;
end;
```

FOR Loops

In a FOR loop, the loop executes a specified number of times. The FOR loop's start is indicated by the keyword **for**, followed by the criteria used to determine when the processing is complete

and the loop can be exited. Since the number of times the loop is executed is set when the loop is begun, an **exit** command isn't needed within the loop.

Example:

```
for radius in 1..7 loop
area := pi*power(radius,2);
insert into AREAS values (radius, area);
end loop;
```

Cursor FOR Loops

In a Cursor FOR loop, the results of a query are used to dynamically determine the number of times the loop is executed. In a Cursor FOR loop, the opening, fetching, and closing of cursors is performed implicitly; you do not need to explicitly specify these actions.

Example:

```
for rad_val in rad_cursor
loop
area := pi*power(rad_val.radius,2);
insert into AREAS values (rad_val.radius, area);
end loop;
```

for rad_val in rad_cursor implicitly opens the rad_cursor cursor and fetches a value into the rad_val variable. When no more records are in the cursor, the loop is exited and the cursor is closed. In a Cursor FOR loop, there is no need for a **close** command. Note that rad_val is not explicitly declared in the block.

WHILE Loops

In a WHILE loop; the loop is processed until an exit condition is met. Instead of specifying the exit condition via an **exit** command within the loop, the exit condition is specified in the **while** command that initiates the loop.

Example:

```
while radius<=7
loop
```

```
area := pi*power(radius,2);
insert into AREAS values (radius, area);
radius := radius+1;
end loop;
```

4.6 CASE Expressions

A CASE expression selects a result and returns it. To select the result, the CASE expression uses a selector, an expression whose value is used to select one of several alternatives. The selector is followed by one or more WHEN clauses, which are checked sequentially. The value of the selector determines which clause is executed. if the value of the selector equals the value of a WHEN-clause expression, that WHEN clause is executed. PL/SQL also provides a searched CASE expression, which has the form:

```
CASE
WHEN search condition1 THEN result1
WHEN search condition2 THEN result2
WHEN search conditionN THEN resultN
[ELSE resultN+1;]
END;
```

Example:

```
BEGIN
v_appraisal :=
CASE v_grade
WHEN 'A' THEN 'Excellent'
WHEN 'B' THEN 'Very Good'
WHEN 'C' THEN 'Good'
ELSE 'No such grade'
END;
```

4.7 Exception Handling Section

When user-defined or system-related exceptions (errors) are encountered, the control of the PL/SQL block shifts to the Exception Handling section. Within the Exception Handling section, the **when** clause is used to evaluate which exception is to be "raised"—that is, executed. If an exception is raised within the Executable Commands section of your PL/SQL block, the flow of commands immediately leaves the Executable Commands section and searches the Exception Handling section for an exception matching the error encountered. PL/SQL provides a set of system-defined exceptions and allows you to add your own exceptions.

The Exception Handling section always begins with the keyword **exception**, and it precedes the **end** command that terminates the Executable Commands section of the PL/SQL block.

Example:

```
declare
        pi constant NUMBER(9,7) := 3.1415927;
        radius INTEGER(5);
        area NUMBER(14,2);
        some_variable NUMBER(14,2);
begin
        radius := 3;
        loop
                some_variable := 1/(radius-4);
                area := pi*power(radius,2);
                insert into AREAS values (radius, area);
                radius := radius+1;
                exit when area >100;
        end loop;
exception
        when ZERO_DIVIDE
        then insert into AREAS values (0,0);
end;
```

4.7.1 Predefined Internal Exceptions

Any ORACLE error "raises"an exception automatically ; some of the more common ones have names.

1. TOO_MANY_ROWS ORA-(01427)- a single row SELECT returned more than one row
2. NO_DATA_FOUND ORA-(01403) -a single row SELECT returned no data
3. INVALID_CURSOR ORA-(01001) -invalid cursor was specified
4. VALUES_ERROR ORA-(06502) -arithmetic ,numeric, string , conversion, or constraint error occurred.
5. ZERO_DIVIDE ORA-(01476) -attempted to divide by zero
6. DUP_VAL_ON_INDEX ORA-(00001) -attempted to insert a duplicate value into a column that has a unique index specified.

4.7.2 User -Defined Exceptions

User -defined Exceptions must be defined and explicitly raised by the user.

Example

```
DECLARE x NUMBER;
my_exception EXCEPTION; --a new object type..
RAISE my_exception;
```

1. Once an exception is RAISED manually, it is treated exactly the same as if it were a predefined internal exception.
2. Declared exceptions are scoped just like variables.
3. A user-defined exception is checked for manually and then RAISED , if appropriate.

4.8 Triggers

A trigger defines an action the database should take when some database related event occurs. Triggers may be used to supplement declarative referential integrity, to enforce complex business rules, or to audit changes to data. The code within a trigger, called the trigger body, is made up of PL/SQL blocks The execution of triggers is transparent to the user. Triggers are executed by the database when specific types of data manipulation commands are performed on specific tables. Such commands may include **insert**s, **update**s, and **delete**s. Updates of specific

columns may also be used as triggering events

4.8.1.Types of Triggers

A trigger's type is defined by the type of triggering transaction and by the level at which the trigger is executed.

4.8.1.1 Row-Level Triggers

Row-level triggers execute once for each row affected by a DML statement. For the BOOKSHELF table auditing example described earlier, each row that is changed in the BOOKSHELF table may be processed by the trigger. Row-level triggers are the most common type of trigger; they are often used in data auditing applications. Row-level triggers are also useful for keeping distributed data in sync. Row-level triggers are created using the **for each row** clause in the **create trigger** command.

4.8.1.2 Statement-Level Triggers

Statement-level triggers execute once for each DML statement. For example, if a single INSERT statement inserted 500 rows into the BOOKSHELF table, a statement-level trigger on that table would only be executed once. Statement-level triggers therefore are not often used for data-related activities; they are normally used to enforce additional security measures on the types of actions that may be performed on a table. Statement-level triggers are the default type of trigger created via the **create trigger** command.

4.8.1.3 Before And After Triggers

Because triggers are executed by events, they may be set to occur immediately before or after those events. Since the events that execute triggers include database DML statements, triggers can be executed immediately before or after **insert**s, **update**s, and **delete**s.

Within the trigger, you can reference the old and new values involved in the DML statement. The access required for the old and new data may determine which type of trigger you need. "Old" refers to the data as it existed prior to the DML statement; **update**s and **delete**s usually reference old values. "New" values are the data values that the DML statement creates. If you

need to set a column value in an inserted row via your trigger, then you need to use a BEFORE INSERT trigger to access the "new" values.

4.8.1.4 INSTEAD OF Triggers

You can use INSTEAD OF triggers to tell Oracle what to do instead of performing the actions that invoked the trigger. For example, you could use an INSTEAD OF trigger on a view to redirect **insert**s into a table or to **update** multiple tables that are part of a view. The code in the INSTEAD OF trigger is executed in place of the **insert**, **update**, or **delete** command you enter.

Syntax

```
CREATE OR REPLACE TRIGGER trigger_name
[BEFORE | AFTER] [INSERT | UPDATE | DELETE]
ON table_name
[FOR EACH ROW] [WHEN condition]
BEGIN
--
-- trigger body
--
END;
```

Example

```
create or replace trigger BOOKSHELF_BEF_UPD_ROW
before update on BOOKSHELF
for each row
when (new.Rating < old.Rating)
begin
insert into BOOKSHELF_AUDIT
(Title, Publisher, CategoryName,
Old_Rating, New_Rating, Audit_Date)
values
(:old.Title, :old.Publisher, :old.CategoryName,
```

```
:old.Rating, :new.Rating, Sysdate);
end;
```

4.8.2 Enabling and Disabling Triggers

By default, a trigger is enabled when it is created. However, there are situations in which you may want to disable a trigger. To enable a trigger, use the **alter trigger** command with the **enable** keyword.

```
alter trigger BOOKSHELF_BEF_UPD_INS_ROW enable;
```

A second method of enabling triggers uses the **alter table** command, with the **enable all triggers** clause.

```
alter table BOOKSHELF enable all triggers;
```

You can disable triggers using the same basic commands (requiring the same privileges) with modifications to their clauses. For the **alter trigger** command, use the **disable** clause:

```
alter trigger BOOKSHELF_BEF_UPD_INS_ROW disable;
```

For the **alter table** command, use the **disable all triggers** clause:

```
alter table BOOKSHELF disable all triggers;
```

4.8.3 Replacing Triggers

The status of a trigger is the only portion that can be altered. To alter a trigger's body, the trigger must be re-created or replaced. When replacing a trigger, use the **create or replace trigger** command

4.8.4 Dropping Triggers

Triggers may be dropped via the **drop trigger** command. To drop a trigger, you must either own the trigger or have the DROP ANY TRIGGER system privilege. An example of this command is shown in the following listing:

drop trigger BOOKSHELF_BEF_UPD_INS_ROW;

4.9 Cursor Overview

Every SQL DML statement processed by PL/SQL has an associated CURSOR. Two Types of CORSORS

1. EXPLICIT . Multiple row SELECT STATEMENTS
2. IMPLICIT

   All INSERT statements

   All UPDATE statements

   All DELETE statements

   Single row SELECT….INTO Statements

Cursors have four attributes you can use in your program:

- %FOUND A record can be fetched from the cursor
- %NOTFOUND No more records can be fetched from the cursor
- %ISOPEN The cursor has been opened
- %ROWCOUNT The number of rows fetched from the cursor so far

The %FOUND, %NOTFOUND, and %ISOPEN cursor attributes are Booleans; they are set to either TRUE or FALSE. Because they are Boolean attributes, you can evaluate their settings without explicitly matching them to values of TRUE or FALSE.

4.9.1 Using explicit cursors

STEP 1 . Declare the cursor

DECLARE CURSOR <cursorname> IS <regular select statement> ;

1. The < regular select statement > must NOT include the INTO clause required in a single-row SELECT….INTO statement.
2. Declared cursors are scoped just like variables.

Cursor Declaration Example

```
DECLARE XNUMBER ( 7, 2 ) ;
Total NUMBER ( 5 )
lower_sal_limit CONSTANT NUMBER ( 4 ) := 1200 ;
CURSOR c1 IS SELECT ename FROM empWHERE sal> lower_sal_limit;
BEGIN ...
```

STEP 2 . Open the cursor

```
OPEN < cursor name > ;
```

STEP 3 . Fetch data from the cursor.

