

---

<b>SUBJECT : DATA BASE MANAGEMENT SYSTEMS</b>	<b>SEMESTER: IV</b>	<b>L T P C</b>
<b>SUBJECT CODE: 17CSU403</b>	<b>CLASS : II B.Sc.CS -A</b>	<b>4 0 0 4</b>

---

## SCOPE

The Objective of Database Management System includes learning of relational data model, database design and Transaction model.

## OBJECTIVES

- Understand the role and nature of relational database management systems in today's IT environment.
- Translate written business requirements into conceptual entity-relationship data models.

## Course Outcome

1. Differentiate database systems from file systems by enumerating the features provided by database systems and describe each in both function and benefit.
2. Define the terminology, features, classifications, and characteristics embodied in database systems.
3. Analyze an information storage problem and derive an information model expressed in the form of an entity relation diagram and other optional analysis forms, such as a data dictionary.
4. Demonstrate an understanding of the relational data model.
5. Transform an information model into a relational database schema and to use a data definition language and/or utilities to implement the schema using a DBMS.
6. Formulate, using relational algebra, solutions to a broad range of query problems.
7. Formulate, using SQL, solutions to a broad range of query and data update problems.
8. Demonstrate an understanding of normalization theory and apply such knowledge to the normalization of a database.

## UNIT-I

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

## **UNIT-II**

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

## **UNIT-III**

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

## **UNIT-IV**

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

## **UNIT-V**

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

### **References:**

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

### **WEB SITES**

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

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

Coimbatore - 641021.

(For the candidates admitted from 2017 onwards)

**DEPARTMENT OF COMPUTER SCIENCE, CA & IT****LECTURE PLAN****STAFF NAME: K. Kathirvel & Mrs K. Banuroopa****SUBJECT NAME: DATA BASE MANAGEMENT SYSTEMS****SUB.CODE: 17CSU403****SEMESTER: I****CLASS: II B.Sc (CS)**

S.No.	Lecture Duration	Topics to be Covered	Support Materials
<b>Unit - I</b>			
1.	1	<b>Introduction to DBMS</b> Characteristics of database approach	R1:4
			R1:9
2.	1	data models	R1:30
3.	1	database system architecture and data independence	R1:33
		data independence	R1:35
4.	1	<b>Entity Relationship(ER) Modeling</b>	R1:57
		Entity types	R1:61
5.	1	relationships	R1:70
6.	1	ER Diagram	W1
		constraints.	R1:70
7.	1	Examples	W1
8.	1	Recapitulation and Discussion of important questions	
<b>Total No. of Hours Planned for Unit-I</b>			<b>8</b>

### Unit – II

1.	1	<b>Relation data model</b>	R1:145
		Relational model concepts	R1:146
2.	1	relational constraints	R1:152
3.	1	Key Constraints	R1:161
4.	1	Foreign key Constraints	R1: 272
		Relational algebra.	R1:282
5.	1	Selection and Projections	R1:182
6.	1	Set Operations, Renaming	R1:185
		,Joins and Division	R1:187
7.	1	Recapitulation and Discussion of important questions	
<b>Total No. of Hours Planned for Unit-2</b>			<b>7</b>



**Unit – III**

1.	1	<b>Relation data model:</b>	R1:245
		SQLqueries <b>Database design Insert, Delete</b>	R1:245
2.	1	Update statement in SQL, Complex Queries,	R1:255
3.	1	Triggers, views	R1:255
		schema Modification	R1:255
4.	1	ERmodel	R1:226
		EER model	R1:226
5.	1	Mapping ERmodel to relational database	R1:226
		Mapping EER model to relational database	R1:226
6.	1	functional dependencies	R1:349
		Lossless decomposition	R1:382
7.	1	Recapitulation and Discussion of important questions	
<b>Total No. of Hours Planned for Unit-III</b>			<b>7</b>

### Unit – IV

1.	1	<b>Database design</b>	R1:337
		Normal forms – Based on Primary Key	R1:370
2.	1	Second and Third Normal forms	R1:373
		Boyce –CoddNormal Form	R1:373
3.	1	<b>Transaction Processing</b> – Transaction Concepts	R1:611
4.	1	A Simple Transaction Model	R1:613
5.	1	ACID properties	R1:623
6.	1	concurrency control – lock based Protocols	R2:661
7.	1	Deadlock Handling	R2:664
8.	1	Multiple Granularity	R2:668
		Timestamp- Based Protocols	R2:686
9.	1	Recapitulation and Discussion of important questions	
<b>Total No. of Hours Planned for Unit-IV</b>			<b>9</b>

## Unit – V

1.	1	<b>File Structure and Indexing</b>	R1:465
		Operations on files	R1:481
		File of Unordered records	R1:482
2.	1	File of ordered records	R1:484
3.	1	Overview of file Organizations	R2:451
		Indexing structures for files	R1:513
4.	1	Primary index	R1:514
		secondary index	R1:514
5.	1	clustering index	R1:515
		Multilevel indexing using B trees and B+ trees	R1:527
6.	1	Recapitulation and Discussion of important questions	
7.	1	Recapitulation and Discussion of previous semester question papers	
8.	1	Recapitulation and Discussion of previous semester question papers	
9.	1	Recapitulation and Discussion of previous semester question papers	
<b>Total No. of Hours Planned for Unit-5</b>			<b>9</b>
Reference Book		R1: Elmasri, R., & Navathe, S.B. (2010). Fundamentals of Database Systems (6th ed.). New Delhi: Pearson Education,  R2: Silberschatz, A., Korth, H.F., & Sudarshan, S. (2010). Database System Concepts (6th ed.). New Delhi: McGraw-Hill	
Web Sites		W1: <a href="https://www.tutorialpoint.com">https://www.tutorialpoint.com</a> W2: <a href="https://www.w3schools.com">https://www.w3schools.com</a>	

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

### SYLLABUS

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

## INTRODUCTION

### UNDERSTANDING DATABASE FUNDAMENTALS

#### DATA

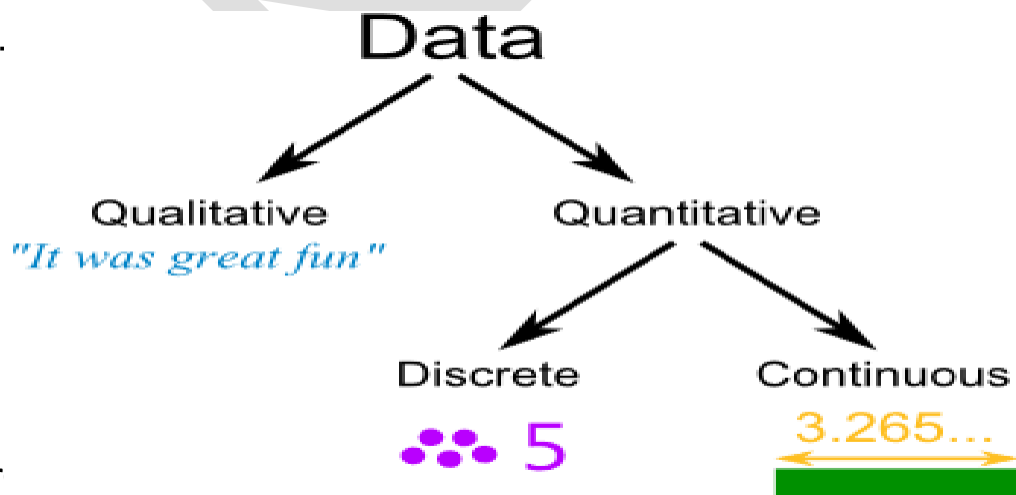
Data is a collection of facts, such as values or measurements.

It can be numbers, words, measurements, observations or even just descriptions of things.

#### Qualitative vs Quantitative

Data can be qualitative or quantitative.

- **Qualitative data** is descriptive information (it *describes* something)
- **Quantitative data**, is numerical information (numbers).



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

And **Quantitative data** can also be Discrete or Continuous:

- **Discrete data** can only take certain values (like whole numbers)
- **Continuous data** can take any value (within a range)

Put simply: **Discrete data** is counted, **Continuous data** is measured

**INFORMATION**

Information is valuable because it can affect behavior, a decision, or an outcome. For example, if a manager is told his/her company's net profit decreased in the past month, he/she may use this information as a reason to cut financial spending for the next month. A piece of information is considered valueless if, after receiving it, things remain unchanged. For a technical definition of information see information theory.

**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.

**DBMS**

A database management system (DBMS) is the software that allows a computer to perform database functions of storing, retrieving, adding, deleting and modifying data. Relational database management systems (RDBMS) implement the relational model of tables and relationships.

**Examples:**

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

Microsoft Access, MySQL, Microsoft SQL Server, Oracle and FileMaker Pro are all examples of database management systems.

The following are examples of database applications:

- computerized library systems
- automated teller machines
- flight reservation systems
- computerized parts inventory systems

**RDBMS**

Short for *relational database management system* and pronounced as separate letters, a type of database management system (DBMS) that stores data in the form of related tables. Relational databases are powerful because they require few assumptions about how data is related or how it will be extracted from the database. As a result, the same database can be viewed in many different ways.

An important feature of relational systems is that a single database can be spread across several tables. This differs from flat-file databases, in which each database is self-contained in a single table.

**SQL**

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database. SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server use SQL as standard database language.

Also, they are using different dialects, such as:

---

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**

**COURSE NAME: DATABASE MANAGEMENT  
SYSTEM**

**COURSE CODE: 17CSU403**

**UNIT: I**

**BATCH-2017-2020**

---

KAHE

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

MS SQL Server using T-SQL,

- Oracle using PL/SQL,

**History:**

- **1970** -- Dr. E. F. "Ted" of IBM is known as the father of relational databases. He described a relational model for databases.
- **1974** -- Structured Query Language appeared.
- **1978** -- IBM worked to develop Codd's ideas and released a product named System/R.
- **1986** -- IBM developed the first prototype of relational database and standardized by ANSI. The first relational database was released by Relational Software and its later becoming Oracle.

SQL is pronounced as "S-Q-L" or "see-quill".

SQL uses -- character sequence as a single line comment identifier.

SQL commands are not case sensitive and the following SQL queries are equivalent:

SELECT \* FROM Users

select \* from Users

**Characteristics of Database Approach****1. Represent Some Aspects of real world applications**

A database represents some features of real world applications. Any change in the real world is reflected in the database. If we have some changes in our real applications like railway reservation system then it will be reflected in database too.



**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

**For example**, let us take railway reservation system; we have in our mind some certain applications of maintaining records of attendance, waiting list, train arrival and departure time, certain day etc. related to each train.

**2. Manages Information**

A database always takes care of its information because information is always helpful for whatever work we do. It manages all the information that is required to us. By managing information using a database, we become more deliberated user of our data.

Also See: What is Database?

**3. Easy Operation implementation**

All the operations like insert, delete, update, search etc. are carried out in a flexible and easy way. Database makes it very simple to implement these operations. A user with little knowledge can perform these operations. This characteristic of database makes it more powerful.

**4. Multiple views of database**

Basically, a view is a **subset of the database**. A view is defined and devoted for a particular user of the system. Different users of the system may have different views of the same system.

Every view contains only the data of interest to a user or a group of users. It is the responsibility of users to be aware of how and where the data of their interest is stored.

**5. Data for specific purpose**

A database is designed for data of specific purpose. **For example**, a database of student management system is designed to maintain the record of student's marks, fees and attendance etc. This data has a specific purpose of maintaining student record.

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

Also See: Advantages Of Database Management System

**6. It has Users of Specific interest**

A database always has some indented group of users and applications in which these user groups are interested.

**For example**, in a library system, there are three users, official administration of the college, the librarian, and the students.

**7. Self Describing nature**

A database is of self describing nature; it always describes and narrates itself. It contains the description of the whole data structure, the constraints and the variables.

It makes it different from traditional file management system in which definition was not the part of application program. These definitions are used by the users and DBMS software when needed.

**8. Logical relationship between records and data**

A database gives a logical relationship between its records and data. So a user can access various records depending upon the logical conditions by a single query from the database.

**Data Model**

A database model shows the logical structure of a database, including the relationships and constraints that determine how data can be stored and accessed. Individual database models are designed based on the rules and concepts of whichever broader data model the designers adopt. Most data models can be represented by an accompanying database diagram.

**Types of database models**

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

There are many kinds of data models. Some of the most common ones include:

- Hierarchical database model
- Relational model
- Network model
- Object-oriented database model
- Entity-relationship model
- Document model
- Entity-attribute-value model
- Star schema
- The object-relational model, which combines the two that make up its name

**Flat Data Model**

Flat data model is the first and foremost introduced model and in this all the data used is kept in the same plane. Since it was used earlier this model was not so scientific.

Roll No	Name	Course
5482	Mark	Web Designing
5486	Steve	Java
5496	Smith	Oracle

**Entity Relationship Data Model**

Entity relationship model is based on the notion of the real world entities and their relationships. While formulating the real world scenario in to the database model an entity set is created and this model is dependent on two vital things and they are :

- Entity and their attributes
- Relationships among entities

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

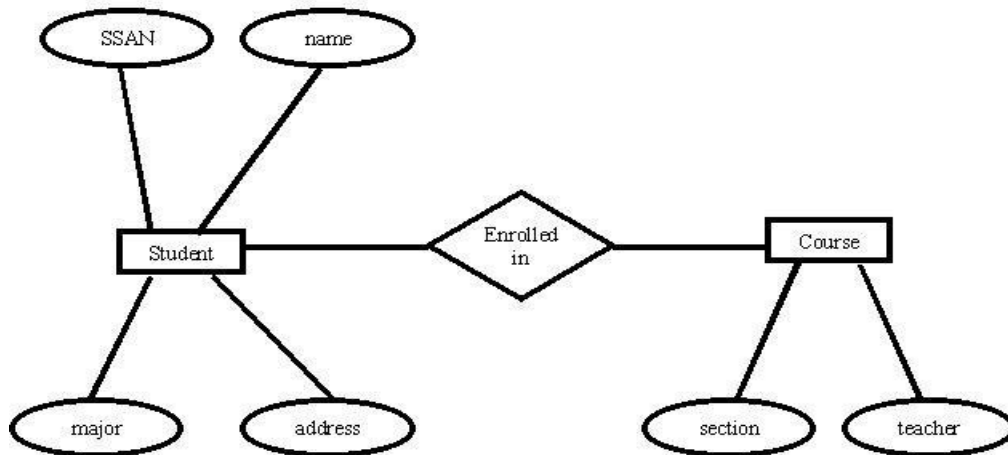
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020



An entity has a real world property called attribute and attribute define by a set of values called domain. For example, in a university a student is an entity, university is the database, name and age and sex are the attributes. The relationships among entities define the logical association between entities.