```
FETCH < cursor name > INTO < var1 ,var2 >> > ;
```

1. Retrieves one row of data from the cursor , and stores it in the specified variables (similar to how a single-row select works) .
2. There must be exactly one INTO variable for each column selected by the SELECT statement .
3. The first column gets assigned to var1 , the second to var2 , etc .

STEP 4 . Close the cursor CLOSE < cursor name > ;

4.9.2 Implicit Cursors

An Implicit Cursor is automatically associated with any SQL DML statement that does not have an explicit cursor associated with it. This includes :

1. ALL INSERT statements
2. ALL UPDATE statements
3. ALL DELETE statements
4. ALL SELECT…INTO statements

- Implicit cursor is called the “SQL”cursor --it stores information concerning the processing of the last SQL statement not associated with an explicit cursor.

- OPEN, FETCH, AND CLOSE don't apply.
- All cursor attributes apply.

4.9.3 Selecting Rows for Update

You can lock rows by using the **select for update** syntax. For example, the following query selects the rows from the BOOK_ORDER table and locks them to prevent other users from acquiring update locks on the rows. Using **select for update** allows you to use the **where current of** clause in **insert**, **update**, and **delete** commands. A **commit** will invalidate the cursor, so you will need to reissue the **select for update** after every **commit**.

```
select * from BOOK_ORDER for update of Title;
```

When the preceding query is executed, the optimizer will first perform a TABLE ACCESS FULL operation to retrieve the rows from the BOOK_ORDER table. The TABLE ACCESS FULL operation returns rows as soon as they are retrieved; it does not wait for the full set to be retrieved. However, a second operation must be performed by this query. The FOR UPDATE optimizer operation is called to lock the records. It is a set-based operation (like the sorting operations), so it does not return any rows to the user until the complete set of rows has been locked.

4.10 Stored procedures and Functions

- Collections of SQL and PL/SQL statements.
- Stored in complied form in the database.
- Can call others and self.
- Can be called from all client environments
- Procedures and function (including remote) are the same, except a function returns a values and a procedure does not.

4.10.1 Uses for procedures

- Define central well-known business functions. -Create an order -Delete a customer
- Store batch job in the database -Weekly account rollups

- Encapsulate a transaction -Gather and process information from remote nodes
- Funnel activity on a table through a single path -all changes to employee salaries should change department budgets.

Procedures vs. Functions

Functions can return a value to the caller, and functions can be directly referenced in queries. This value is returned through the use of the return keyword within the function.

4.10.2 Creating a procedure - Argument Modes

- IN Data value comes in from the calling process and is not changed
- OUT No data value comes in from the calling process; on normal exit ,value of argument is passed back to caller
- IN OUT Data value comes in from the calling process, and another value is returned on normal exit

Creating a procedure- create procedure Syntax

```
create [or replace] procedure procedurename
[( argument [ in | out | in out ] [nocopy] datatype
[, argument [ in | out | in out ] [nocopy] datatype]...
)]
{ is | as }
{ pl/sql_subprogram_body };
```

Example:

```
create or replace procedure NEW_BOOK (aTitle IN VARCHAR2,
aPublisher IN VARCHAR2, aCategoryName IN VARCHAR2)
as
begin
```

```
insert into BOOKSHELF (Title, Publisher, CategoryName, Rating)
values (aTitle, aPublisher, aCategoryName, NULL);
delete from BOOK_ORDER
where Title = aTitle;
end;
```

4.10.3 Create function Syntax

The syntax is:

```
create [or replace] function  functionname
[( argument [ in | out | in out ] [nocopy] datatype
[, argument [ in | out | in out ] [nocopy] datatype]...
)]
return datatype
{ is | as }
 { pl/sql_function_body };
```

Both the header and the body of the function are created by this command.

The **return** keyword specifies the datatype of the function's return value. This can be any valid PL/SQL datatype .Every function must have a **return** clause, since the function must, by definition, return a value to the calling environment. The **nocopy** keyword tells Oracle to pass the variable value back to the user as quickly as possible.

Example

```
CREATE FUNCTION get_bal  (acc_no  NUMBER(4))
RETURN      NUMBER(11,2) IS acc_bal   NUMBER(11,2);
BEGIN
SELECT balance INTO acc_bal FROM accounts WHERE
account_id_no=acc_no;
RETURN  (acc_bal);
END;
```

4.10.4 Drop Procedure Syntax

Syntax

Drop procedure procedure_name

Example

Drop procedure NEW_BOOK;

4.10.5 Drop Function Syntax

Syntax

Drop function function_name

Example

Drop function get_bal;

Possible Questions

Part-A

(Online – Multiple choice questions)

(Each questions carries one mark each)

PART – B (2 Marks)

1. Write a short note on any one of the conditional logic command.
2. What are cursors?
3. What is user defined exception?
4. What are data types?
5. What are triggers and give its types?

PART – C (8 Marks)

1. What are cursors? Discuss how you will use cursors in PL/SQL with syntax.
2. What are Triggers. How are triggers used in PL/SQL. Explain with syntax
3. What are CASE expressions? Give the syntax and write a PL/SQL block to print Grade of a student given the mark percentage.
4. Write in detail about i) Internal exception ii) User defined exceptions
5. Write in detail about various data types used in PL/SQL.
6. Describe the characteristics of using functions in PL/SQL. Give the syntax for defining a function.
7. Write in detail about the conditional structures in PL/SQL. Give the syntax and explain with an example.
8. What are procedures? With a detailed explanation give the structure of a procedure with example.
9. Describe the looping statements used in PL/SQL with example.

Part – A - Included in Excel file – **File name** Unit-IV(MCQ).xls

- Objective type / Multiple choice questions. Each question carries one mark
- It is for online examination

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

Subject: Relational Database Management System Sub.code: 17ITU303

S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____marks a sequence of statements that has to be repeated.	procedure	subrouting	package	loop	**loop**
2	_____is called private work area	triggers	packages	cursors	packages	**cursors**
3	The data that is stored in the cursor is called the_____	Active data set.	packages	relations	tables	**Active data set.**
4	_____is a procedural language extension to SQL.	PSQL	PL/SQL	PRO/SQL	RA	**PL/SQL**
5	_____is a superset of SQL.	PSQL	PL/SQL	PRO/SQL	RA	**PL/SQL**
6	code block starts with _____section in PL/SQL	begin	end	start	declare	**declare**
7	_____deals with handling of errors that arise during execution of the data manipulation statements which make up the PL/SQL code block.	begin	exception	end	declare	**exception**
8	_____is a numeric value or a character string used to represent itself.	character set	identifiers	Lexical Units	literals	**literals**

9	_____literal returns integers or floats.	boolean	numeric	decimal	character	**numeric**
10	_____are predetermined constants	logical literal	numeric literal	float literal	character literal	**float literal**
11	_____is used to declare the variable in the PL/SQL.	character literal	numeric literal	logical literal	identifiers	**logical literal**
12	_____can not be used as a variable names.	words	literals	numbers	reserved words	**reserved words**
13	_____data type is used to store binary data	varchar2	char	raw	date	**raw**
14	The _____section marks the end of a PL/SQL block.	Terminate	end	last	next	**end**
15	The maximum length of the long raw column is_____	2GB	4GB	6GB	1GB	**2GB**
16	User defined cursor is known as_____	implicit cursor	explicit cursor	internal cursor	external cursor	**explicit cursor**
17	If the Oracle engine opened a cursor for its internal processing it is known as_____	explicit cursor	implicit cursor	internal cursor	external cursor	**implicit cursor**
18	The loop statement must end with an _____statement	end loop	loop	end	goto	**end loop**
19	Which keyword should be placed before the first statement in the sequence of PL/SQL block?	loop	begin	declare	for loop	**begin**
20	Which attribute is used to count the number of rows affected by an insert, update, delete statement?	%COUNT	%ROWCOUNT	%CALCULATE	%ROWCALCULATE	**%ROWCOUNT**
21	Which attribute is used to open and close the cursor automatically?	%FOUND	%NOTFOUND	%ISOPEN	%ROWCOUNT	**%ISOPEN**

22	An _____ cursor is automatically declared by Oracle every time an SQL statement is executed.	implicit cursor	explicit cursor	internal cursor	external cursor	**implicit cursor**
23	A cursor is defined in _____part of a PL/SQL block.	begin	declarative part	end	exception	**declarative part**
24	_____are composite types that have internal components that can be manipulated individually, such as the elements of an array, record, or table.	collections and records	composite attribute and single attribute	collection types	nested tables	**collections and records**
25	_____statement disables the cursor	disable cursor	close cursor	end cursor	remove cursor	**close cursor**
26	_____are similar to C or Pascal pointers, which hold the memory location (address) of some item instead of the item itself.	trigger	procedure	pointers	cursor variables	**cursor variables**
27	_____can be a procedure or function	stored procedure	query	nested program	subprogram	**subprogram**
28	subprogram created inside a PL/SQL block is a_____	combined subprogram	collective subprogram	nested subprogram	packaged subprogram	**nested subprogram**
29	_____is a logically grouped set of SQL and PL/SQL statements that perofrm a specific task.	procedure	cursors	package	trigger	**procedure**
30	A _____is the building block of modular programming.	packages	procedure	cursor	exception	**procedure**

31	The parameter passed in a call statement are called the _____	parameter marker	parameter statements	actual parameters	formal parameters	**actual parameters**
32	The parameter names in the header of a module are called_____	parameter marker	parameter statements	actual parameters	formal parameters	**formal parameters**
33	_____returns a value back to the calling block.	procedure	package	function	trigger	**function**
34	The function header comes before the reserved word_____	IN	IS	NOT IN	RETURN	**IS**
35	_____is a collection of PL/SQL objects	procedure	functions	trigger	package	**package**
36	The objects in the specification section of a package are called_____	private objects	public objects	package variable	modules header	**public objects**
37	The module code in the body without its description in the specification is called _____	private module	public module	definition	function	**private module**
38	The execution of a trigger is also known as _____	executing the trigger	firing the trigger	perfrom the trigger	implement trigger	**firing the trigger**
39	_____trigger is fired before execution of a DML statement.	previous	after	before	prior	**before**
40	_____allows to access the currently processed row.	another	:new	:fresh	:recent	**:new**
41	_____trigger fires after a DML statement is executed.	after	later	next	when	**after**
42	for each row trigger is known as_____	trigger	tuple trigger	record trigger	row trigger	**row trigger**
43	_____can also be used to keep an audit trail of a table.	trigger	procedure	package	function	**trigger**

44	Which keyword is used to specifies the trigger restriction?	control	restrict	check	when	**when**
45	Which keyword is used to delete the trigger?	delete	drop	remove	truncate	**drop**
46	_____allows programmers to issue user-defined error messages.	raise_error	application error	raise_application_error	error_message	**raise_application_error**
47	What are the ranges for error numbers in the trigger?	2000 to 2099	20001-29000	-20000 to -20999	5000-10000	**-20000 to -20999**
48	Procedure and functions are stored in the _____	oracle database	DBMS	repository	data warehouse	**oracle database**
49	When the procedure or function is invoked, the oracle engine loads the compiled procedure or function in a memory are called_____	process global area	procedure global area	System global area	function global area	**System global area**
50	How many copies are needed to be loaded for execution by multiple user?	1	2	3	4	**1**
51	A block of PL/SQL code contains _	one	two	three	four	**three**
52	If the block of PL/SQL code is not	anonymous bloc	null block	empty block	fuctional block	**anonymous block**
53	The _____ section begins a l	exception	declaration	begin	end	**declaration**
54	_____ data types hold a si	Vector	Selected	Scalar	Comitte	**Scalar**
55	Reference data types hold values, c	pointers	arrays	constants	composite	**pointers**
56	A _____ loop that repeats a	FOR	WHILE	Simple loops	IF	**FOR**
57	An _____ clause identifies the	exit	exit loop	end loop	end	**end loop**
58	Since the number of times the loop	end	last	ending	exit	**exit**
59	In a Cursor FOR loop, there is no r	close	exit loop	end loop	end	**close**
60	A _____ expression selects a res	WHILE	CASE	DO	FOR	**CASE**

UNIT-V
SYLLABUS

Package header – package body – calling package members - Replacing and dropping package. Overview of Normalization : advantages - disadvantages. Normal forms: first normal form – second normal form – third normal form – boyce codd normal form – Introduction to fourth, fifth and sixth normal forms – denormalization. Parallel Databases: Introduction – Design of Parallel Databases – Advantages and Disadvantages of Parallel Databases.

Package

- A database object that groups related package constructs like Procedures, functions, cursor definitions, variables and constants, exception definitions.
- Package variables and cursors have persistent state.
- Variables retain values and cursors retain contexts and positions for the duration of a user session.
- State persists across a user's calls in one session (not multiple sessions or users).

5.1 Parts of a package- Package specification

- Declares (specifies) package constructs, including names and parameters publicly available procedures and functions.

5.2 Package body

- May declare additional, private package constructs that are not publicly available. Defines all package constructs (public and private).
- May be replaced without affecting package specification (Breaks dependency chain) Each session has own version of state.

5.3 Create package Syntax

When creating packages, the package specification and the package body are created separately. Thus, there are two commands to use: **create package** for the package specification, and **create package body** for the package body.

create [or replace] package packagename

{is | as}

package specification;

A package specification consists of the list of functions, procedures, variables, constants, cursors, and exceptions that will be available to users of the package.

A package body contains the blocks and specifications for all of the public objects listed in the package specification. The package body may include objects that are not listed in the package specification; such objects are said to be private and are not available to users of the package. Private objects may only be called by other objects within the same package body.

The syntax for creating package bodies is

create [or replace] package body packagebody
{is | as}
package body;

The name of the package body should be the same as the name of the package specification.

Example:

```
create or replace package BOOK_MANAGEMENT
as
function OVERDUE_CHARGES(aName IN VARCHAR2) return NUMBER;
procedure NEW_BOOK (aTitle IN VARCHAR2, aPublisher IN VARCHAR2,
aCategoryName IN VARCHAR2);
end BOOK_MANAGEMENT;
create or replace package body BOOK_MANAGEMENT
as
function OVERDUE_CHARGES (aName IN VARCHAR2)
return NUMBER
is
owed_amount NUMBER(10,2);
begin
```

```
select SUM(((ReturnedDate-CheckoutDate) -14)*0.20) into owed_amount
from BOOKSHELF_CHECKOUT
where Name = aName and (ReturnedDate-CheckoutDate) > 14;
RETURN(owed_amount);
EXCEPTION
when NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20100,'No books borrowed.');
end OVERDUE_CHARGES;
procedure NEW_BOOK (aTitle IN VARCHAR2,
aPublisher IN VARCHAR2, aCategoryName IN VARCHAR2)
is
begin
insert into BOOKSHELF (Title, Publisher, CategoryName, Rating)
values (aTitle, aPublisher, aCategoryName, NULL);
delete from BOOK_ORDER where Title = aTitle;
end NEW_BOOK;
end BOOK_MANAGEMENT
```

5.4 Replacing Procedures, Functions, and Packages

Procedures, functions, and packages may be replaced via their respective **create or replace** command. Using the **or replace** clause keeps in place any existing grants that have been made for those objects.

Dropping Procedures, Functions, and Packages

- To drop a procedure, use the **drop procedure** command, as follows:

Syntax