**Relational Data Model**

Relational model is the most popular model and the most extensively used model. In this model the data can be stored in the tables and this storing is called as relation, the relations can be normalized and the normalized relation values are called atomic values. Each row in a relation contains unique value and it is called as tuple, each column contains value from same domain and it is called as attribute.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

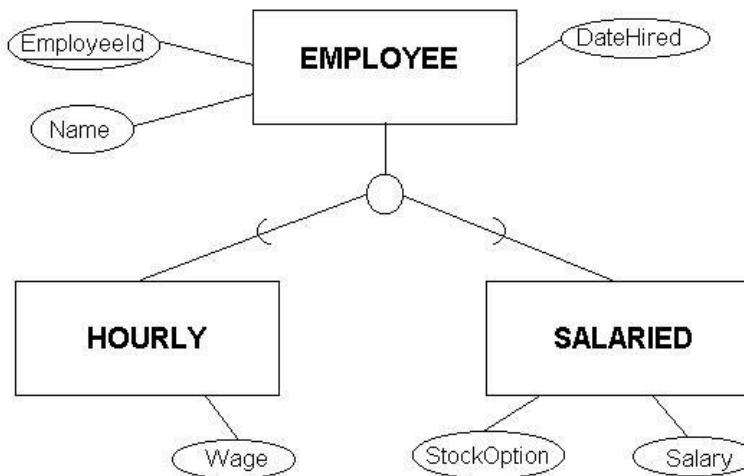
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

**Network Data Model**

Network model has the entities which are organized in a graphical representation and some entities in the graph can be accessed through several paths.

**Hierarchical Data Model**

Hierarchical model has one parent entity with several children entity but at the top we should have only one entity called root. For example, department is the parent entity called root and it has several children entities like students, professors and many more.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

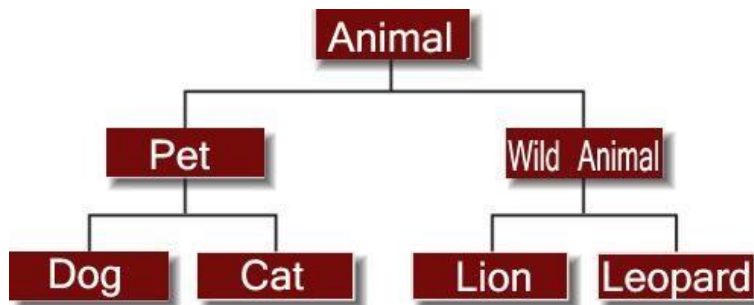
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

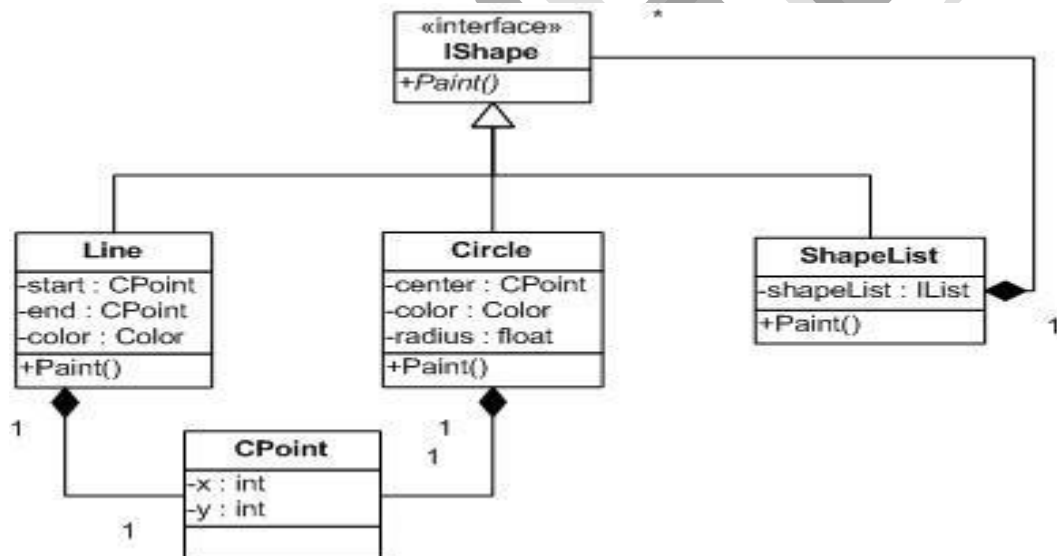
COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

**Object oriented Data Model**

Object oriented data model is one of the developed data model and this can hold the audio, video and graphic files. These consist of data piece and the methods which are the DBMS instructions.

**Record base Data Model**

Record base model is used to specify the overall structure of the database and in this there are many record types. Each record type has fixed no. of fields having the fixed length.

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020****Object relation Data Model**

Object relation model is a very powerful model but coming to it's design it is quiet complex. This complexity is not problem because it gives efficient results and widespread with huge applications. It has a feature which allows working with other models like working with the very known relation model.

**Semi structured Data Model**

Semi structured data model is a self describing data model, in this the information that is normally associated with a scheme is contained within the data and this property is called as the self describing property.

**Associative Data Model**

Associative model has a division property, this divides the real world things about which data is to be recorded in two sorts i.e. between entities and associations. Thus, this model does the division for dividing the real world data to the entities and associations.

**Database Architecture****Three Level Architecture of DBMS**

**Following are the three levels of database architecture,**

1. Physical Level
2. Conceptual Level
3. External Level

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

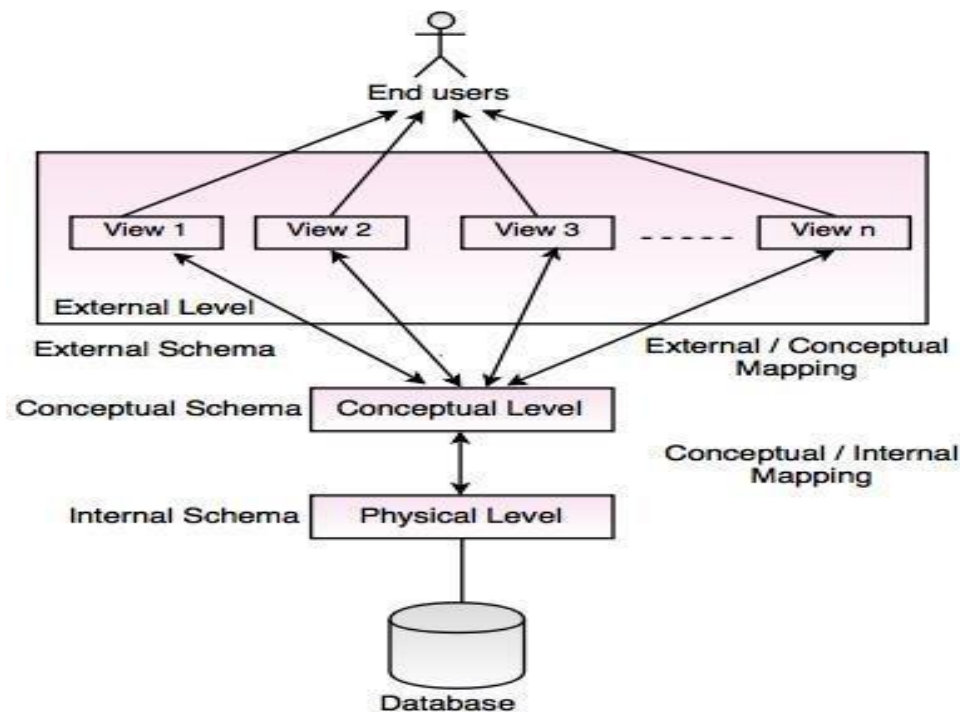


Fig. Three Level Architecture of DBMS

In the above diagram,

- It shows the architecture of DBMS.
- Mapping is the process of transforming request response between various database levels of architecture.
- Mapping is not good for small database, because it takes more time.
- In External / Conceptual mapping, DBMS transforms a request on an external schema against the conceptual schema.
- In Conceptual / Internal mapping, it is necessary to transform the request from the conceptual to internal levels.

### 1. Physical Level

- Physical level describes the physical storage structure of data in database.
- It is also known as Internal Level.
- This level is very close to physical storage of data.



**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

- At lowest level, it is stored in the form of bits with the physical addresses on the secondary storage device.
- At highest level, it can be viewed in the form of files.
- The internal schema defines the various stored data types. It uses a physical data model.

**2. Conceptual Level**

- Conceptual level describes the structure of the whole database for a group of users.
- It is also called as the data model.
- Conceptual schema is a representation of the entire content of the database.
- These schema contains all the information to build relevant external records.
- It hides the internal details of physical storage.

**3. External Level**

- External level is related to the data which is viewed by individual end users.
- This level includes a no. of user views or external schemas.
- This level is closest to the user.
- External view describes the segment of the database that is required for a particular user group and hides the rest of the database from that user group.

**Data Abstraction and Data Independence**

Database systems comprise of complex data-structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hide irrelevant details from the users. This approach simplifies database design.

There are mainly **3** levels of data abstraction:

**Physical:** This is the lowest level of data abstraction. It tells us how the data is actually stored in memory. The access methods like sequential or random access and file organisation methods like B+ trees, hashing used for the same. Usability, size of

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

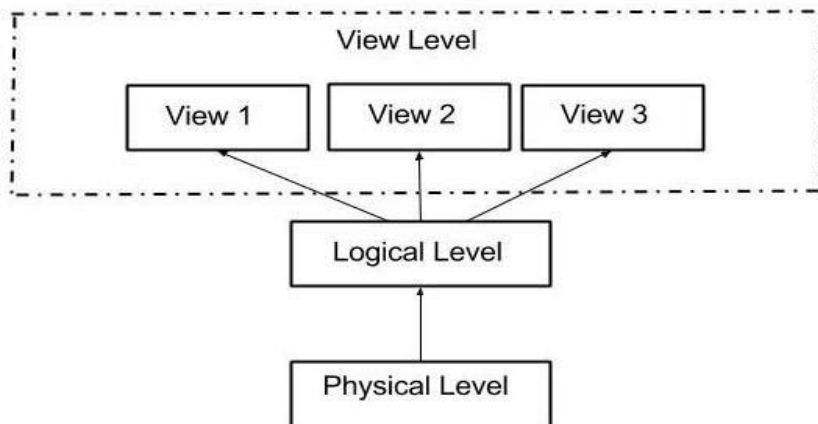
memory, and the number of times the records are factors which we need to know while designing the database.

Suppose we need to store the details of an employee. Blocks of storage and the amount of memory used for these purposes is kept hidden from the user.

**Logical:** This level comprises of the information that is actually stored in the database in the form of tables. It also stores the relationship among the data entities in relatively simple structures. At this level, the information available to the user at the view level is unknown.

We can store the various attributes of an employee and relationships, e.g. with the manager can also be stored.

**View:** This is the highest level of abstraction. Only a part of the actual database is viewed by the users. This level exists to ease the accessibility of the database by an individual user. Users view data in the form of rows and columns. Tables and relations are used to store data. Multiple views of the same database may exist. Users can just view the data and interact with the database, storage and implementation details are hidden from them.



The main purpose of data abstraction is achieving data independence in order to save time and cost required when the database is modified or altered. We have namely two levels of data independence arising from these levels of abstraction :

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020****Physical level data independence :**

It refers to the characteristic of being able to modify the physical schema without any alterations to the conceptual or logical schema, done for optimisation purposes, e.g., Conceptual structure of the database would not be affected by any change in storage size of the database system server. Changing from sequential to random access files is one such example. These alterations or modifications to the physical structure may include:

- Utilising new storage devices.
- Modifying data structures used for storage.
- Altering indexes or using alternative file organisation techniques etc.

**Logical level data independence:**

It refers characteristic of being able to modify the logical schema without affecting the external schema or application program. The user view of the data would not be affected by any changes to the conceptual view of the data. These changes may include insertion or deletion of attributes, altering table structures entities or relationships to the logical schema etc.

**ENTITY RELATIONSHIP (ER) MODELING****Introduction to ER Model**

ER Model is a high-level data model, developed by Chen in 1976. This model defines the data elements and relationships for a specified system. It is useful in developing a conceptual design for the database & is very simple and easy to design logical view of data.

**Importance of ER Model**

- ER Model is plain and simple for designing the structure.
- It saves time.
- Without ER diagrams you cannot make a database structure & write production code.

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**


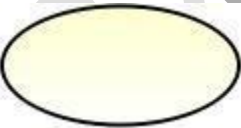
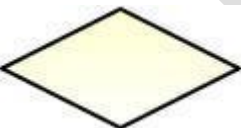

- It displays the clear picture of the database structure.

**ER Diagrams**

- ERD stands for Entity Relationship diagram.
- It is a graphical representation of an information system.
- ER diagram shows the relationship between objects, places, people, events etc. within that system.

- It is a data modeling technique which helps in defining the business process.
- It used for solving the design problems.

Following are the components of ER Diagram,

Notations	Representation	Description
	Rectangle	It represents the Entity.
	Ellipse	It represents the Attribute.
	Diamond	It represents the Relationship.
	Line	It represents the link between attribute and entity set to relationship set.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

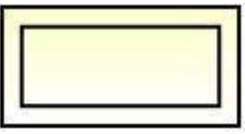
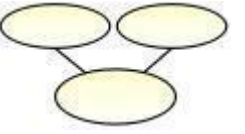
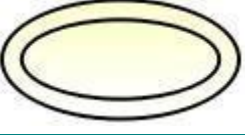

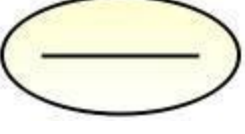
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

	Double Rectangle	It represents the weak entity.
	Composite Attribute	It represents composite attribute which can be divided into subparts. For eg. Name can be divided into First Name and Last Name
	Multi valued Attribute	It represents multi valued attribute which can have many values for a particular entity. For eg. Mobile Number.
	Derived Attribute	It represents the derived attribute which can be derived from the value of related attribute.
	Key Attribute	It represents key attribute of an entity which have a unique value in a table. For eg. Employee → EmpId (Employee Id is Unique).

**Entity, Entity Type, Entity Set**

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as:

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

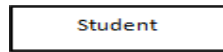
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

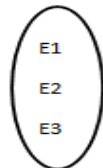
COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020



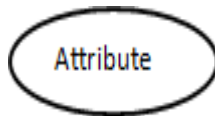
Entity Type



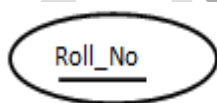
Entity Set

**Attribute**

Attributes are the **properties which define the entity type**. For example, Roll\_No, Name, DOB, Age, Address, Mobile\_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

**Key Attribute**

The attribute which **uniquely identifies each entity** in the entity set is called key attribute. For example, Roll\_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.

**Composite Attribute**

An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

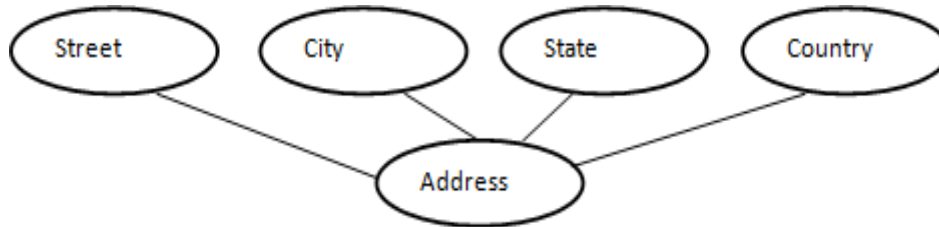
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

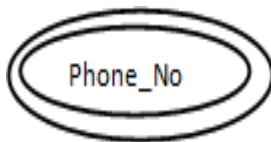
COURSE CODE: 17CSU403

UNIT: I

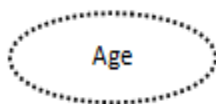
BATCH-2017-2020

**Multivalued Attribute**

An attribute consisting **more than one value** for a given entity. For example, Phone\_No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.

**Derived Attribute**

An attribute which can be **derived from other attributes** of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.



The complete entity type **Student** with its attributes can be represented as:

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

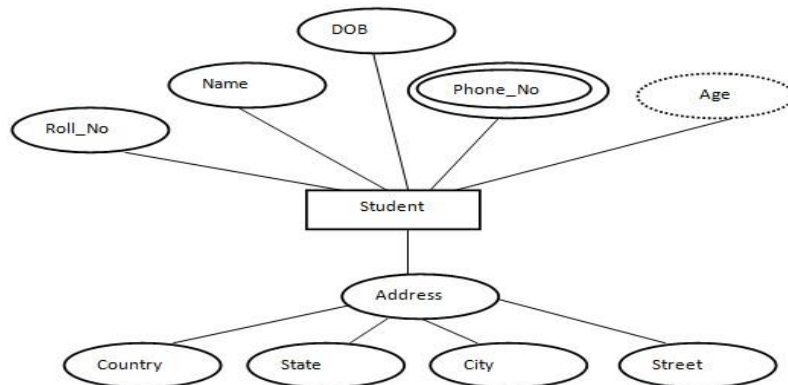
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

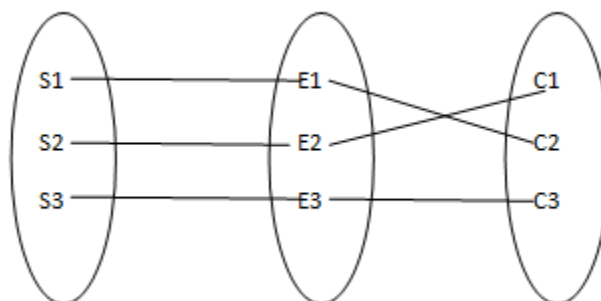
BATCH-2017-2020

**Relationship Type and Relationship Set**

A relationship type represents the **association between entity types**. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.



A set of relationships of same type is known as relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.

**Degree of a relationship set**

The number of different entity sets **participating in a relationship** set is called as degree of a relationship set.



---

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**

**COURSE NAME: DATABASE MANAGEMENT  
SYSTEM**

**COURSE CODE: 17CSU403**

**UNIT: I**

**BATCH-2017-2020**

---

KAHE

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

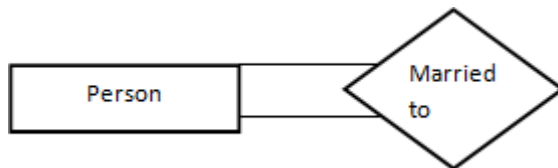
COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

**Unary Relationship**

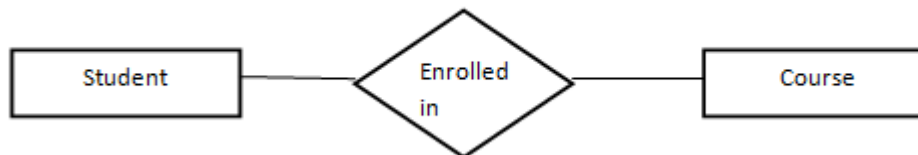
When there is **only ONE entity set participating in a relation**, the relationship is called



as unary relationship. For example, one person is married to only one person.

**Binary Relationship**

When there are **TWO entities set participating in a relation**, the relationship is called as binary relationship. For example, Student is enrolled in Course.

**n-ary Relationship**

When there are **n entities set participating in a relation**, the relationship is called as n-ary relationship.

**Weak Entity Type and Identifying Relationship**

As discussed before, an entity type has a key attribute which uniquely identifies each entity in the entity set. But there exists **some entity type for which key attribute can't be defined**. These are called Weak Entity type.

For example, A company may store the information of dependants (Parents, Children, Spouse) of an Employee. But the dependents don't have existence without the employee. So Dependent will be weak entity type and Employee will be Identifying Entity type for Dependant.

A weak entity type is represented by a double rectangle. The participation of weak entity type is always total. The relationship between weak entity type and its identifying

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

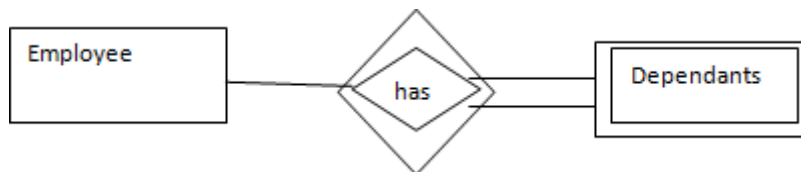
COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

strong entity type is called identifying relationship and it is represented by double diamond.

**Relationship Mapping in ER Diagram of Databases****Cardinality**

The **number of times an entity of an entity set participates in a relationship** set is known as cardinality.

Cardinality can be of different types:

**Following are the types of Relationship Mapping,**

1. One - to - One Relationship
2. One - to - Many Relationship
3. Many - to - One Relationship
4. Many - to - Many Relationship

**1. One - to - One Relationship**

- In One - to - One Relationship, one entity is related with only one other entity.
- One row in a table is linked with only one row in another table and vice versa.

**For example:** A Country can have only one Capital City.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**

**COURSE NAME: DATABASE MANAGEMENT  
SYSTEM**

**COURSE CODE: 17CSU403**

**UNIT: I**

**BATCH-2017-2020**

KAHE

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

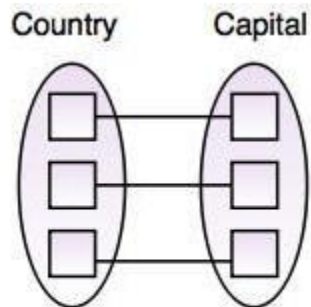


Fig. One to One Mapping

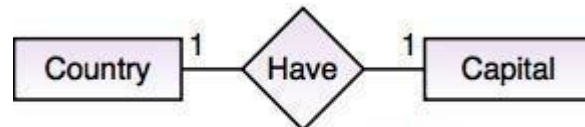
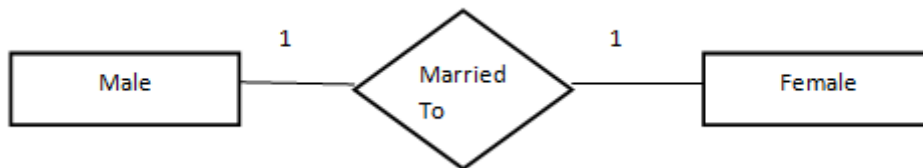


Fig. Representation in ER Diagram

**2. One - to - Many Relationship**

- In One - to - Many Relationship, one entity is related to many other entities.
- One row in a table A is linked to many rows in a table B, but one row in a table B is linked to only one row in table A.

**For example:** One Department has many Employees.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

Department Employee

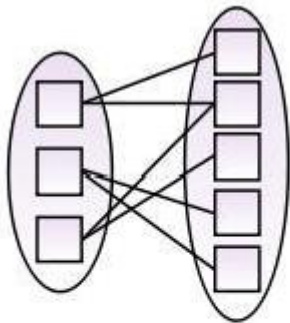


Fig. One to Many Mapping

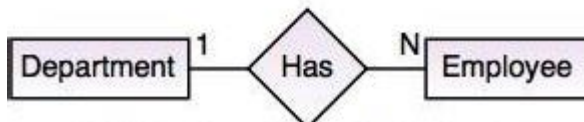


Fig. Representation in ER Diagram

**3. Many - to - One Relationship**

- In Many - to - One Relationship, many entities can be related with only one other entity.

**For example:** No. of Employee works for Department.

- Multiple rows in Employee table is related with only one row in Department table.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

Employee Department

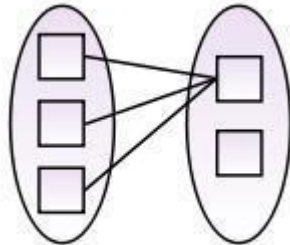


Fig. Many to One Mapping



Fig. Representation in ER Diagram

**4. Many - to - Many Relationship**

- In Many - to - Many Relationship, many entities are related with the multiple other entities.
- This relationship is a type of cardinality which refers the relation between two entities.

**For example:** Various Books in a Library are issued by many Students.

Book Student

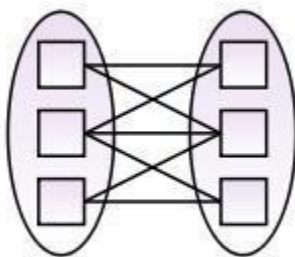


Fig. Many to Many Mapping

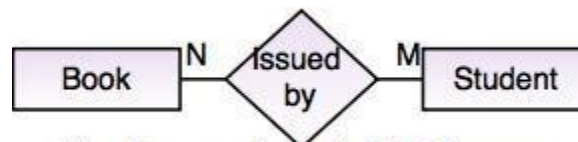



Fig. Representation in ER Diagram

**Participation Constraint**

Participation Constraint is applied on the entity participating in the relationship set.

**Total Participation:**

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

- Each entity in the entity set **must participate** in the relationship. If each student must enroll in a course.
- In Total Participation, every entity in the set is involved in some association of the relationship.
- It is indicated by a double line (  ) between entity and relationship.

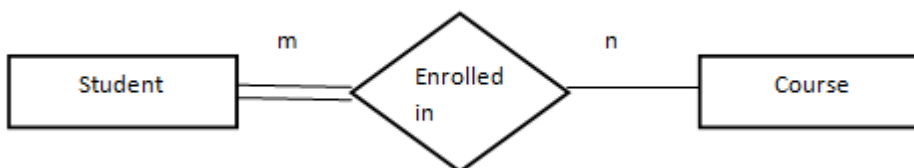
**For example:** Every Department must have a Manager.




Fig. Total Participation

**Partial Participation:**

- The entity in the entity set **may or may NOT participate** in the relationship.
- If some courses are not enrolled by any of the student, the participation of course will be partial.
- The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.



- In Partial Participation, not all entities in the set are involved in association of the relationship.
- It is indicated by a single line (  ) between entity and relationship.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020



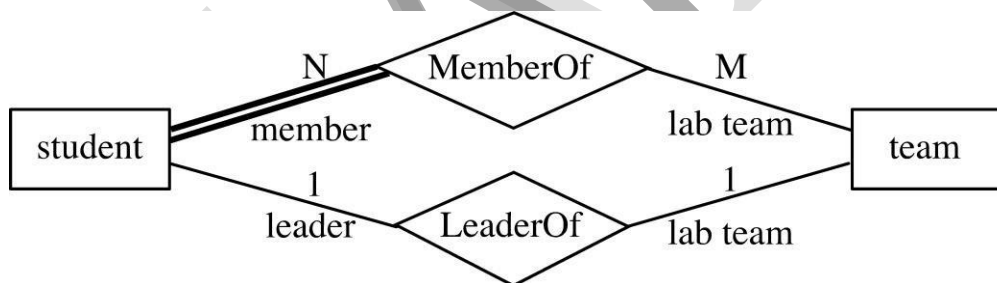
Fig. Partial Participation

Example : Participation Constraints

Every student must be a member of a team, or, in other words, a student entity is of interest only if it participates in a *MemberOf* relationship. Thus, we can include in an ER diagram a **participation constraint** in which participation

of *student* in *MemberOf* is **total**. A double line indicates the total participation constraint in an ER model

of *student* in *LeaderOf* is **partial**, because a student might be a team leader.

**Figure 17.** ER diagram notation for total participation constraint

Using the above components,

**Figure 18.** ER diagram notation for total participation constraint

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

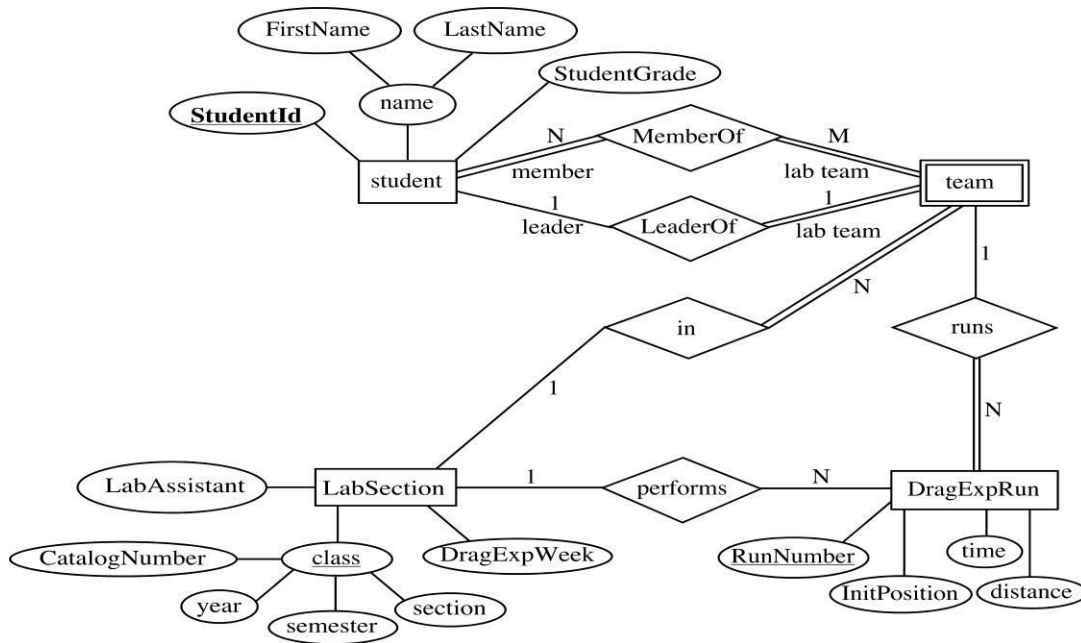
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

**Entity Relationship (ER) Diagram – Relationship****Example**

1) When there is One to Many cardinality in ER diagram.  
For example, a student can be enrolled only in one course, but a course can be enrolled by many students

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

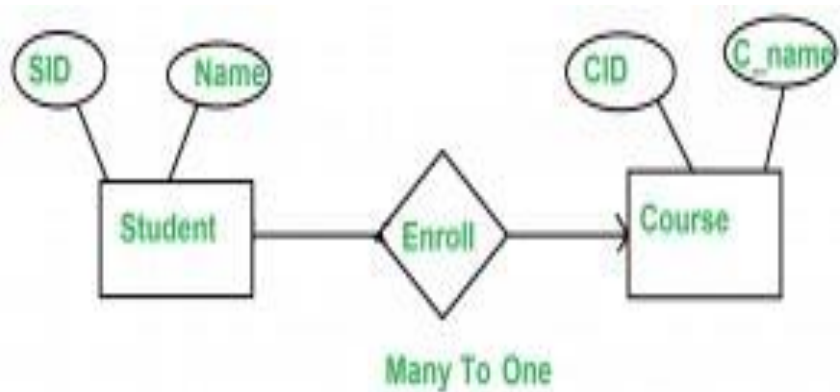
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020



For Student(SID, Name), SID is the primary key. For Course ( CID, C\_name ), CID is the primary key

Student		Course	
(SID Name)		( CID C_name )	
1	A	c1	Z
2	B	c2	Y
3	C		
4	D		
(SID CID)			
1	C1		
2	C1		
3			

Now the question is, what should be the primary key for Enroll SID or CID or combined. We can't have CID as primary key as you can see in enroll for the same CID we have multiples SID. (SID , CID) can distinguish table uniquely, but it is not minimum. So SID is the primary key for the relation enroll.