```
drop procedure procedure_name;
```

Example

```
drop procedure NEW_BOOK;
```

- To drop a function, use the **drop function** command, as follows:

Syntax

drop function function_name;

Example

drop function OVERDUE_CHARGES;

- To drop a package (both the specification and the body), use the **drop package** command, as follows:

Syntax

drop package package_name;

Example

drop package BOOK_MANAGEMENT;

- To drop a package body, use the **drop package** command with the **body** clause, as follows:

Syntax

drop package package_body_name;

Example

drop package body BOOK_MANAGEMENT;

5.5 Overview of Normalization

Normalization is the process of reducing the redundancy of data in a relational database. Redundant data refers to the data that is stored in more than one location in the database. Data should not be redundant, which means that the duplication of data should be kept to a minimum for several reasons.

Normalization is the application of a set of simple rules called FIRST, SECOND, and THIRD NORMAL FORM to assign attributes to entities in the ERD. Although there are additional levels

of normalization beyond THIRD NORMAL FORM such as Boyce-Codd, FOURTH, and FIFTH levels of NORMAL FORM , normalization of a production relational database generally stops at the THIRD NORMAL FORM.

The normalization process is fundamental to the modeling and design of a relational database. Its purpose is to eliminate data redundancy, avoid data update anomalies that can occur in unnormalized databases (databases that have not been normalized), and to simplify enforcement of integrity constraints

5.6 Advantages of Normalization

Normalization provides numerous benefits to the design of a database, the design of an application, and the implementation of the production database. Some of the major benefits include

- Greater overall database organization will be gained.
- The amount of unnecessary redundant data is reduced.
- Data integrity is easily maintained within the database.
- The database and application design processes are much more flexible.
- Security is easier to manage.

5.7 Disadvantages of Normalization

The disadvantage of normalization is that it produces a lot of tables with a relatively small number of columns. These columns then have to be joined using their primary/foreign key relationships in order to put the information back together so we can use it. For example, a query might require retrieval of data from multiple normalized tables. This can result in complicated table joins. The required tables for the query were probably one before decomposition (normalization).

Decomposition of tables has two primary impacts. The first is performance. All the joins required to merge data slow processing down and place additional stress on your hardware. The second impact challenges developers to code queries that return desired information, without experiencing the impact of the relational database's insistence on returning a row for every possible combination of matching values if the tables are not properly joined by the developer. Additional rows that are returned because of tables that are not properly joined (using their key values) are extraneous nonsense. This collection of extraneous data is called a Cartesian Product.

5.8 Overview of the Normal Forms

NORMAL FORM is a way of measuring the levels, or depth, to which a database has been

normalized. A database's level of normalization is determined by the NORMAL FORM. The NORMAL FORMS discussed here are as follows:

- FIRST NORMAL FORM
- SECOND NORMAL FORM
- THIRD NORMAL FORM
- Boyce-Codd NORMAL FORM
- FOURTH NORMAL FORM
- FIFTH NORMAL FORM

The three most common NORMAL Forms implemented in production databases are the FIRST, SECOND, and THIRD NORMAL FORMS. Every instance in the entity must have a unique identifier (UID) or primary key before you implement FIRST NORMAL FORM. Each NORMAL FORM builds on the previous. FIRST NORMAL FORM must be complete before SECOND NORMAL FORM begins, and second must be complete before beginning third. Boyce-Codd builds on THIRD NORMAL FORM. FOURTH NORMAL FORM builds on Boyce-Codd. FIFTH NORMAL FORM builds on FOURTH NORMAL FORM and so on.

5.8.1 Zero Normal Form

Zero normal form (ZNF) is simply a set of data that has not gone through the process of normalization. It is quite possible that it meets the requirements of normalization already. However, going through the normalization steps on any data set is a best practice. In our example, the data is not normalized and the normalization process will be followed.

MemberCompanyDatabase			
MID	MEMBER	COMPANY	DATABASE
1	John Smith	ABC	Sybase, Oracle
2	Jane Doe	MNO	Oracle
3	Dick Jones	ABC	Informix, Sybase
4	Carol Brown	XYZ	MySQL
5	Ben Adams	IJK	DB2, Oracle
6	Shirley Kent	ABC	MySQL
7	Steve Date	ABC	DB2
8	Anne May	MNO	Sybase, MySQL

5.8.2 First Normal Form (1NF)

The objective of the FIRST NORMAL FORM is to divide the base data into logical units called entities, or tables. When each entity has been designed, a primary key is assigned to it. The entity has a UID (key) and all attributes must be single valued. A repeating or multivalued attribute is an attribute or group of attributes that will have multiple values for one occurrence of the UID.

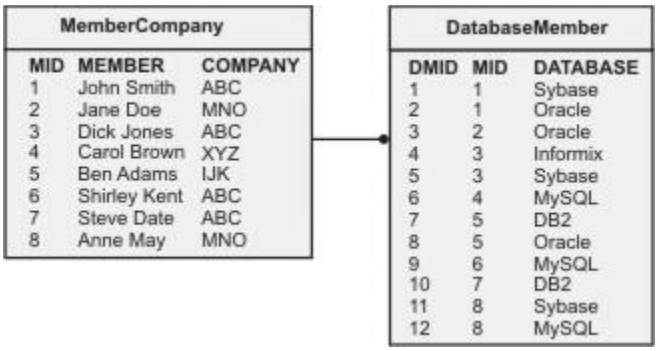
You would have to repeat the attribute(s) multiple times in the entity structure in order to capture all the possible occurrences of the event being modeled. The solution is to move repeating attribute(s) to their own entity and create a relationship between the two decomposed entities. Working with tables in design, you'd move the repeating columns to their own table along with a copy of the original entity's UID as a foreign key.

Conversion to First Normal Form

- A relational table must not contain repeating groups.
- Repeating groups can be eliminated by adding the appropriate entry in at least the primary key column(s).

The first **normal form** (1NF) requires that the values in each column of a table are atomic. By atomic we mean that there are no sets of values within a column.

In our example, the Database column in the MemberCompanyDatabase table contains repeating groups. Normalization requires us to create a new table called DatabaseMember where there is only one column with one value for Database per row.



MemberCompany

MID	MEMBER	COMPANY
1	John Smith	ABC
2	Jane Doe	MNO
3	Dick Jones	ABC
4	Carol Brown	XYZ
5	Ben Adams	IJK
6	Shirley Kent	ABC
7	Steve Date	ABC
8	Anne May	MNO

DatabaseMember

DMID	MID	DATABASE
1	1	Sybase
2	1	Oracle
3	2	Oracle
4	3	Informix
5	3	Sybase
6	4	MySQL
7	5	DB2
8	5	Oracle
9	6	MySQL
10	7	DB2
11	8	Sybase
12	8	MySQL

5.8.3 Second Normal Form (2NF)

A table is in 2NF if:

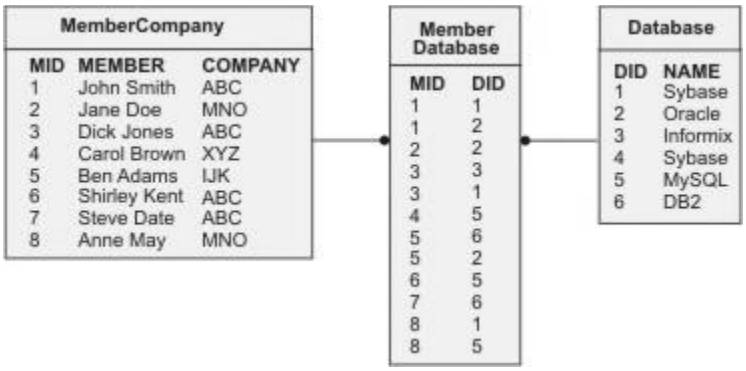
- It is in 1NF and
- It includes no partial dependencies; that is, no attribute is dependent on only a portion of the primary key. (It is still possible for a table in 2NF to exhibit transitive dependency; that is, one or more attributes may be functionally dependent on non-key attributes.)

Conversion to Second Normal Form

Where the First **Normal Form** deals with atomicity of data, the Second **Normal Form** (or 2NF) deals with relationships between composite key columns and non-key columns.

In the second **normal form** (or 2NF) any non-key columns must depend on the entire primary key. In the case of a composite primary key, this means that a non-key column cannot depend on only part of the composite key.

In our example, the Database column was only dependent on part of the MemberDatabase composite key. Normalization requires creating a new table we call Database where the column values are unique.



MemberCompany

MID	MEMBER	COMPANY
1	John Smith	ABC
2	Jane Doe	MNO
3	Dick Jones	ABC
4	Carol Brown	XYZ
5	Ben Adams	IJK
6	Shirley Kent	ABC
7	Steve Date	ABC
8	Anne May	MNO

Member Database

MID	DID
1	1
1	2
2	2
3	3
3	1
4	5
5	6
5	2
6	5
7	6
8	1
8	5

Database

DID	NAME
1	Sybase
2	Oracle
3	Informix
4	Sybase
5	MySQL
6	DB2

5.8.4 Third Normal Form (3NF)

3NF Definition

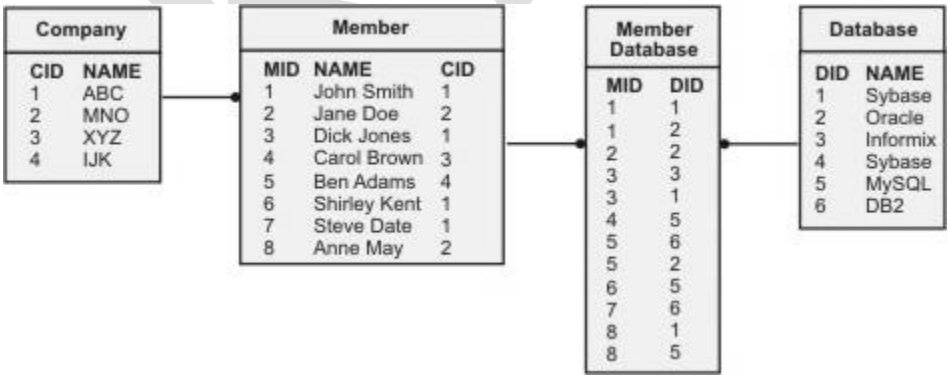
A table is in 3NF if:

- It is in 2NF and
- It contains no transitive dependencies.

Conversion to Third Normal Form

Third **Normal Form** (3NF) requires that all columns depend directly on the primary key. Tables violate the Third **Normal Form** when one column depends on another column, which in turn depends on the primary key (a transitive dependency).

In our example, the Company column in the MemberCompany table is a transitive dependency. Normalization requires creating a new table we call Company where the column values are unique



Company

CID	NAME
1	ABC
2	MNO
3	XYZ
4	IJK

Member

MID	NAME	CID
1	John Smith	1
2	Jane Doe	2
3	Dick Jones	1
4	Carol Brown	3
5	Ben Adams	4
6	Shirley Kent	1
7	Steve Date	1
8	Anne May	2

Member Database

MID	DID
1	1
1	2
2	2
3	3
3	1
4	5
5	6
5	2
6	5
7	6
8	1
8	5

Database

DID	NAME
1	Sybase
2	Oracle
3	Informix
4	Sybase
5	MySQL
6	DB2

5.8.5 Boyce-Codd Normal Form (BCNF)

A table is in BCNF if every determinant in that table is a candidate key. If a table contains only one candidate key, 3NF and BCNF are equivalent.

- A table is in Boyce-Codd normal form (BCNF) if every determinant in the table is a candidate key. (A determinant is any attribute whose value determines other values with a row.)
- If a table contains only one candidate key, the 3NF and the BCNF are equivalent.
- BCNF is a special case of 3NF.

Decomposition into BCNF

Problem

In our example, the purpose of the table is to show which members use which databases. The table's candidate keys are:

- {MemberID, DatabaseID}
- {MemberSocialSecurityNumber, DatabaseID}

Therefore all three attributes of the table are **prime** attributes: that is, all three attributes belong to candidate keys. 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. Since the table above lacks any non-prime attributes, it adheres to both 2NF and 3NF.

BCNF is more stringent than 3NF in that it does not permit any functional dependency in which the determining set of attributes is not a candidate key (or superset thereof). The dependency of MemberID on Member Social Security Number is such a dependency. Accordingly, the table above is not in BCNF.

Any table that falls short of BCNF will be vulnerable to logical inconsistencies. In the table above, there is nothing to prevent two different Member IDs from being shown, illegitimately, as corresponding to the same Member Social Security Number.

Solution

Correcting the problem in this case would be a simple matter of using only one scheme of identifiers for Members: either IDs or Social Security Numbers, but not both. In this case I would move Member Social Security Number into the Member Table.

5.8.6 Fourth Normal Form (4NF)

4NF Definition

A table is in 4NF if it is in 3NF and has no multiple sets of multivalued dependencies**.**

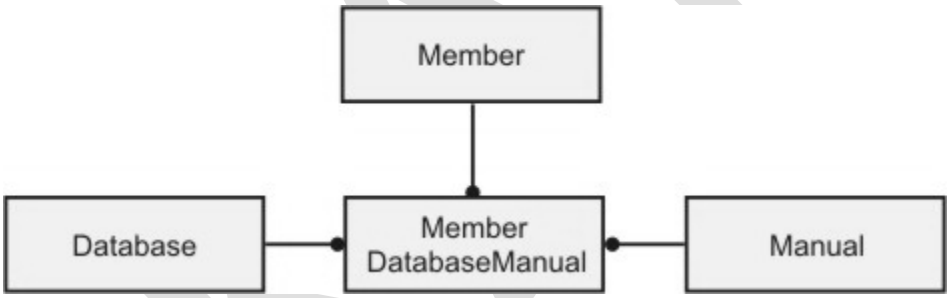
Problem

Fourth **normal form** (4NF) requires that independent multivalued facts are correctly and efficiently represented.

In our example, we have three entities with two many to many relationships

Incorrect Solution

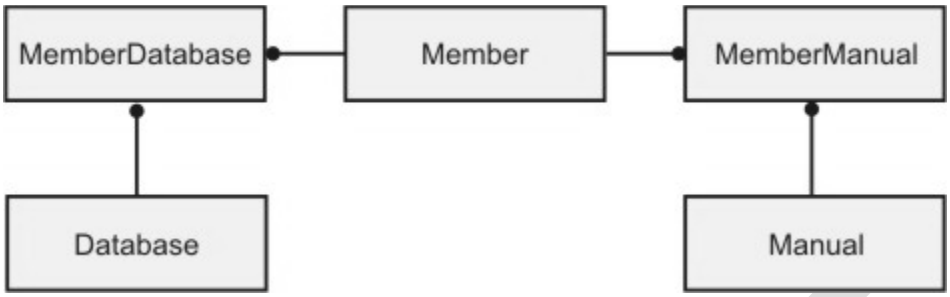
Often what happens is these three tables are combined into a ternary (three way) relationship which satisfies third **normal form**, but not fourth **normal form**.



Correct Solution

Notice that because the table has a unique key and no non-key attributes, it does not violate any **normal form** up to BCNF. But because the manuals a member owns are independent from the databases the member uses, there is redundancy in the table. In formal terms, this is described as manual having a multivalued dependency on database.

To satisfy 4NF, we must place the facts about manuals owned into a different table from the facts about databases used:



5.8.7 Fifth Normal Form (5NF)

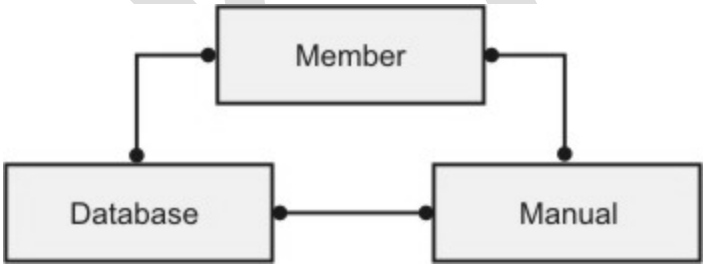
A 5NF (5th Normal Form) violation describes

- any relation that could be decomposed into 2 or more relations, where
- the relations do not all have the same set of candidate keys, and
- the relations are joined together to get back the original relation

These include the 4NF violations, but more complicated ones as well

Problem

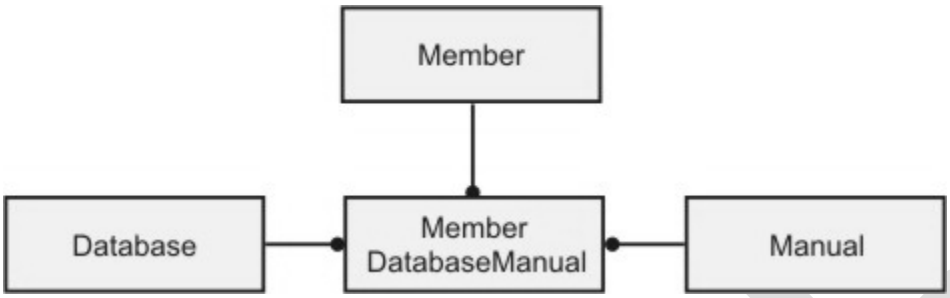
Fifth **normal form** (5NF) is required to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships.



No Rule Solution

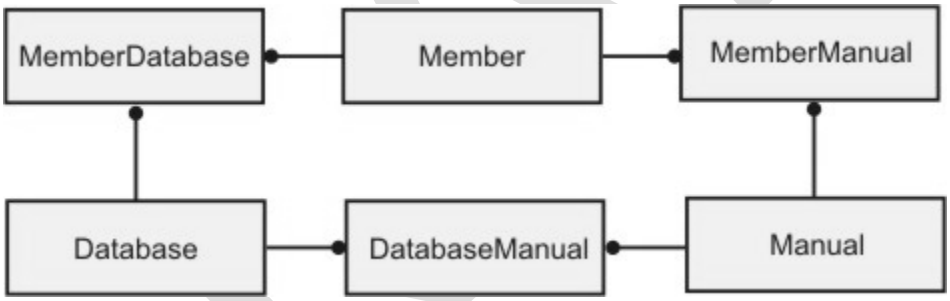
The member is able to offer support for the given database and when he owns a given manual. In the absence of any rules restricting the valid possible combinations of member, database, and

manual, the three-attribute table Member-to-Database-to-Manual is necessary in order to model the situation correctly.



Rule Solution

However, what if the above model the following rule applies: When a member owns a manual M and the member uses database D, then – in the event that the manual M covers the database D– it must be true that the member is able to offer support on database D and is supported by manual M. With these constraints it is possible to split the relation into three parts.



5.8.8 Domain/Key Normal Form

Domain/key **normal form** (DKNF) is a **normal form** used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.

A domain constraint specifies the permissible values for a given attribute, while a key constraint

specifies the attributes that uniquely identify a row in a given table.

The domain/key **normal form** is achieved when every constraint on the relation is a logical consequence of the definition of keys and domains, and enforcing key and domain restraints and conditions causes all constraints to be met. Thus, it avoids all non-temporal anomalies.

5.8.9 Sixth Normal Form (6NF)

A table is in sixth **normal form** (6NF) if and only if it satisfies no non-trivial join dependencies at all, meaning that the fifth **normal form** is also satisfied. The sixth **normal form** was only defined when extending the relational model to take into account the temporal dimension. Most SQL technologies, as of 2005, do not take into account this work, and most temporal extensions to SQL are not relational

5.8a.10.Denormalization

Denormalization is the process of taking a normalized database and modifying table structures to allow controlled redundancy for increased database performance. Data redundancy is increased in a denormalized database, which might improve performance but requires more extraneous efforts in order to keep track of related data.

A denormalized database is not the same as a database that has not been normalized. Denormalizing a database is the process of taking the level of normalization within the database down a notch or two. Denormalization might involve recombining separate tables, or creating duplicate data within tables. This will reduce the number of tables that need to be joined in order to retrieve the requested data, which results in less I/O and CPU time.

Denormalization generally involves one of these three tasks:

1. Replicating some data columns to several tables in order to have them more easily accessible without multi-table joins. This can include data columns themselves or foreign key columns. With replication of data columns, the data is often available to the application without joins. Additional foreign key columns can potentially allow direct

connections between tables rather than multiple joins including tables that are only providing a connection, but carry a large time cost in the join.