For above ER diagram, we considered three tables in database

Student

Enroll

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020

Course

But we can combine Student and Enroll table renamed as Student\_enroll.

Student\_Enroll  
( SID Name CID )

1	A	c1
2	B	c1
3	C	c3
4	D	c2

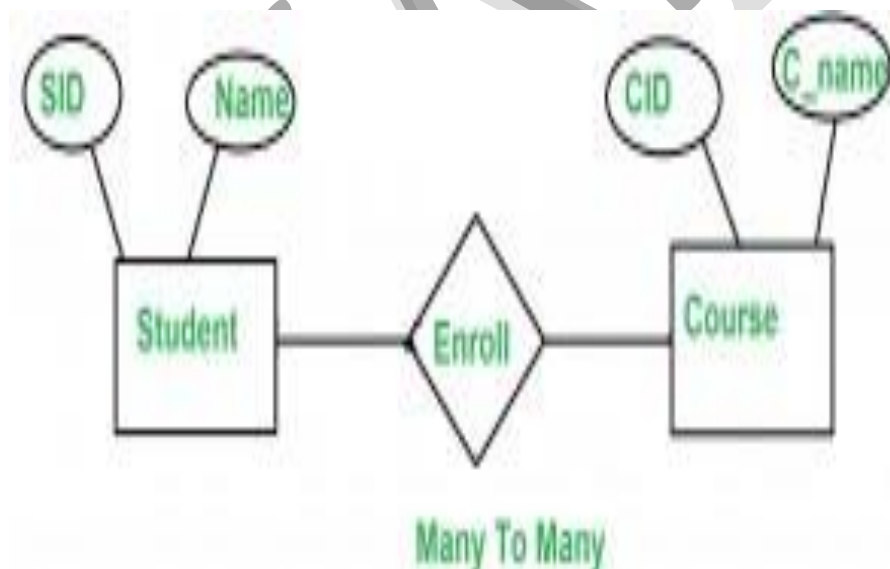
Student and enroll tables are merged now .

So require minimum two DBMS tables for Student\_enroll and Course.

**Note:** In One to Many relationship we can have minimum two tables.

## 2. When there is Many to Many cardinality in ER Diagram.

Let us consider above example with the change that now student can also enroll more than 1 course.



Student

Course

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT  
SYSTEM****COURSE CODE: 17CSU403****UNIT: I****BATCH-2017-2020**

-----		-----	
1	A	c1	Z
2	B	c2	Y
	C		
Enroll			
( SID CID )			
-----			
1	C1		
1	C2		
2	C1		
3	C2		

Now, same question what is the primary key of Enroll relation, if we carefully analyse the Enroll primary key for Enroll table is ( SID , CID ).

But in this case we can't merge Enroll table with any one of Student and Course. If we try to merge Enroll with any one of the Student and Course it will create redundant data.

**Note:** Minimum three tables are required in Many to Many relationship.

### 3. One to One Relationship

There are two possibilities

**A) If we have One to One relationship and we have total participation at at-least one end.**

For example, consider the below ER diagram.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

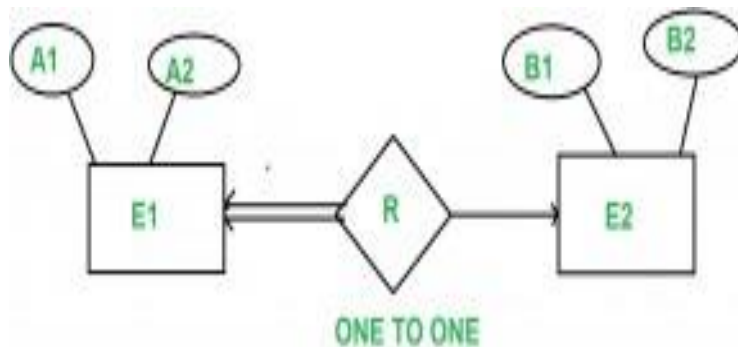
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT  
SYSTEM

COURSE CODE: 17CSU403

UNIT: I

BATCH-2017-2020



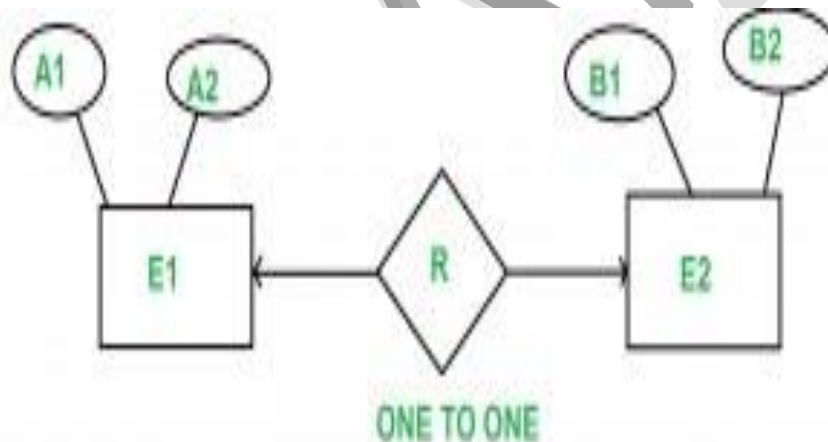
A1 and B1 are primary keys of E1 and E2 respectively.

In the above Diagram we have total participation at E1 end.

Only the primary key of E1, which is in total participation should be allowed as the primary key of the reduced table, since if the primary key of E2 is used, it might have null values for many of its entries, since its participation is only partial and may not have corresponding entries for all its values.

**Note** – Only one table required.

**B) One to One relationship with no total participation.**



A1 and B1 are primary keys of E1 and E2 respectively.

Primary key of R can be A1 or B1, but we can't still combine all the three table into one. if we do, so some entries in combined table may have NULL entries. So idea of merging all three table into one is not good.

But we can merge R into E1 or E2. So minimum 2 tables are required.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**

**COURSE NAME: DATABASE MANAGEMENT  
SYSTEM**

**COURSE CODE: 17CSU403**

**UNIT: I**

**BATCH-2017-2020**

**POSSIBLE QUESTIONS**

**2 MARK QUESTIONS**

1. List the types of database models.
2. List the data types used in DBMS.
3. What is the difference between char and varchar2?
4. Mention the three levels of database schema.
5. Define primary key with example.

**8 MARK QUESTIONS**

1. Define database and explain various database models with neat diagram.
2. Draw and explain data base architecture.
3. What are the typical ERD symbols and draw a neat ERD for an employee table.  
b) Explain i) Entity types ii) Relationships
4. Discuss i) Data Independence ii) Constraints
5. What are the different data models present and explain briefly?

**UNIT-I**

sno	QUESTIONS	opt1	opt2	opt3	opt4	Answer
1	_____ was adopted by the ANSI and ISO.	PSQL	SQL	R-SQL	Sequel	SQL
2	_____ is a collection of high-level data description constructs that hide many low-level storage details	Data Model	ER Model	Network Model	none	Data Model
3	Database management System based on _____	Network model	Hierarchical model	Relational model	Object-based model	Relational model
4	A widely used Semantic model called _____	Network model	ER Model	Object-based model	Hierarchical model	ER Model
5	_____ is a more abstract	ER model	Semantic data model	Conceptual data Model	Physical data model	Semantic data model
6	_____ model is used to pictorially denote entities & relationships	Physical data model	ER model	network model	structure chart	ER model
7	A description Of data in terms of a data model is called _____	Schema	relation	record	entities	Schema
8	Field is otherwise known as _____	Column	Entity	Relationship	Relation	Column
9	Column is otherwise known as _____	Entity	Relationship	Relation	attribute	attribute
10	_____ is a software designed to assist in maintaining and utilizing large collections of data.	Database	DBMS	Entities	attributes.	DBMS
11	_____ model used Object store & versant.	Network Model	Hierarchical Model	Object Oriented Model	Record based Model	Object Oriented Model
12	_____ is used to define the external and conceptual model	DDL	DML	DCL	TCL	DDL
13	Conceptual model otherwise called as _____	Physical Schema	Internal Schema	Logical Schema	relations	Logical Schema



14	Physical model specifies _____ details	Information	data	Storage	relationships	Storage
15	The _____ enviroment involves dumb terminals	mainframe	client/server	internet computing	LAN	mainframe
16	A host computer in internet computing eniroment is called _	server	data server	PC	web server	web server
17	_____ is the primary unit of storage in a database	table	column	row	number	table
18	database design involves conversion of _____ to stuctured database model.	business process	business model	entity	relationships	business model
19	The architecture of a hierarchical database is based on _____ the concept of relationships.	set structure	tree/node	parent/child	server/client	parent/child
20	The relationship between tables in the network model is called a	parent/child	set structure	client/server	tree/node	set structure
21	Set structures can represent a __ relationship between tables	one-to-one	one-to-many	many-to-many	many-to-one	one-to-many
22	SQL has been developed and used for _____ model	relational	Hierarchical	network	flat file	relational
23	A class is the equivalent of a _____ in a relational database	row	column	table	primary key	table
24	SQL3 is also referred to as	SQL97	SQL98	SQL99	SQL100	SQL99
25	_____ is the process of creating an interface for the end user through which the database can be accessed	dabase design	business model	interface design	Application design	Application design
26	The process of reducing data redundancy in a relational database is called	data security	data accuracy	data protection	normalization	normalization
27	Static, or _____ data is seldom or never modified once stored in the database.	dynamic	historic	information	transactional	historic
28	_____ or transactional data, is data that is frequently modified once stored in the database.	dynamic	historic	information	transactional	dynamic
29	BPR is	Business product re-engineering	Business product repair	Business process re-engineering	Business procedure re-engineering	Business process re-engineering

30	____ is the process of ensuring that data is consistent between related tables	primary key	database security	performance	Referential integrity	Referential integrity
31	Foreign keys are defined in ____ tables	parent	child	one	database	child
32	_____ is an object in the real world	Entity	Attribute	Relationship	Property	Entity
33	in database model the data is stored in objects	hierarchical	network	relational	object_oriented	object_oriented
34	In relational model the data is stored in ____	table	files	objects	sets	table
35	Information about the conceptual, external and physical schemas is stored in _____	Directory	System Catalogs	IMS	Information System	System Catalogs
36	Conceptual schema otherwise called as _____	Physical Schema	Internal Schema	Logical Schema	relations	Logical Schema
37	Physical Schema specifies _____ details	Information	data	Storage	relationships	Storage
38	_____ is used to speed up data retrieval operations.	DML Operations	Select Operation	Indexes	select operation with where condition	Indexes
39	A Structure of database using the given data model is called a _____	Database	Relation	Schema	design	Schema
40	SQL was developed as an integral part of	A hierarchical database	A relational database	A OO database	A network database	A relational database
41	Which of the following is CORRECT about database management system's languages?	Data definition languages are used to specify	Data manipulation languages are	Data manipulation languages are used	Data definition languages are only used to update data in the DBMS	Data manipulation languages are used
42	An E-R modelling for given application leads to	conceptual data model	logical data model	external data model	internal data model	conceptual data model
43	A conceptual data model is converted using a Relational Data Base Management System to a	logical data model	external data model	internal data model	an entity-relation data model	logical data model
44	A subset of logical data model accessed by programmers is called a	conceptual data model	external data model	internal data model	an entity-relation data model	external data model
45	When a logical model is mapped into a physical storage such as a disk store the resultant data model is known as	conceptual data model	external data model	internal data model	disk data model	internal data model

46	By data integrity we mean	maintaining consistent data values	integrated data values	banning improper access to data	not leaking data values	maintaining consistent data values
47	Data integrity is ensured by	good data editing	propagating data changes to all data items	preventing unauthorized access	preventing data duplication	propagating data changes to all data
48	By data security in DBMS we mean	preventing access to data	allowing access to data only to authorized users	preventing changing data	introducing integrity constraints	allowing access to data only to
49	By redundancy in a file based system we mean that	unnecessary data is stored	same data is duplicated in many files	data is unavailable	files have redundant data	same data is duplicated in many
50	Data integrity in a file based system may be lost because	the same variable may have different	files are duplicated	unnecessary data is stored in files	redundant data is stored in files	the same variable may have different
51	Data availability is often difficult in file based system	as files are duplicated	as unnecessary data are stored in files	as one has to search different files and these	redundant data are stored in files	as one has to search different files and
52	An entity is	an inanimate object in an application	a collection of items in an application	a data structure	a distinct real world item in an application	a distinct real world item in an application
53	A relationship is	an item in an application	a meaningful dependency between entities	a collection of related entities	related data	a meaningful dependency between entities
54	Pick the relationship from the following:	a classroom	teacher	attends	cost per dozen	attends
55	Pick the meaningful relationship between entities	vendor supplies goods	vendor talks with customers	vendor complains to vendor	vendor asks prices	vendor supplies goods
56	The entity set is a	set of entities	collection of different entities	collection of related entities	collection of similar entities	collection of similar entities
57	Pick entity set from the following	all vendors supplying to an organization	vendors and organizations they supply	vendors and transporters	a vendor supplying to many organizations	all vendors supplying to an organization
58	The expansion of E-R diagram is	Entity-Relationship diagram	Entity-Relative diagram	Entity-Relation diagram	Entity-Rationalized diagram	Entity-Relationship diagram
59	In an E-R diagram entities are represented by	circles	rectangles	diamond shaped box	ellipse	rectangles
60	In an E-R diagram relationship is represented by	circles	rectangles	diamond shaped box	ellipse	diamond shaped box

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: II  
BATCH-2017-2020

## SYLLABUS

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

**Relation Data Model**

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

**Concepts**

**Tables** – In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.

**Tuple** – A single row of a table, which contains a single record for that relation is called a tuple.

**Relation instance** – A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

**Relation schema** – A relation schema describes the relation name (table name), attributes, and their names.

**Relation key** – Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

**Attribute domain** – Every attribute has some pre-defined value scope, known as attribute domain.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

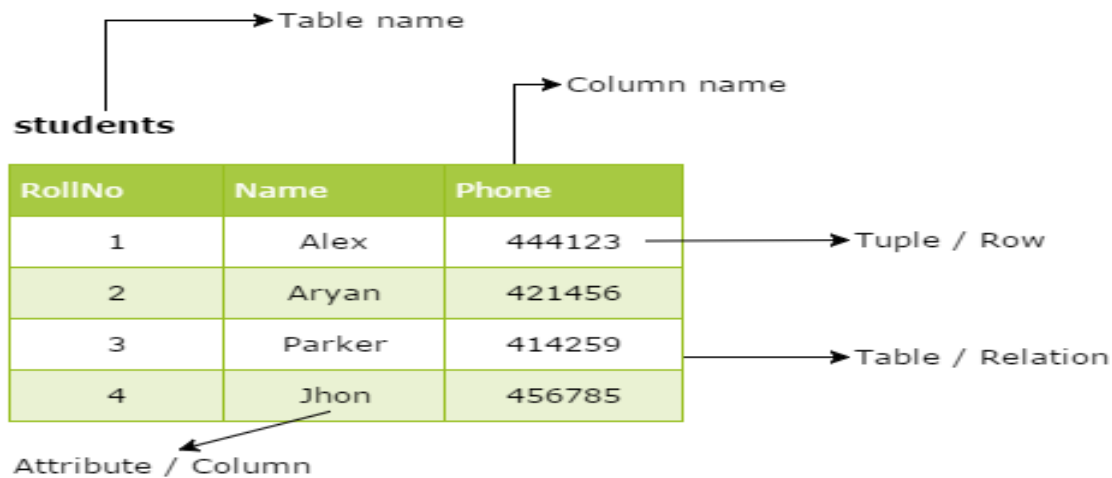
CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CSU403

UNIT: II

BATCH-2017-2020

**Some Common Relational Model Terms****Relational Model Terms**

- **Relation:** A relation is a table with columns and rows.
- **Attribute:** An attribute is a named column of a relation.
- **Domain:** A domain is the set of allowable values for one or more attributes.
- **Tuple:** A tuple is a row of a relation.

**Relational Constraints****What are Database Constraints in DBMS ??**

Database constraints are restrictions on the contents of the database or on database operations. It is a condition specified on a database schema that restricts the data to be inserted in an instance of the database.

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**.

**Need of Constraints :**

**Constraints in the database provide a way to guarantee that :**

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

- the values of individual columns are valid.
- in a table, rows have a valid primary key or unique key values.
- in a dependent table, rows have valid foreign key values that reference rows in a parent table.

**Types of constraints in DBMS:**

- **Domain Constraints**
- **Tuple Uniqueness Constraints**
- **Key Constraints**
- **Single Value Constraints**
- **Integrity Rule 1 (Entity Integrity Rule or Constraint)**
- **Integrity Rule 2 (Referential Integrity Rule or Constraint)**
- **General Constraints**

**Domain Constraints –**

Domain Constraints specifies that what set of values an attribute can take. Value of each attribute X must be an atomic value from the domain of X. The data type associated with domains include integer, character, string, date, time, currency etc. An attribute value must be available in the corresponding domain. Consider the example below –

SID	Name	Class (semester)	Age
8001	Ankit	1 <sup>st</sup>	19
8002	Srishti	1 <sup>st</sup>	18
8003	Somvir	4 <sup>th</sup>	22
8004	Sourabh	6 <sup>th</sup>	A

Not Allowed. Because Age is an Integer Attribute.

**Tuple Uniqueness Constraints –**

A relation is defined as a set of tuples. All tuples or all rows in a relation must be unique or distinct. Suppose if in a relation, tuple uniqueness constraint is applied, then all the rows of that table must be unique i.e. it does not contain the duplicate values. For example,

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

SID	Name	Class (semester)	Age
8001	Ankit	1 <sup>st</sup>	19
8002	Srishti	2 <sup>nd</sup>	18
8003	Somvir	4 <sup>th</sup>	22
8004	Sourabh	6 <sup>th</sup>	19

Not Allowed. Because all rows must be unique.

**Key Constraints –**

Keys are attributes or sets of attributes that uniquely identify an entity within its entity set. An Entity set E can have multiple keys out of which one key will be designated as the primary key. Primary Key must have unique and not null values in the relational table. In an subclass hierarchy, only the root entity set has a key or primary key and that primary key must serve as the key for all entities in the hierarchy.

**Types of keys in DBMS**

1. **Primary Key** – A primary is a column or set of columns in a table that uniquely identifies tuples (rows) in that table.

Example of Key Constraints in a simple relational table –

<u>SID</u>	Name	Class (semester)	Age
8001	Ankit	1 <sup>st</sup>	19
8002	Srishti	1 <sup>st</sup>	18
8003	Somvir	4 <sup>th</sup>	22
8004	Sourabh	6 <sup>th</sup>	45
8002	Tony	5 <sup>th</sup>	23

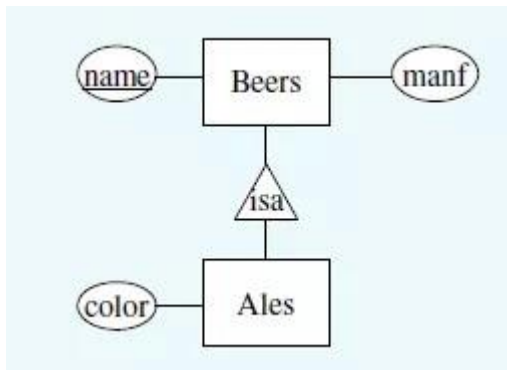
Not allowed as Primary  
Key Values must be unique

Example of Key Constraints in an subclass hierarchy –

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: II  
BATCH-2017-2020



2. **Super Key** – A super key is a set of one or more columns (attributes) to uniquely identify rows in a table. Often people get confused between super key and candidate key, so we will also discuss a little about candidate key here.

**How candidate key is different from super key?**

Answer is simple – Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is: It should not have any redundant attribute. That's the reason they are also termed as minimal super key.

Let's take an example to understand this: **Employee table**

Emp_SSN	Emp_Number	Emp_Name
123456789	226	Steve
999999321	227	Ajeet
888997212	228	Chaitanya
777778888	229	Robert

**Super keys:**

- {Emp\_SSN}



**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

- {Emp\_Number}
- {Emp\_SSN, Emp\_Number}
- {Emp\_SSN, Emp\_Name}
- {Emp\_SSN, Emp\_Number, Emp\_Name}
- {Emp\_Number, Emp\_Name}

All of the above sets are able to uniquely identify rows of the employee table.

3. **Candidate Key** – A super key with no redundant attribute is known as candidate key

A candidate key is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data. Each table may have one or more candidate keys, but one candidate key is unique, and it is called the primary key. This is usually the best among the candidate keys to use for identification.

When a key is composed of more than one column, it is known as a composite key.

A super key with no redundant attribute is known as candidate key. Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is: It should not have any redundant attributes. That's the reason they are also termed as minimal super key.

**For example:**

<u>Emp_Id</u>	Emp_Number	Emp_Name
E01	2264	Steve
E22	2278	Ajeet
E23	2288	Chaitanya
E45	2290	Robert

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

There are two candidate keys in above table:

{Emp\_Id}

{Emp\_Number}

**Note:** A primary key is being selected from the group of candidate keys. That means we can either have Emp\_Id or Emp\_Number as primary key.

4. **Alternate Key** – Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate or secondary keys.

**For example: Consider the below table**

<u>Emp_Id</u>	Emp_Number	Emp_Name
E01	2264	Steve
E22	2278	Ajeet
E23	2288	Chaitanya
E45	2290	Robert

There are two candidate keys in above table:

{Emp\_Id}

{Emp\_Number}

Since we have selected Emp\_Id as primary key, the remaining key Emp\_Number would be called alternative or secondary key.

5. **Composite Key** – A key that consists of more than one attribute to uniquely identify rows (also known as records & tuples) in a table is called composite key.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CSU403

UNIT: II

BATCH-2017-2020

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS : II.B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : DTATBASE MANAGEMENT SYSTEM

COURSE CODE : 16CSU403

UNIT II

**Example: Table – Sales**

cust_Id	order_Id	product_code	product_count
C01	O001	P007	23
C02	O123	P007	19
C02	O123	P230	82
C01	O001	P890	42

Key in above table: {cust\_id, order\_id}

This is a composite key as it consists of more than one attribute.

6. **Foreign Key** – Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

**Single Value Constraints –**

Single value constraints refers that each attribute of an entity set has a single value. If the value of an attribute is missing in a tuple, then we can fill it with a “null” value. The null value for a attribute will specify that either the value is not known or the value is not applicable. Consider the below example-

SID	Name	Class (semester)	Age	Driving License Number
8001	Ankit	1 <sup>st</sup>	19	DL-45698
8002	Srishti	2 <sup>nd</sup>	18	DL-45871, DL-89740
8003	Somvir	4 <sup>th</sup>	22	DL-95687
8004	Sourabh	6 <sup>th</sup>	19	

Not allowed as a person does not have two driving licenses.

Allowed as a person may or may not have a driving license.

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020****Integrity Rule 1 (Entity Integrity Rule or Constraint) –**

The Integrity Rule 1 is also called Entity Integrity Rule or Constraint. This rule states that no attribute of primary key will contain a null value. If a relation have a null value in the primary key attribute, then uniqueness property of the primary key cannot be maintained. Consider the example below-

<u>SID</u>	Name	Class (semester)	Age
8001	Ankit	1 <sup>st</sup>	19
8002	Srishti	2 <sup>nd</sup>	18
8003	Somvir	4 <sup>th</sup>	22
	Sourabh	6 <sup>th</sup>	19

Not allowed as primary  
key cannot contain a NULL  
value

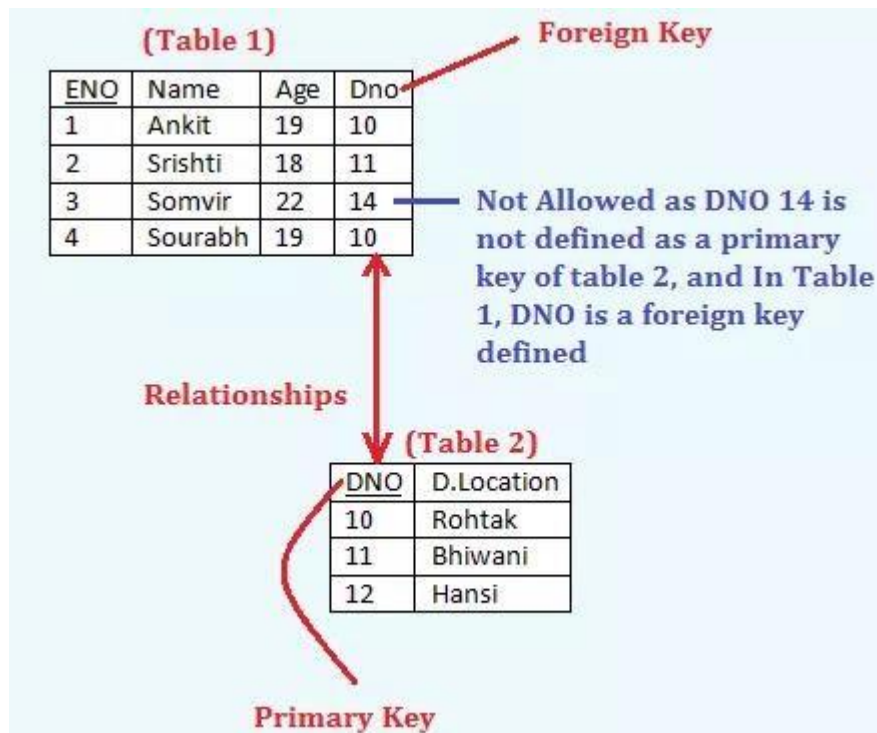
**Integrity Rule 2 (Referential Integrity Rule or Constraint) –**

The integrity Rule 2 is also called the Referential Integrity Constraints. This rule states that if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2. For example,

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: II  
BATCH-2017-2020

**Some more Features of Foreign Key –**

Let the table in which the foreign key is defined is Foreign Table or details table i.e. Table 1 in above example and the table that defines the primary key and is referenced by the foreign key is master table or primary table i.e. Table 2 in above example. Then the following properties must be hold :

- Records cannot be **inserted** into a **Foreign table** if corresponding records in the master table do not exist.
- Records of the **master table** or **Primary Table** cannot be **deleted** or **updated** if corresponding records in the detail table actually exist.

**General Constraints –**

General constraints are the arbitrary constraints that should hold in the database. Domain Constraints, Key Constraints, Tuple Uniqueness Constraints, Single Value Constraints, Integrity Rule 1 (Entity Integrity) and 2 (Referential Integrity Constraints) are considered to be a fundamental part of the relational data model.

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

However, sometimes it is necessary to specify more general constraints like the CHECK Constraints or the Range Constraints etc.

**Check constraints** can ensure that only specific values are allowed in certain column. For example, if there is a need to allow only three values for the color like 'Bakers Chocolate', 'Glistening Grey' and 'Superior White', then we can apply the check constraint. All other values like 'GREEN' etc would yield an error. Range Constraints is implemented by BETWEEN and NOT BETWEEN. For example, if it is a requirement that student ages be within 16 to 35, then we can apply the range constraints for it.

The below example will explain Check Constraint and Range Constraint –

CarID	Name	Model	Color
C-12378	Wagon-R	2008	Bakers Chocolate
C-23478	Wagon-R	2008	Glistening Grey
C-45823	Wagon-R	2004	Superior White
C-45874	Wagon-R	2009	Green

**Not Allowed, as CHECK Constraint is applied.**

SID	Name	Class (semester)	Age
8001	Ankit	1 <sup>st</sup>	19
8002	Srishti	2 <sup>nd</sup>	18
8003	Somvir	4 <sup>th</sup>	22
NULL	Sourabh	6 <sup>th</sup>	65

**Not allowed, as the range defined is in between 16 and 35**

**RELATIONAL ALGEBRA**

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

Relational Algebra is procedural query language, which takes Relation as input and generate relation as output. Relational algebra mainly provides theoretical foundation for relational databases and SQL.