2. Pre-calculation and storage of derived data such as summary calculations or concatenations can speed processing. Create special purpose tables that contain pre-calculated information required by a commonly run report or create tables that contain data often needed for queries.
3. Undoing some decomposition of entities to avoid the price of multiple joins. This would primarily be undertaken for tables that have one-to-one relationships.

Possible Questions

Part-A

(Online – Multiple choice questions)

(Each questions carries one mark each)

PART – B (2 Marks)

1. Give the advantages of normalization.
2. Write the syntax for create package.
3. What is normalization?
4. What is de-normalization?
5. What is package?

PART – C (8 Marks)

1. Define normalization. Write in detail about any two normal forms with examples.
2. Explain in detail about package specification and package body?
3. What are the main parts to be defined for using a package? Explain them in detail with an example package.
4. Illustrate the procedure to convert a table into first normal form. Describe the procedure with example.
5. Write in detail about Second Normal Form with example
6. Explain in detail about replacing a package and dropping a Package
7. What are packages? Explain in detail the components of a PL/SQL package.
8. What are the conditions to be satisfied if a table is to be in third normal form? Explain in detail how will you convert a table to third normal form.
9. What are the uses of Normalization? With a neat table structure explain the procedure to convert a table in Zero Normal form to First normal form.
10. What are the main parts of a package? How packages are made use of in the extraction of information from relational databases.

Part – A - Included in Excel file – **File name** Unit-V(MCQ).xls

- Objective type / Multiple choice questions. Each question carries one mark
- It is for online examination

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

Subject: Relational Database Management System Sub.code: 17ITU303

S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____is an oracle object , which holds another object.	procedure	function	package	trigger	**package**
2	subprogram created inside a package is a_____	combined subprogram	collective subprogram	nested subprogram	packaged subprogram	**packaged subprogram**
3	_____is a package that includes a number of procedures and functions that accumulate informationin a buffer so that is can be retrieved later.	output	db_output	DBMS_OUTPUT	DBMS-OUTPUT	**DBMS_OUTPUT**
4	_____put a piece of information in the package buffer followed by an and-of-line marker.	PUT_LINE	DBMS_PUTLINE	DBMS-PUTLINE	PUTLINE	**PUT_LINE**
5	_____is a SQL * PLUS environment parameter that displays the information passed as a parameter to the PUT_LINE function.	SERVEROUTPUT	OUTPUT	SERVER PARAM	BUFFER	**SERVEROUTPUT**
6	_____property enables us to recover any instance of the decomposed relation from corresponding instance of the smaller relations.	dependency preservation	lossless-join	normal forms	decomposition	**lossless-join**

7	BCNF stands for_____	Boyce-Codd Normal Form	Boy Codd Normal Form	Basic Codd Normal Form	Basic Codd Normalization Form	**Boyce-Codd Normal Form**
8	Boye Codd Normal Form (BCNF) is needed when	two non-key attributes are dependent	there is more then one possible composite key	there are two or more possible composite overlapping keys and one attribute of a composite key is dependent on an attribute of another composite key	there are two possible keys and they are dependent on one another	**there are two or more possible composite overlapping keys and one attribute of a composite key is dependent on an attribute of another composite key**
9	A relation is said to be in BCNF when	it has overlapping composite keys	it has no composite keys	it has no multivalued dependencies	it has no overlapping composite keys which have related attributes	**it has no overlapping composite keys which have related attributes**
10	A 3 NF relation is converted to BCNF by	removing composite keys	removing multivalued dependencies	dependent attributes of overlapping composite keys are put in a separate relation	dependent non-key attributes are put in a separate table	**dependent attributes of overlapping composite keys are put in a separate relation**

11	BCNF is needed because	otherwise tuples may be duplicated	when a data is deleted tuples may be lost	updating is otherwise difficult	when there is dependent attributes in two possible composite keys one of the attributes is unnecessarily duplicated in the tuples	**when there is dependent attributes in two possible composite keys one of the attributes is unnecessarily duplicated in the tuples**
12	Fourth normal form (4 NF) relations are needed when	there are multivalued dependencies between attributes in composite key	there are more than one composite key	there are two or more overlapping composite keys	there are multivalued dependency between non-key attributes	**there are multivalued dependencies between attributes in composite key**
13	A 3 NF relation is split into 4 NF	by removing overlapping composite keys	by splitting into relations which do not have more than one independent multivalued dependency	removing multivalued dependency	by putting dependent non-key attribute in a separate table	**by removing overlapping composite keys**
14	A third Normal Form (3 NF) relation should	be in 2 NF	not have complete key	not be 1 NF	should not have non-key attributes depend on key attribute	**be in 2 NF**
15	The process of normalization	is automatic using a computer program	requires one to understand dependency between attributes	is manual and requires semantic information	is finding the key of a relation	**requires one to understand dependency between attributes**

16	A relation is said to be in 1NF if	there is no duplication of data	there are no composite attributes in the relation	there are only a few composite attributes	all attributes are of uniform type	**there are no composite attributes in the relation**
17	The number of normal forms which has been proposed and discussed in the book are	3	4	5	6	**6**
18	A relation which is in a higher normal form	implies that it also qualifies to be in lower normal form	does not necessarily satisfy the conditions of lower normal form	is included in the lower normal form	is independent of lower normal forms	**implies that it also qualifies to be in lower normal form**
19	Given an attribute x, another attribute y is dependent on it, if for a given x	there are many y values	there is only one value of y	there is one or more y values	there is none or one y value	**there is only one value of y**
20	Given the following relation vendor order (vendor no, order no, vendor name, qty supplied , price/unit) the second normal form relations are	vendor (vendor no, vendor name) qty (qty supplied, price/unit) order (order no, qty supplied)	vendor (vendor no, vendor name) order (order no, qty supplied, price/unit)	vendor (vendor no, vendor name) order (order no, qty supplied, price/unit) vendor order (vendor no, order no)	vendor (vendor no, vendor name, qty supplied, price/unit) vendor order (order no, vendor no)	**vendor (vendor no, vendor name) order (order no, qty supplied, price/unit) vendor order (vendor no, order no)**
21	_____technique is used to reduce the redundancy	Closure set	Decomposition	Normalization	Null Values	**Normalization**
22	3NF is also referred to as_____	LCNF	BCNF	information-preserving	desirable form	**BCNF**
23	Projection-join normal form also known as_____	1NF	2NF	3NF	5NF	**5NF**

24	if and only if the right-hand side is not a subset of the left-hand side, then functional dependency is said to be as _____	Non-trivial	Trivial	Transitive	Augmentation	**Non-trivial**