- *Domain*: set of relations
- Based on set theory
- Contains extensions to manipulate tables
- Functional language
- Procedural, i.e., order to operations, algorithm implicit in the functional evaluation

**Relational Algebra Operations**

Below are fundamental operations that are "complete". That is, this set of operations alone can define any retrieval.

- Select
- Project
- Rename
- Union
- Set Difference
- Cartesian Product

Convenient, natural additions to the set of operations makes

- Set Intersection
- Natural Join
- Division
- Assignment

**Projection**

- Produce a subset of attributes from a relation
- Unselected columns are eliminated
- Duplicate rows are eliminated
- Result is a relation

**Syntax:**  $\pi_{\text{attribute-list}}(\text{relation})$

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CSU403

UNIT: II

BATCH-2017-2020

Example: The table E (for EMPLOYEE)

nr	name	salary
1	John	100
5	Sarah	300
7	Tom	100

SQL	Result	Relational algebra								
select salary from E	<table><tr><th>salary</th></tr><tr><td>100</td></tr><tr><td>300</td></tr></table>	salary	100	300	$\text{PROJECT}_{\text{salary}}(E)$					
salary										
100										
300										
select nr, salary from E	<table><tr><th>nr</th><th>salary</th></tr><tr><td>1</td><td>100</td></tr><tr><td>5</td><td>300</td></tr><tr><td>7</td><td>100</td></tr></table>	nr	salary	1	100	5	300	7	100	$\text{PROJECT}_{\text{nr, salary}}(E)$
nr	salary									
1	100									
5	300									
7	100									

Note that there are no duplicate rows in the result.

**Selection**

Choose a subset of tuples from a relation based on some criteria, results in another relation called a "**result set**"

Notation uses lower case sigma:

**Syntax:**  $\sigma_{\text{condition}}(\text{relation})$

The same table E (for EMPLOYEE) as above.

SQL	Result	Relational algebra									
select * from E where salary < 200	<table> <tr><th>nr</th><th>name</th><th>salary</th></tr> <tr><td>1</td><td>John</td><td>100</td></tr> <tr><td>7</td><td>Tom</td><td>100</td></tr> </table>	nr	name	salary	1	John	100	7	Tom	100	$\text{SELECT}_{\text{salary} < 200}(E)$
nr	name	salary									
1	John	100									
7	Tom	100									



**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

select * from E where salary < 200 and nr >= 7	<table> <tr> <th>nr</th><th>name</th><th>salary</th></tr> <tr> <td>7</td><td>Tom</td><td>100</td></tr> </table>	nr	name	salary	7	Tom	100	<b>SELECT</b> salary < 200 and nr >= 7(E)
nr	name	salary						
7	Tom	100						

**Set Operators**

One of the characteristics of RDBMS is that it should support all the transaction on the records in the table by means relational operations. That means it should have strong query language which supports relational algebra. There are three main relational algebras on sets – UNION, SET DIFFERENCE and SET INTERSECT. The same is implemented in database query language using set operators.

There are 4 main set operators used in the query language.

**1. UNION**

It combines the similar columns from two tables into one resultant table. All columns that are participating in the UNION operation should be Union Compatible. This operator combines the records from both the tables into one. If there are duplicate values as a result, then it eliminates the duplicate. The resulting records will be from both table and distinct.

---

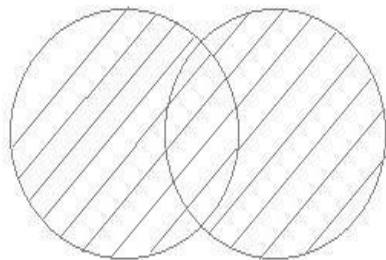
**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

EMP_TEST			
EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_SSN
100	James	Troy	232434
104	Kathy	Holland	324343

Union

EMP_DESIGN			
EMP_ID	ENAME	EMP_ADDRESS	SSN
103	Rose	Freser Town	6744545
102	Marry	Novi	343613
105	Laurry	Rochester Hills	97676
104	Kathy	Holland	324343

UNION			
EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_SSN
100	James	Troy	232434
102	Marry	Novi	343613
103	Rose	Freser Town	6744545
104	Kathy	Holland	324343
105	Laurry	Rochester Hills	97676



We can notice that Result will have same column names as first query. Duplicate record – 104 from EMP\_TEST and EMP\_DESIGN are showed only once in the result set. Records are sorted in the result.

**UNION ALL**

This operation is also similar to UNION, but it does not eliminate the duplicate records. It shows all the records from both the tables. All other features are same as UNION. We can have conditions in the SELECT query. It need not be a simple SELECT query.

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

Look at the same example below with UNION ALL operation.

EMP_TEST			
EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_SSN
100	James	Troy	232434
104	Kathy	Holland	324343

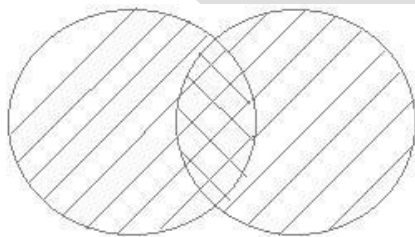
Union ALL

EMP_DESIGN			
EMP_ID	ENAME	EMP_ADDRESS	SSN
103	Rose	Freser Town	6744545
102	Marry	Novi	343613
105	Laurry	Rochester Hills	97676
104	Kathy	Holland	324343

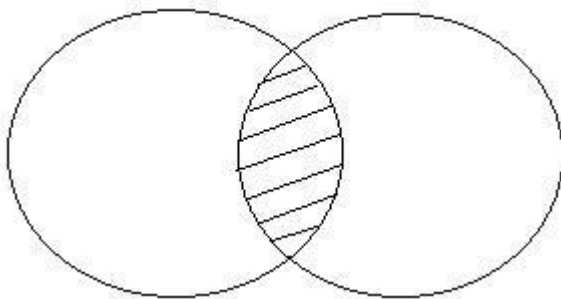
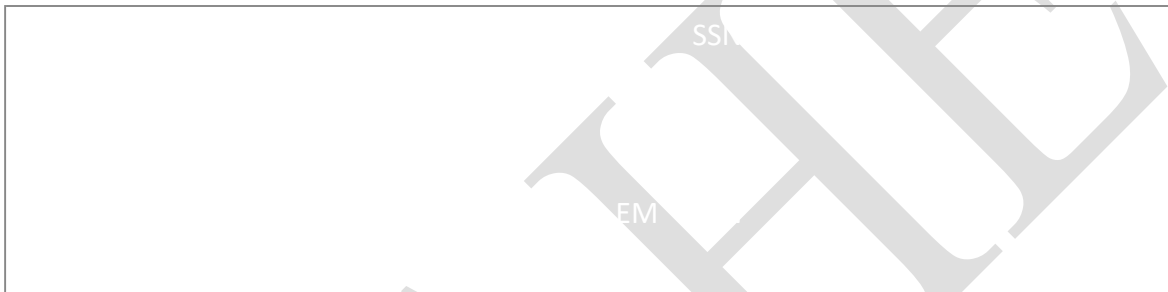
UNION			
EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_SSN
100	James	Troy	232434
102	Marry	Novi	343613
103	Rose	Freser Town	6744545
104	Kathy	Holland	324343
104	Kathy	Holland	324343
105	Laurry	Rochester Hills	97676



**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020****2. INTERSECT**

This operator is used to pick the records from both the tables which are common to them. In other words it picks only the duplicate records from the tables. Even though it selects duplicate records from the table, each duplicate record will be displayed only once in the result set. It should have UNION Compatible columns to run the query with this operator.

Same example above when used with INTERSECT operator, gives below result.



**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

EMP_TEST			
EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_SSN
100	James	Troy	232434
104	Kathy	Holland	324343

INTERSECT

EMP_DESIGN			
EMP_ID	ENAME	EMP_ADDRESS	SSN
103	Rose	Freser Town	6744545
102	Marry	Novi	343613
105	Laurry	Rochester Hills	97676
104	Kathy	Holland	324343

UNION

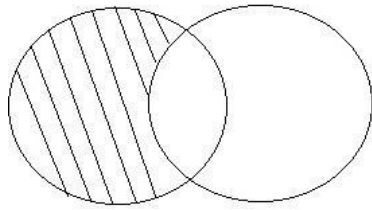
EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_SSN
104	Kathy	Holland	324343

**3. MINUS**

This operator is used to display the records that are present only in the first table or query, and doesn't present in second table / query. It basically subtracts the first query results from the second.

Let us see the same example with MINUS operator.



**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

EMP_TEST			
EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_SSN
100	James	Troy	232434
104	Kathy	Holland	324343

**MINUS**

EMP_DESIGN			
EMP_ID	ENAME	EMP_ADDRESS	SSN
103	Rose	Freser Town	6744545
102	Marry	Novi	343613
105	Laurry	Rochester Hills	97676
104	Kathy	Holland	324343



UNION			
EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_SSN
100	James	Troy	232434

We can notice in the above result that only the records that do not exist in EMP\_DESIGN are displayed in the result. The record which appears in both the tables is eliminated. Similarly, the records that appear in second query but not in the first query are also eliminated.

**4. Division Operator ( $\div$ ):****Table 1****STUDENT\_SPORTS**

ROLL_NO	SPORTS
1	Badminton
2	Cricket

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

2	Badminton
4	Badminton

**Table 2****ALL\_SPORTS**

<b>SPORTS</b>
Badminton
Cricket

**Division Operator ( $\div$ ):** Division operator  $A \div B$  can be applied if and only if:

- Attributes of B is proper subset of Attributes of A.
- The relation returned by division operator will have attributes = (All attributes of A – All Attributes of B)
- The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple.

**STUDENT\_SPORTS  $\div$  ALL\_SPORTS**

- The operation is valid as attributes in ALL\_SPORTS is a proper subset of attributes in STUDENT\_SPORTS.
- The attributes in resulting relation will have attributes {ROLL\_NO,SPORTS}- {SPORTS}=ROLL\_NO
- The tuples in resulting relation will have those ROLL\_NO which are associated with all B's tuple {Badminton, Cricket}. ROLL\_NO 1 and 4 are associated to Badminton only. ROLL\_NO 2 is associated to all tuples of B. So the resulting relation will be:

<b>ROLL_NO</b>
2

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: II  
BATCH-2017-2020

**Join in SQL**

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. **Join** Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is **(n-1)** where **n**, is number of tables. A table can also join to itself known as, **Self Join**.

**Types of Join**

The following are the types of JOIN that we can use in SQL.

- Inner
- Outer
- Left
- Right

**Cross JOIN or Cartesian Product**

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

```
SELECT column-name-list
```

```
from table-name1
```

**CROSS JOIN**

```
table-name2;
```

---

**Example of Cross JOIN**

The **class** table,



**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

ID	NAME
1	abhi
2	adam
4	alex

The **class\_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

**Cross JOIN** query will be,

```
SELECT *  
from class,  
cross JOIN class_info;
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	1	DELHI
4	alex	1	DELHI
1	abhi	2	MUMBAI
2	adam	2	MUMBAI

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

4	alex	2	MUMBAI
1	abhi	3	CHENNAI
2	adam	3	CHENNAI
4	alex	3	CHENNAI

**INNER Join or EQUI Join**

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the query.

Inner Join Syntax is,

SELECT column-name-list

from *table-name1*

**INNER JOIN**

*table-name2*

WHERE table-name1.column-name = table-name2.column-name;

**Example of Inner JOIN**

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

The **class\_info** table,

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

**Inner JOIN** query will be,

```
SELECT * from class, class_info where class.id = class_info.id;
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI

**Natural JOIN**

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

Natural Join Syntax is,

```
SELECT *  
from table-name1
```

**NATURAL JOIN**

```
table-name2;
```

**Example of Natural JOIN**

The **class** table,

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

ID	NAME
1	abhi
2	adam
3	alex
4	anu

The **class\_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

**Natural join query will be,**

```
SELECT * from class NATURAL JOIN class_info;
```

The result table will look like,

ID	NAME	Address
1	abhi	DELHI
2	adam	MUMBAI
3	alex	CHENNAI

In the above example, both the tables being joined have ID column(same name and same datatype), hence the records for which value of ID matches in both the tables will be the result of Natural Join of these two tables.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: II  
BATCH-2017-2020

**Outer JOIN**

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

- Left Outer Join
- Right Outer Join
- Full Outer Join

**Left Outer Join**

The left outer join returns a result table with the **matched data** of two tables then remaining rows of the **left** table and null for the **right** table's column.

Left Outer Join syntax is,

```
SELECT column-name-list
```

```
from table-name1
```

```
LEFT OUTER JOIN
```

```
table-name2
```

```
on table-name1.column-name = table-name2.column-name;
```

Left outer Join Syntax for **Oracle** is,

```
select column-name-list
```

```
from table-name1,
```

```
table-name2
```

```
on table-name1.column-name = table-name2.column-name(+);
```

**Example of Left Outer Join**

The **class** table,

ID	NAME
----	------

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020**

1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class\_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

**Left Outer Join** query will be,

```
SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id=class_info.id);
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020****II**

4	anu	null	null
5	ashish	null	null

**Right Outer Join**

The right outer join returns a result table with the **matched data** of two tables then remaining rows of the **right table** and null for the **left** table's columns.

Right Outer Join Syntax is,

select column-name-list

from *table-name1*

**RIGHT OUTER JOIN**

*table-name2*

on table-name1.column-name = table-name2.column-name;

Right outer Join Syntax for **Oracle** is,

select column-name-list

from *table-name1*,

*table-name2*

on table-name1.column-name(+) = table-name2.column-name;

**Example of Right Outer Join**

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: II**  
**BATCH-2017-2020**

4	anu
5	ashish

The **class\_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

**Right Outer Join** query will be,

```
SELECT * FROM class RIGHT OUTER JOIN class_info on (class_id=class_info.id);
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: II  
BATCH-2017-2020

**Full Outer Join**

The full outer join returns a result table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

Full Outer Join Syntax is,

select column-name-list

from *table-name1*

**FULL OUTER JOIN**

*table-name2*

on table-name1.column-name = table-name2.column-name;

**Example of Full outer join is,**

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class\_info** table,

ID	Address
1	DELHI
2	MUMBAI

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: II****BATCH-2017-2020****KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS : II.B.Sc CS****BATCH : 2016 - 2019****COURSE NAME : DTATBASE MANAGEMENT SYSTEM****COURSE CODE : 16CSU403****UNIT II**