25	A relvar is in _____ if and only if the nonkey attributes are mutually independent	3NF	1NF	2NF	5NF	**3NF**
26	_____attribute does not participate in the primary key of the relvar concerned	key attribute	Non-key attribute	Variable	none	**Non-key attribute**
27	A relvar is in _____if and only if , in every legal value of that relvar, every tuple contains exactly one value for each attribute.	2NF	3NF	1NF	4NF	**1NF**
28	_____technique is used to eliminate the redundancy	Null Values	Decomposition	Normalizations	Concatenation	**Normalizations**
29	A relvar is in _____ if and only if it is in 1NF and every nonkey attribute is irreducibly dependent on the primary key.	1NF	2NF	3NF	4NF	**2NF**
30	A relvar is in _____ if and only if it is in 2NF and every nonkey attribute is non transitively dependent on the primary key.	1NF	2NF	3NF	4NF	**3NF**
31	A relvar is _____ if and only if the only determinants are candidate keys.	1NF	2NF	3NF	4NF	**2NF**

32	In the functional dependency, left-hand side indicates_____	determinants	Dependencies	trivial	non-trivial	**determinants**
33	In the functional dependency, right-hand side indicates_____	determinants	Dependencies	trivial	non-trivial	**Dependencies**
34	_____property enables us to enforce any constraint on the original relation by simply enforcing some constraints on each of the smaller relations.	dependency preservation	lossless-join	normal forms	decomposition	**dependency preservation**
35	When creating packages, the packa	declaration	initialization	specification	functions	**specification**
36	A package body contains the _____ and specifications for all of the public objects listed in the package specification	block	package	tigger	procedure	**block**
37	The normalization process is funda	OO-database model	Network model	relational database	Flat file database	**relational database**
38	_____ is simply a set of dat	Zero normal form	First normal form	Second normal form	Third normal form	**Zero normal form**
39	BCNF is a special case of _____	2NF	3NF	ZNF	1NF	**3NF**
40	In the _____ normal form, a	First	Second	Third	Fourth	**First**
41	Tables in second normal form elim	insertion anomalies	hidden dependencies	composite key	primary key	**hidden dependencies**
42	Functional dependencies are the ty	key	key revisited	superset key	primary key	**key**
43	Which is the bottom up approach t	functional dependency	database modeling	normalization	decomposition	**Normalization**
44	Which forms simplifies and ensure	1NF	2NF	3NF	4NF	**3NF**
45	Which forms has a relation that po	1NF	2NF	3NF	4NF	**4NF**
46	_____ keyword is used t	create package	begin package	new package	form package	**create package**
47	The command used to delete the pr	delete procedur	deny procedure	drop procedure	cut procedure	**drop procedure**
48	_____ is the pr	Normalization	Generalization	Specialization	decomposition	**Normalization**

49	A 3NF table which does not have r	super	primary	candidate	foreign	**candidate**
50	In first normal form each table cell	double	dependent	single	more than one	**single**
51	In first normal form record needs t	simlar	unique	dependent	derived	**unique**
52	In second normal should have sing	primary key	foreign key	composite key	candidate key	**primary key**
53	A _____ helps connect tabl	primary key	foreign key	composite key	candidate key	**foreign key**
54	Most database systems are normali	1NF	2NF	3NF	4NF	**3NF**
55	Normalization helps to have a bett	security	hidden dependencies	dependent	decomposition	**security**
56	The operations and rules for a relation is defined by	Kevin	E.F.Codd	Gerald	George	**E.F.Codd**
57	For X ® Y, if Y is a subset of X it is called as	Functional Dependency	Transitive Functional Dependency	Trivial Functional Dependency	Augmented Functional Dependency	**Trivial Functional Dependency**
58	_____ key is used to implement Entity integrity	Primary key	Foreign key	Candidate key	Composite key	**Primary key**
59	Which key is used to implement Referential integrity	Primary key	Foreign key	Candidate key	Composite key	**Foreign key**
60	If X ® YÈZ, then X ® Y and X ® Z. this property is called as	Decomposition	Reflexivity	Augmentation	Pseudotransitivity	**Decomposition**

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under Section 3 of UGC Act 1956)
Coimbatore-641 021.
THIRD SEMESTER
FIRST INTERNAL EXAMINATION – JULY 2018
RELATIONAL DATABASE MANAGEMNT SYSTEMS

Subject Code : 17ITU203 **Duration : 2 hours**
Class & Section : II BSc(IT) **Maximum Marks: 50**
Date & Session :

PART A (20*1=20 Marks)
Answer ALL the questions

1. _____is a software designed to assist in maintaining and utilizing large collections of data.

   a. Database b. DBMS c. Entities d. attributes

2. _____ data is never modified once stored in the database

   a. Dynamic b. Static c. Physical d. Logical

3. _____is a Collection of data

   a. DBMS b. Relationships c. Database d. Entities

4. _____is the task of reworking business processes in order to streamline the operations of an organization

   a. Business Rules b. Business Process Reengineering

   c. Business Modeling d. Requirement Analysis

5. Which key is used to implement Referential integrity

   a. Primary key b. candidate key c. foreign key d. composite key

6. A _____ is a well defined collection of objects

   a. Entity b. Relation c. Table d. Set

7. An object or entity is characterized by its properties also called as

   a. Entity b. Relation c. attributes d. domain

8. A model that pictorially represents a schema of a table is called _____

   a. network model b. ER Model c. Object-based model d. hierarchical

9. Graphical representation of an entity is called

a. Data flow diagram b. ER diagram c. Use case diagram d. d. Structure diagram

10. _____ operation is both associative and commutative

a. Difference b. Union c. Selection d. Projection

11. _____is the collection of schemas for the relations in the database.

a. Physical Schema b. Conceptual Schema

c. Logical Schema d. relational database

12. _____statement is used to define a new table.

a. create b.produce c. insert d. add

13. Tuples are inserted using the _____command

a. create b.make c. insert d. add

14. _____ is a single instance in an entity collection

a. Entity b. Entity set c. Entity instance d. Entity collection

15. DBA stands for _____

a. Database Administrator b. Database Assistant

c. Database Admission d. Database Access

16. _____ attribute cannot be further divided into smaller components

a. simple b. multi-valued c. stored d. single-valued

17. In _____ relationship one record in a table can be related to many records in another table

a. one-many b. one-one c. many-many d. many-one

18. What symbol is used to represent multi-valued attribute

a. double-lined ellipse b. ellipse c. dotted line d. thick line

19. Each tuple is a _____

a. Column b. row c. table d. instance

20. The language used to manipulate data in the database is called _____

a. DDL b. DML c. DAL d. DBL

PART B (3 * 2 = 6 Marks)

Answer ALL the questions

21. What are attribute and its types?

22. Write a short note on DBA?

23. Define DDL and DML.

PART C (3 * 8 = 24 Marks)

Answer ALL the questions

24. a) Enumerate in detail the major components involved in the structure of a DBMS.

(OR)

b) Differentiate between Hierarchical and Network database models.

25. a) Describe ER Model in detail with its attributes classification.

(OR)

b) Enumerate in detail various types of Relationships with examples for each.

26. a) What terms describe a Relational model? Sketch it with a neat example.

(OR)

b) Explain Relational Algebraic Operations on Set with examples.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under Section 3 of UGC Act 1956)
Coimbatore-641 021.
THIRD SEMESTER
FIRST INTERNAL EXAMINATION – July 2018
RELATIONAL DATABASE MANAGEMNT SYSTEMS

Subject Code : 17ITU303 **Duration : 2 hours**
Class & Section : II BSc(IT) **Maximum Marks: 50**
Date & Session :

PART A (20*1=20 Marks)
Answer ALL the questions

1. _____is a software designed to assist in maintaining and utilizing large collections of data.

   a. Database b. **DBMS** c. Entities d. attributes

2. _____ data is never modified once stored in the database

   a. Dynamic b. **Static** c. Physical d. Logical

3. _____is a Collection of data

   a. DBMS b. Relationships c. **Database** d. Entities

4. _____is the task of reworking business processes in order to streamline the operations of an organization

   a. Business Rules b. **Business Process Reengineering**

   c. Business Modeling d. Requirement Analysis

5. Which key is used to implement Referential integrity

   a. Primary key b. candidate key c. **foreign key** d. composite key

6. A _____ is a well defined collection of objects

   a. Entity b. Relation c. Table d**. Set**

7. An object or entity is characterized by its properties also called as

   a. Entity b. Relation c. **attributes** d. domain

8. A model that pictorially represents a schema of a table is called _____

   a. network model b**. ER Model** c. Object-based model d. hierarchical

9. Graphical representation of an entity is called

   a. Data flow diagram b. **ER diagram** c. Use case diagramd. d. Structure diagram

10. _____ operation is both associative and commutative

a. Difference b. **Union** c. Selection d. Projection

11. _____is the collection of schemas for the relations in the database.

a. Physical Schema b. Conceptual Schema

c. Logical Schema d. **relational database**

12. _____statement is used to define a new table.

a. **create** b. Produce c. insert d. add

13. Tuples are inserted using the _____command

a. create b. make c. **insert** d. add

14. _____ is a single instance in an entity collection

a. Entity b. Entity set c. **Entity instance** d. Entity collection

15. DBA stands for _____

a. **Database Administrator** b. Database Assistant

c. Database Admission d. Database Access

16. _____ attribute cannot be further divided into smaller components

a. **simple** b. multi-valued c. stored d. single-valued

17. In _____ relationship one record in a table can be related to many records in another table

a. **one-many** b. one-one c. many-many d. many-one

18. What symbol is used to represent multi-valued attribute

a. **double-lined ellipse** b. ellipse c. dotted line d. thick line

19. Each tuple is a _____

a. Column b. **row** c. table d. instance

20. The language used to manipulate data in the database is called _____

a. DDL b. **DML** c. DAL d. DBL

PART B (3 * 2 = 6 Marks)

Answer ALL the questions

21. An attribute is a sub-group of information within an entity. For example, suppose you have an entity for book titles. Within the book titles' entity, several attributes are found, such as the actual title of the book, the publisher of the book, the author, the date the book was published, and so on. Attributes are used to organize specific data within an entity.

22. 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).

23. DDL : Data Definition Language which is used to define a database values.
DML: Data Manipulation Language which is used to manipulate the database values.

PART C (3 * 8 = 24 Marks)

Answer ALL the questions

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

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.

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.

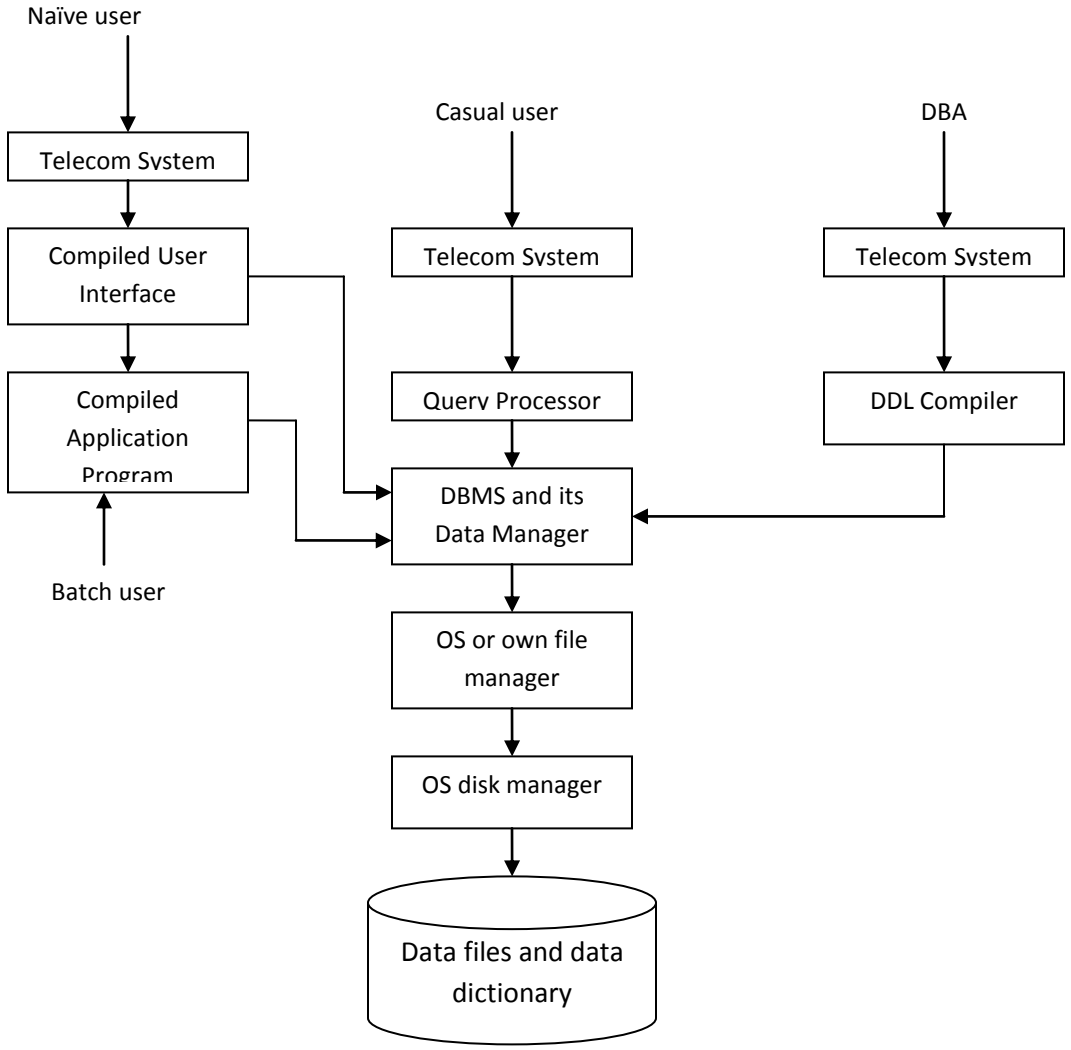
e) Query processor – It interprets online users query and convert it into an efficient series of operations in the form capable of being sent to the data manager for execution.

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.



24. b) 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.

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.

Network database models.

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.

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.

25. a) 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.

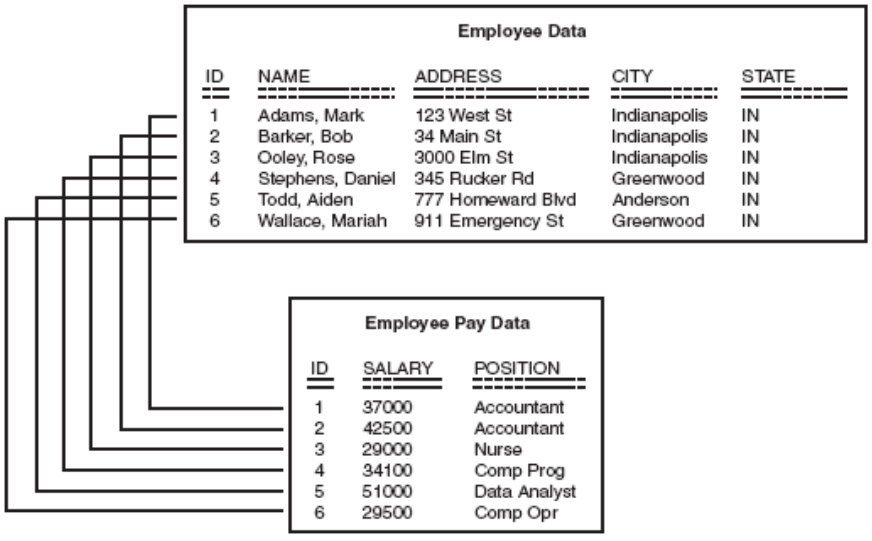
Key Attribute

- 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.

25. b) A relationship set is a grouping of all matching relationship instances

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.



Employee Data

ID	NAME	ADDRESS	CITY	STATE
1	Adams, Mark	123 West St	Indianapolis	IN
2	Barker, Bob	34 Main St	Indianapolis	IN
3	Ooley, Rose	3000 Elm St	Indianapolis	IN
4	Stephens, Daniel	345 Rucker Rd	Greenwood	IN
5	Todd, Aiden	777 Homeward Blvd	Anderson	IN
6	Wallace, Mariah	911 Emergency St	Greenwood	IN

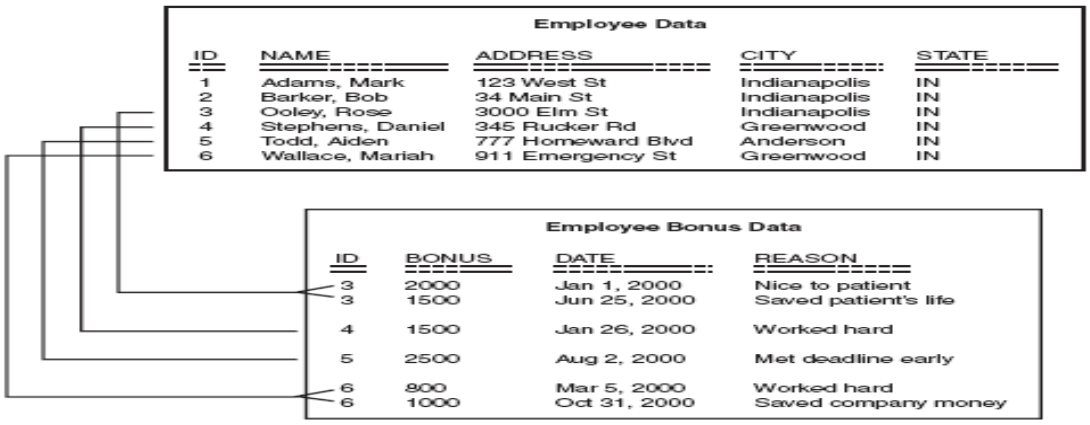
Employee Pay Data

ID	SALARY	POSITION
1	37000	Accountant
2	42500	Accountant
3	29000	Nurse
4	34100	Comp Prog
5	51000	Data Analyst
6	29500	Comp Opr

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.



Employee Data

ID	NAME	ADDRESS	CITY	STATE
1	Adams, Mark	123 West St	Indianapolis	IN
2	Barker, Bob	34 Main St	Indianapolis	IN
3	Ooley, Rose	3000 Elm St	Indianapolis	IN
4	Stephens, Daniel	345 Rucker Rd	Greenwood	IN
5	Todd, Aiden	777 Homeward Blvd	Anderson	IN
6	Wallace, Mariah	911 Emergency St	Greenwood	IN

Employee Bonus Data

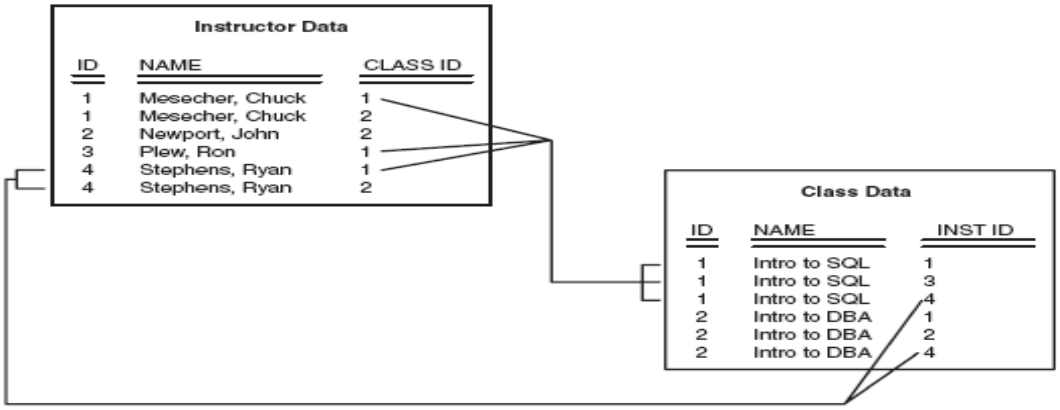
ID	BONUS	DATE	REASON
3	2000	Jan 1, 2000	Nice to patient
3	1500	Jun 25, 2000	Saved patient's life
4	1500	Jan 26, 2000	Worked hard
5	2500	Aug 2, 2000	Met deadline early
6	800	Mar 5, 2000	Worked hard
6	1000	Oct 31, 2000	Saved company money

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

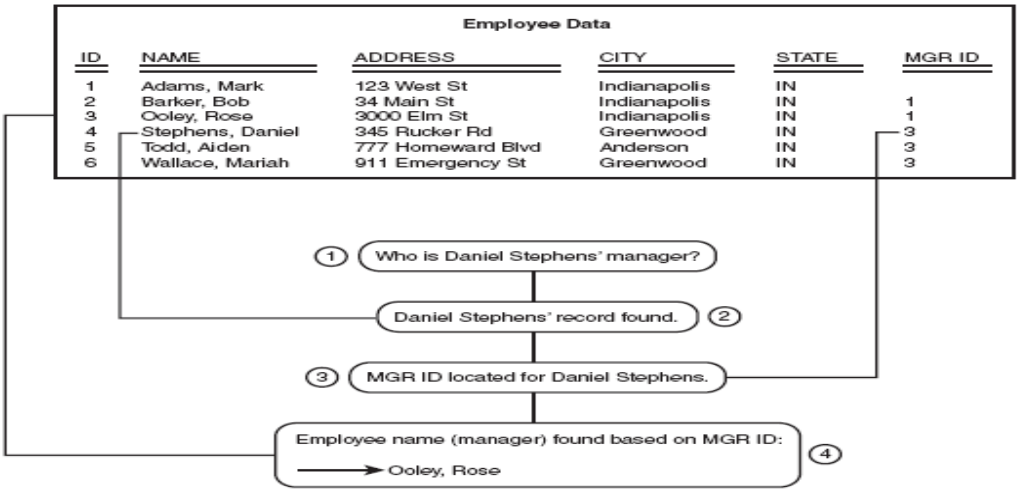


instructors who teach a particular class, you would look under Class Data.

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.



26. a) In modeling, we represent an entity set in the database by a set of its properties. Only those properties of the entity type of interest to the application are used in the model. A data model allows capturing of these properties using its data structures

The relational model like all data models, consists of three basic components:

- A set of domains and a set of relations
- Operations on relations
- Integrity rules

Eg. A sample relation **Student**

Reg_No	**Name**
14ITU001	Ramesh
14CSU001	Sunitha
14CAU004	Arun
14ITU002	Sudha
14CSU002	Ravi

A Relation with its attributes are specified as a relation scheme with the general format

Relation_Scheme_Name(Attribute_name1,Attribute_name2,…..).

For the above example it is represented as

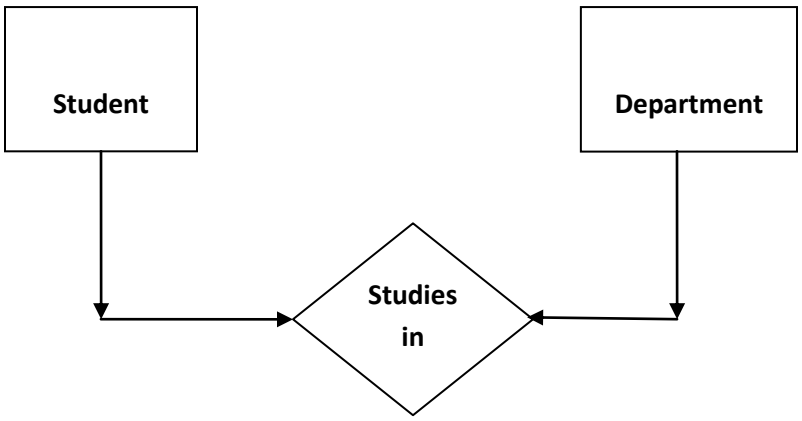
Student(Reg_No, Name)

- The “Student” relation consists of 6 tuples. So its cardinality is 6.
- The number of attributes is called degree or arity. The degree of Student relation is 2.
- Here the attribute Reg_No can be used to uniquely identify a student.

Consider there is another relation called Department with the following attribute domain

Course_code	**Course_name**
111	IT
112	CS
113	CA

A relationship can be exhibited between these two relations as follows



26. b) The basic set operations which provide a very limited data manipulation facility nave 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 (π)

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

Selection(σ)

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 σ and it is sometimes known as restriction operation.

Consider the selection operation

$$\sigma_{id<105}(Personnel)$$

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 ∞ 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 (÷)

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:

Product(Prod#, Prod_Name, Prod_details)

Developed_By(Prod#,Emp#)

Temp=Product÷ Developed_By