3	CHENNAI
7	NOIDA
8	PANIPAT

**Full Outer Join** query will be like,

```
SELECT * FROM class FULL OUTER JOIN class_info on (class.id=class_info.id);
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

**Aggregate Functions in SQL**

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

**Various Aggregate Functions**

1) Count()

Prepared By : K.Kathirvel &amp; Mrs K.Banuroopa Dept of CS,CA &amp; IT,KAHE

31/35

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS**  
**COURSE CODE: 17CSU403****COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: II**  
**BATCH-2017-2020**

3) Avg()

4) Min()

5) Max()

Now let us understand each Aggregate function with a example: Table : E

Id	Name	Salary
1	A	80
2	B	40
3	C	60
4	D	70
5	E	60
6	F	Null

**1. COUNT()****Syntax:**

```
SELECT COUNT(column_name)
FROM table_name
```

```
WHERE condition;
```

**Example**

```
SELECT COUNT(ID) FROM E;
```

**Count(\*):** Returns total number of records .i.e 6.

**Count(salary):** Return number of Non Null values over the column salary. i.e 5.

**Count(Distinct Salary):** Return number of distinct Non Null values over the column salary .i.e 4

**2. SUM()**

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: II  
BATCH-2017-2020

**Syntax**

```
SELECT SUM(column_name)  
FROM table_name
```

```
WHERE condition;
```

**Example**

```
SELECT sum(ID) FROM E
```

**sum(salary):** Sum all Non Null values of Column salary i.e., 310

**sum(Distinct salary):** Sum of all distinct Non-Null values i.e., 250.

**3. AVG()****Syntax**

```
SELECT AVG(column_name)  
FROM table_name
```

```
WHERE condition;
```

**Example**

```
SELECT AVG(salary) FROM E;
```

**Avg(salary)** = Sum(salary) / count(salary) = 310/5

**Avg(Distinct salary)** = sum(Distinct salary) / Count(Distinct Salary) = 250/4

**4. MIN()****Syntax**

```
SELECT MIN(column_name)  
FROM table_name
```

```
WHERE condition;
```

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CSU403

UNIT: II

BATCH-2017-2020

**Example**

```
SELECT MIN(salary) FROM E;
```

**Min(salary):** Minimum value in the salary column except NULL i.e., 40.

**5. MAX()****Syntax**

```
SELECT MAX(column_name)
```

```
FROM table_name
```

```
WHERE condition;
```

**Example**

```
SELECT MAX(salary) FROM E;
```

**Max(salary):** Maximum value in the salary i.e., 80.

**POSSIBLE QUESTIONS****2 MARK QUESTIONS**

1. Draw an ER diagram for 1:M relationship.
2. Draw an ER diagram for M:M relationship
3. Write the syntax for DELETE FROM WHERE command in SQL.
4. What is an Entity type?
5. Define cardinalities.
6. Draw an ER model for M:M relationship.



**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
**DEPARTMENT OF COMPUTER SCIENCE, CA & IT**  
**II BSc CS- Batch 2017-2020**  
**PART-A OBJECTIVE TYPE/ One mark -Online Examination**  
**DATABASE MANAGEMENT SYSTEM(17CSU403)**  
**UNIT-II**

SNO	Questions	opt1	opt2	opt3	opt4	Answer
1	The rows of a relation	must be in specified order	may be in any order	in ascending order of key	in descending order of key	may be in any order
2	The columns of a relation	must be in specified order	may be in any order	with key field in first column	with largest width column last	may be in any order
3	A _____ is a dependency of one non primary key attribute on another non primary key attribute.	Functional dependency	Transitive dependency	Partial dependency	Non primary key dependency	Partial dependency
4	functional depency of the form $X \rightarrow Y$ where Y is a subset of X are called _____	Functional dependency	Transitive dependency	Partial dependency	Trivial dependency	Trivial dependency
5	Group function is otherwise known as _____	Collection	Aggregate	function	Count	Aggregate
6	Pattern matching has done through _____ operator.	Comparison	arithmetic	Logical	Aggregate	Aggregate
7	Which keyword is used to check if an element is in a given set?	not	in	not exist	except	in
8	Any two tables that are Union-Compatible that is, have the same number of _____	Columns	rows	Columns and rows	null values	Columns
9	_____ keyword is used to eliminates the duplicates	Union	Union all	Intersect all	Except all	Union
10	_____ keyword is used to retain the duplicates	Union	Union all	Intersect	Except	Union all
11	_____ technique is used to reduce the redundancy	Closure set	Decomposition	Normalization	Null Values	Normalization
12	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
13	The Closure of F denoted as _____	Fc	F—	FXX	F+	F+
14	All decomposition are used to eliminate _____	duplicates	null values	empty values	not null values	duplicates
15	_____ is a kind of IC, that generalizes the concept of a key.	Decomposition	Functional dependency	cursors	Triggers	Functional dependency
16	_____ Consists of replacing the relation schema by two relation schemas that each contain a subsets of the attributes	Decomposition	Functional dependency	cursors	Triggers	Decomposition
17	X is a proper subset of some key K. Such a dependency is sometimes called _____	dependency	Partial dependency	transitive dependency	Decompostion	Partial dependency
18	X is not a proper subset of some key K. Such a dependency is sometimes called _____	dependency	Partial dependency	transitive dependency	Decompostion	transitive dependency
19	Indicated by using arrow from entities to relationships in the ER diagram.	Arrow	Thick line	Dotted line	Shaded line	Arrow
20	Aggragation is indicated by _____ in ER diagram	Solid line	Thick line	Thin line	Dotted line	Dotted line
21	ISA is indicated by _____ symbol	Rectangle	Ellipse	Triangle	Diamond	Triangle
22	_____ is a set of associated values	Entity	Attribute	Relationships	Domain	Domain
23	_____ consists of a relation schema and a relation instance.	relation	table	domain	entity	relation
24	An instance of a relation is a set of _____	tuple	domain	attribute	relationships	tuple
25	Each tuple is a _____	Column	row	table	instance	row
26	_____ is an object in the real world	Entity	Attribute	Relationship	Property	Entity
27	Collection Of Similar entities are _____	Attributes	Entity	Entity Set	Relationship	Entity Set
28	An Entity is described using a set Of _____	Entity	Entity Set	Attributes	Relationship	Attributes
29	_____ is used to uniquely identify an entity in the set.	Key	Lock	Attributes	Entity	Key
30	DBMS is a collection of ..... that enables user to create and maintain a database.	Keys	Translators	Program	Language Activity	Program

31	In a relational schema, each tuple is divided into fields called	Relations	Domains	Queries	All of the above	Domains
32	In an ER model, ..... is described in the database by storing its data.	Entity	Attribute	Relationship	Notation	Entity
33	DFD stands for	Data Flow Document	Data File Diagram	Data Flow Diagram	Non of the above	Data Flow Diagram
34	A top-to-bottom relationship among the items in a database is established by a	Hierarchical schema	Network schema	Relational Schema	All of the above	Hierarchical schema
35	..... table store information about database or about the system.	SQL	Nested	System	None of these	System
36	.....defines the structure of a relation which consists of a fixed set of attribute-domain pairs.	Instance	Schema	Program	Super Key	Schema
37	..... clause is an additional filter that is applied to the result.	Select	Group-by	Having	Order by	Having
38	A logical schema	is the entire database	way of organizing	data is actually stored on disk.	All of the above	way of organizing
39	..... is a full form of SQL.	Standard query language	Sequential query language	Structured query language	Server side query language	Structured query language
40	A relational database developer refers to a record as	a criteria	a relation	a tuple	an attribute	a tuple
41	..... keyword is used to find the number of values in a column.	TOTAL	COUNT	ADD	SUM	COUNT
42	An advantage of the database management approach is	data is dependent on programs	redundancy increases	and can be accessed by	none of the above	integrated and can be accessed
43	The collection of information stored in a database at a particular moment is called as .....	schema	instance of the database	data domain	independence	instance of the database
44	A ..... is used to define overall design of the database	schema	application program	data definition language	code	schema
45	Key to represent relationship between tables is called	primary key	secondary key	foreign key	none of the above	foreign key
46	Grant and revoke are ..... statements	DDL	TCL	DCL	DML	DCL
47	DBMS helps achieve	Data independence	Centralized control of data	Neither A nor B	Both A and B	Both A and B
48	..... command can be used to modify a column in a table	alter	update	set	create	alter
49						
50	The candidate key is that you choose to identify each row uniquely is called .....	Alternate Key	Primary Key	Foreign Key	None of the above	Primary Key
51	..... is used to determine whether of a table contains duplicate rows.	Unique predicate	Like Predicate	Null predicate	In predicate	Unique predicate
52	To eliminate duplicate rows ..... is used	NODUPPLICATE	ELIMINATE	DISTINCT	None of these	DISTINCT
53	DCL stands for	Data Control Language	Data Console Language	Data Console Level	Data Control Level	Data Control Language
54	..... is the process of organizing data into related tables.	Normalization	Generalization	Specialization	None of the above	Normalization
55	if its own and mostly are dependent entities, which are part of some another entity.	Weak entity	Strong entity	Non attributes entity	Dependent entity	Weak entity
56	..... is the complex search criteria in the where clause.	Sub string	Drop Table	Predict	Predicate	Predicate
57	..... is preferred method for enforcing data integrity	Constraints	Stored Procedure	Triggers	Cursors	Constraints
58	While the number of attributes in a relation is called it's .....	Degree, Cardinality	Cardinality, Degree	Rows, Columns	Columns, Rows	Cardinality, Degree
59	The language that requires a user to specify the data to be retrieved without specifying exactly how to get it is	Procedural DML	Procedural DML	Procedural DDL	Non-Procedural DDL	Procedural DML
60	Which two files are used during operation of the DBMS?	Query languages and utilities	DML and query language	and transaction log	and query language	and transaction log



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

### What is SQL

- SQL stands for **Structured Query Language**.
- It is designed for managing data in a relational database management system (RDBMS).
- It is pronounced as S-Q-L or sometime **See-Qwell**.
- SQL is a database language, it is used for database creation, deletion, fetching rows and modifying rows etc.
- SQL is based on relational algebra and tuple relational calculus.

All DBMS like MySQL, Oracle, MS Access, Sybase, Informix, Postgres and SQL Server use SQL as standard database language.

### **Why SQL is required**

SQL is required:

- To create new databases, tables and views
- To insert records in a database
- To update records in a database
- To delete records from a database
- To retrieve data from a database

### **What SQL does**

- With SQL, we can query our database in a numbers of ways, using English-like statements.
- With SQL, user can access data from relational database management system.
- It allows user to describe the data.
- It allows user to define the data in database and manipulate it when needed.
- It allows user to create and drop database and table.

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: III****BATCH-2017-2020**

- It allows user to create view, stored procedure, function in a database.
- It allows user to set permission on tables, procedure and view.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

SQL Table

Table is a collection of data, organized in terms of rows and columns. In DBMS term, table is known as relation and row as tuple.

**Note: A table has a specified number of columns, but can have any number of rows.**

Table is the simple form of data storage. A table is also considered as a convenient representation of relations.

Let's see an example of an employee table:

Employee		
EMP_NAME	ADDRESS	SALARY
Ankit	Lucknow	15000
Raman	Allahabad	18000
Mike	New York	20000

In the above table, "Employee" is the table name, "EMP\_NAME", "ADDRESS" and "SALARY" are the column names. The combination of data of multiple columns forms a row e.g. "Ankit", "Lucknow" and 15000 are the data of one row.

SQL TABLE Variable

The **SQL Table variable** is used to create, modify, rename, copy and delete tables. Table variable was introduced by Microsoft.

It was introduced with SQL server 2000 to be an alternative of temporary tables.

It is a variable where we temporary store records and results. This is same like temp table but in the case of temp table we need to explicitly drop it.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

Table variables are used to store a set of records. So declaration syntax generally looks like CREATE TABLE syntax.

```
create table "tablename"  
("column1" "data type",  
"column2" "data type",  
...  
"columnN" "data type");
```

When a transaction rolled back the data associated with table variable is not rolled back.

A table variable generally uses lesser resources than a temporary variable.

Table variable cannot be used as an input or an output parameter.

### SQL CREATE TABLE

SQL CREATE TABLE statement is used to create table in a database.

If you want to create a table, you should name the table and define its column and each column's data type.

Let's see the simple syntax to create the table.

```
create table "tablename"  
("column1" "data type",  
"column2" "data type",  
"column3" "data type",  
...  
"columnN" "data type");
```

The data type of the columns may vary from one database to another. For example, NUMBER is supported in Oracle database for integer value whereas INT is supported in MySQL.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

Let us take an example to create a STUDENTS table with ID as primary key and NOT NULL are the constraint showing that these fields cannot be NULL while creating records in the table.

```
SQL> CREATE TABLE STUDENTS (
ID INT NOT NULL,
NAME VARCHAR (20) NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR (25),
PRIMARY KEY (ID)
);
```

You can verify it, if you have created the table successfully by looking at the message displayed by the SQL Server, else you can use DESC command as follows:

```
SQL> DESC STUDENTS;
```

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
ID	Int(11)	NO	PRI		
NAME	Varchar(20)	NO			
AGE	Int(11)	NO			
ADDRESS	Varchar(25)	YES		NULL	

4 rows in set (0.00 sec)

Now you have the STUDENTS table available in your database and you can use to store required information related to students.

### SQL INSERT STATEMENT

SQL INSERT statement is a SQL query. It is used to insert a single or a multiple records in a table.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

There are two ways to insert data in a table:

1. By SQL insert into statement
  1. By specifying column names
  2. Without specifying column names
2. By SQL insert into select statement

### 1) Inserting data directly into a table

You can insert a row in the table by using SQL INSERT INTO command. But there are 2 ways to do this.

You can specify or ignore the column names while using INSERT INTO statement.

To insert partial column values, you must have to specify the column names. But if you want to insert all the column values, you can specify or ignore the column names.

If you specify the column names, syntax of the insert into statement will be as follows:

**INSERT INTO** TABLE\_NAME

[(col1, col2, col3, ....col N)]

**VALUES** (value1, value2, value 3, .... Value N);

Here col1, col2, col3, .... colN are the columns of the table in which you want to insert data.

But, If you ignore the column names, syntax of the insert into statement will be as follows:

**INSERT INTO** TABLE\_NAME

**VALUES** (value1, value2, value 3, .... Value N);

### 2) Inserting data through SELECT Statement

**SQL INSERT INTO SELECT Syntax**

**INSERT INTO** table\_name

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

[(column1, column2,.... column)]

**SELECT** column1, column2,..... Column N

**FROM** table\_name [**WHERE** condition];

**Note:** when you add a new row, you should make sure that data type of the value and the column should be matched.

If any integrity constraints are defined for the table, you must follow them.

### SQL INSERT INTO VALUE

There are two ways to insert values in a table.

In the first method there is no need to specify the column name where the data will be inserted, you need only their values.

**INSERT INTO** table\_name

**VALUES** (value1, value2, value3... );

The second method specifies both the column name and values which you want to insert.

**INSERT INTO** table\_name (column1, column2, column3....)

**VALUES** (value1, value2, value3... );

Let's take an example of table which has five records within it.

**INSERT INTO** STUDENTS (ROLL\_NO, NAME, AGE, CITY)

**VALUES** (1, ABHIRAM, 22, ALLAHABAD);

**INSERT INTO** STUDENTS (ROLL\_NO, NAME, AGE, CITY)

**VALUES** (2, ALKA, 20, GHAZIABAD);

**INSERT INTO** STUDENTS (ROLL\_NO, NAME, AGE, CITY)

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**VALUES** (3, DISHA, 21, VARANASI);

**INSERT INTO** STUDENTS (ROLL\_NO, **NAME**, AGE, CITY)

**VALUES** (4, ESHA, 21, DELHI);

**INSERT INTO** STUDENTS (ROLL\_NO, **NAME**, AGE, CITY)

**VALUES** (5, MANMEET, 23, JALANDHAR);



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

It will show the following table as the final result.

ROLL_NO	NAME	AGE	CITY
1	ABHIRAM	22	ALLAHABAD
2	ALKA	20	GHAZIABAD
3	DISHA	21	VARANASI
4	ESHA	21	DELHI
5	MANMEET	23	JALANDHAR

You can create a record in CUSTOMERS table by using this syntax also.

**INSERT INTO** CUSTOMERS

**VALUES** (6, PRATIK, 24, KANPUR);

The following table will be as follow:

ROLL_NO	NAME	AGE	CITY
1	ABHIRAM	22	ALLAHABAD
2	ALKA	20	GHAZIABAD
3	DISHA	21	VARANASI
4	ESHA	21	DELHI
5	MANMEET	23	JALANDHAR
6	PRATIK	24	KANPUR

### SQL SELECT

The most commonly used SQL command is **SELECT statement**. It is used to query the database and retrieve selected data that follow the conditions we want.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

In simple words, we can say that the select statement used to query or retrieve data from a table in the database.

Let's see the syntax of select statement.

**SELECT** expressions

**FROM** tables

**WHERE** conditions;

Here expression is the column that we want to retrieve.

Tables indicate the tables, we want to retrieve records from.

### Optional clauses in SELECT statement

There are some optional clauses in SELECT statement:

**[WHERE Clause]** : It specifies which rows to retrieve.

**[GROUP BY Clause]** : Groups rows that share a property so that the aggregate function can be applied to each group.

**[HAVING Clause]** : It selects among the groups defined by the GROUP BY clause.

**[ORDER BY Clause]** : It specifies an order in which to return the rows.

For example, let a database table: student\_details;

ID	First_name	Last_name	Age	Subject	Hobby
1	Amar	Sharma	20	Maths	Cricket
2	Akbar	Khan	22	Biology	Football
3	Anthony	Milton	25	Commerce	Gambling

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

From the above example, select the first name of all the students. To do so, query should be like this:

```
SELECT first_name FROM student_details;
```

Note: the SQL commands are not case sensitive. We can also write the above SELECT statement as:

```
select first_name from student_details;
```

Now, you will get following data:

Amar
Akbar
Anthony

We can also retrieve data from more than one column. For example, to select first name and last name of all the students, you need to write

```
SELECT first_name, last_name FROM student_details;
```

Now, you will get following data:

Amar	Sharma
Anthony	Milton

We can also use clauses like WHERE, GROUP BY, HAVING, ORDER BY with SELECT statement.

Here a point is notable that only SELECT and FROM statements are necessary in SQL SELECT statements. Other clauses like WHERE, GROUP BY, ORDER BY, HAVING may be optional.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

SQL SELECT DISTINCT

The **SQL DISTINCT command** is used with SELECT key word to retrieve only distinct or unique data.

In a table, there may be a chance to exist a duplicate value and sometimes we want to retrieve only unique values. In such scenarios, SQL SELECT DISTINCT statement is used.

**Note: SQL SELECT UNIQUE and SQL SELECT DISTINCT statements are same.**

Let's see the syntax of select distinct statement.

**SELECT DISTINCT** column\_name ,column\_name

**FROM** table\_name;

Let's try to understand it by the table given below:

Student_Name	Gender	Mobile_Number	HOME_TOWN
Rahul Ojha	Male	7503896532	Lucknow
Disha Rai	Female	9270568893	Varanasi
Sonoo Jaiswal	Male	9990449935	Lucknow

Here is a table of students from where we want to retrieve distinct information For example: distinct home-town.

**SELECT DISTINCT** home\_town

**FROM** students

Now, it will return two rows.

**HOME\_TOWN**

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

Lucknow
Varanasi

**SQL SELECT COUNT**

The **SQL COUNT()** function is used to return the number of rows in a query.

The COUNT() function is used with SQL SELECT statement and it is very useful to count the number of rows in a table having enormous data.

**For example:** If you have a record of the voters in selected area and want to count the number of voters then it is very difficult to do it manually but you can do it easily by using the SQL SELECT COUNT query.

Let's see the syntax of SQL COUNT statement.

**SELECT COUNT** (expression)

**FROM** tables

**WHERE** conditions;

Let's see the examples of sql select count function.

**SQL SELECT COUNT(column\_name)**

**SELECT COUNT(name) FROM** employee\_table;

It will return the total number of names of employee\_table. But null fields will not be counted.

**SQL SELECT COUNT(\*)**

**SELECT COUNT(\*) FROM** employee\_table;

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: III****BATCH-2017-2020**

The "select count(\*) from table" is used to return the number of records in table.

**SQL SELECT COUNT(DISTINCT column\_name)**

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

```
SELECT COUNT(DISTINCT name) FROM employee_table;
```

It will return the total distinct names of employee\_table.

### SQL SELECT AS

**SQL AS** is used to assign temporarily a new name to a table column.

It makes easy presentation of query results and allows the developer to label results more accurately without permanently renaming table columns.

Let's see the example of select as:

```
SELECT day_of_order AS "Date"  
Customer As "Client",
```

```
Product,  
Quantity,  
FROM orders;
```

Let us take a table named orders, it contains:

Day_of_order	Customer	Product	Quantity
11-09-2001	Ajeet	Mobile	2
13-12-2001	Mayank	Laptop	20
26-12-2004	Balaswamy	Water cannon	35

After applying this SQL AS example syntax

```
SELECT day_of_order AS "Date"  
Customer As "Client",
```

```
Product,  
Quantity,  
FROM orders;
```

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

Result will be shown as this table:

Date	Client	Product	Quantity
11-09-2001	Ajeet	Mobile	2
13-12-2001	Mayank	Laptop	20
26-12-2004	Balaswamy	Water cannon	35

Note: SQL AS is same as SQL ALIAS.

**SQL SELECT IN**

SQL IN is an operator used in a SQL query to help reduce the need to use multiple SQL "OR" conditions.

It is used in SELECT, INSERT, UPDATE or DELETE statement.

**Advantage of SQL SELECT IN**

It minimizes the use of SQL OR operator.

Let's see the syntax for SQL IN:

Expression IN (value 1, value 2 ... value n);

Take an example with character values.

**SELECT \***

**FROM** students

**WHERE** students\_name IN ( Amit , Raghav, Rajeev)

Let's take another example with numeric values.

**SELECT \***

**FROM** marks



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**WHERE** roll\_no IN (001, 023, 024);

**SQL UPDATE**

The SQL commands (*UPDATE* and *DELETE*) are used to modify the data that is already in the database. The SQL *DELETE* command uses a *WHERE* clause.

**SQL UPDATE** statement is used to change the data of the records held by tables. Which rows is to be update, it is decided by a condition. To specify condition, we use *WHERE* clause.

The *UPDATE* statement can be written in following form:

**UPDATE** table\_name **SET** [column\_name1= value1,... column\_nameN = valueN] [**WHERE** condition]

Let's see the Syntax:

**UPDATE** table\_name

**SET** column\_name = expression

**WHERE** conditions

Let's take an example: here we are going to update an entry in the source table.

SQL statement:

**UPDATE** students

**SET** User\_Name = 'beinghuman'

**WHERE** Student\_Id = '3'

Source Table:

Student_Id	FirstName	LastName	User_Name
------------	-----------	----------	-----------

**KARPAGAM ACADEMY OF HIGHER EDUCATION****CLASS: II BSC CS****COURSE NAME: DATABASE MANAGEMENT SYSTEM****COURSE CODE: 17CSU403****UNIT: III****BATCH-2017-2020**

1	Ada	Sharma	sharmili
---	-----	--------	----------

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

2	Rahul	Maurya	sofamous
3	James	Walker	jonny

See the result after updating value:

Student_Id	FirstName	LastName	User_Name
1	Ada	Sharma	sharmili
2	Rahul	Maurya	sofamous
3	James	Walker	beinghuman

### Updating Multiple Fields:

If you are going to update multiple fields, you should separate each field assignment with a comma.

SQL UPDATE statement for multiple fields:

**UPDATE** students

**SET** User\_Name = 'beserious', First\_Name = 'Johnny'

**WHERE** Student\_Id = '3'

Result of the table is given below:

Student_Id	FirstName	LastName	User_Name
1	Ada	Sharma	sharmili
2	Rahul	Maurya	sofamous
3	Johnny	Walker	beserious

### SQL DROP TABLE

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

A SQL DROP TABLE statement is used to delete a table definition and all data from a table.

This is very important to know that once a table is deleted all the information available in the table is lost forever, so we have to be very careful when using this command.

Let's see the syntax to drop the table from the database.

**DROP TABLE** "table\_name";

Let us take an example:

First we verify STUDENTS table and then we would delete it from the database.

SQL> **DESC** STUDENTS;

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
ID	Int(11)	NO	PRI		
NAME	Varchar(20)	NO			
AGE	Int(11)	NO			
ADDRESS	Varchar(25)	YES		NULL	

4 rows in set (0.00 sec)

This shows that STUDENTS table is available in the database, so we can drop it as follows:

SQL> **DROP TABLE** STUDENTS;

Now, use the following command to check whether table exists or not.

SQL> **DESC** STUDENTS;

Query OK, 0 rows affected (0.01 sec)

As you can see, table is dropped so it doesn't display it.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**SQL DELETE**

The **SQL DELETE statement** is used to delete rows from a table. Generally DELETE statement removes one or more records from a table.

**SQL DELETE Syntax**

Let's see the Syntax for the SQL DELETE statement:

**DELETE FROM** table\_name [**WHERE** condition];

Here table\_name is the table which has to be deleted. The *WHERE clause* in SQL DELETE statement is optional here.

**SQL DELETE Example**

Let us take a table, named 'EMPLOYEE' table.

ID	EMP_NAME	CITY	SALARY
101	Adarsh Singh	Obra	20000
102	Sanjay Singh	Meerut	21000
103	Priyanka Sharma	Raipur	25000
104	Esha Singhal	Delhi	26000

Example of delete with WHERE clause is given below:

**DELETE FROM** EMPLOYEE **WHERE** ID=101;

Resulting table after the query:

ID	EMP_NAME	CITY	SALARY
----	----------	------	--------

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CSU403

UNIT: III

BATCH-2017-2020

102	Sanjay Singh	Meerut	21000
103	Priyanka Sharma	Raipur	25000
104	Esha Singhal	Delhi	26000

Another example of delete statement is given below

**DELETE FROM** EMPLOYEE;

Resulting table after the query:

ID	EMP_NAME	CITY	SALARY
----	----------	------	--------

It will delete all the records of EMPLOYEE table.

It will delete the all the records of EMPLOYEE table where ID is 101.

The WHERE clause in the SQL DELETE statement is optional and it identifies the rows in the column that gets deleted.

WHERE clause is used to prevent the deletion of all the rows in the table, If you don't use the WHERE clause you might loss all the rows.

### SQL DELETE TABLE

The DELETE statement is used to delete rows from a table. If you want to remove a specific row from a table you should use WHERE condition.

**DELETE FROM** table\_name [**WHERE** condition];

But if you do not specify the WHERE condition it will remove all the rows from the table.

**DELETE FROM** table\_name;

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

There are some more terms similar to DELETE statement like as DROP statement and TRUNCATE statement but they are not exactly same there are some differences between them.

### Difference between DELETE and TRUNCATE statements

There is a slight difference b/w delete and truncate statement. The **DELETE statement** only deletes the rows from the table based on the condition defined by WHERE clause or delete all the rows from the table when condition is not specified.

But it does not free the space containing by the table.

The **TRUNCATE statement**: it is used to delete all the rows from the table **and free the containing space**.

Let's see an "employee" table.

Emp_id	Name	Address	Salary
1	Aryan	Allahabad	22000
2	Shurabhi	Varanasi	13000
3	Pappu	Delhi	24000

Execute the following query to truncate the table:

**TRUNCATE TABLE** employee;

### Difference b/w DROP and TRUNCATE statements

When you use the drop statement it deletes the table's row together with the table's definition so all the relationships of that table with other tables will no longer be valid.

**When you drop a table:**

- Table structure will be dropped

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

- Relationship will be dropped
- Integrity constraints will be dropped
- Access privileges will also be dropped

On the other hand when we **TRUNCATE** a table, the table structure remains the same, so you will not face any of the above problems.

**SQL DELETE ROW**

Let us take an example of student.

**Original table:**

ID	STUDENT_NAME	ADDRESS
001	AJEET MAURYA	GHAZIABAD
002	RAJA KHAN	LUCKNOW
003	RAVI MALIK	DELHI

If you want to delete a student with id 003 from the student\_name table, then the SQL DELETE query should be like this:

**DELETE FROM** student\_name

**WHERE** id = 003;

Resulting table after SQL DELETE query:

ID	STUDENT_NAME	ADDRESS
001	AJEET MAURYA	GHAZIABAD
002	RAJA KHAN	LUCKNOW

**SQL DELETE ALL ROWS**



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

The statement SQL DELETE ALL ROWS is used to delete all rows from the table. If you want to delete all the rows from student table the query would be like,

**DELETE FROM** STUDENT\_NAME;

Resulting table after using this query:

ID	STUDENT_NAME	ADDRE
----	--------------	-------

### SQL ALTER TABLE

The ALTER TABLE statement is used to add, modify or delete columns in an existing table. It is also used to rename a table.

You can also use SQL ALTER TABLE command to add and drop various constraints on an existing table.

#### SQL ALTER TABLE Add Column

If you want to add columns in SQL table, the SQL alter table syntax is given below:

**ALTER TABLE** table\_name **ADD** column\_name column-definition;

If you want to add multiple columns in table, the SQL table will be

```
ALTER TABLE table_name
ADD (column_1 column-definition,
      column_2 column-definition,
      .....
      column_n column-definition);
```

#### SQL ALTER TABLE Modify Column

If you want to modify an existing column in SQL table, syntax is given below:

**ALTER TABLE** table\_name **MODIFY** column\_name column\_type;

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

If you want to modify multiple columns in table, the SQL table will be

**ALTER TABLE** table\_name

**MODIFY** (column\_1 column\_type,  
column\_2 column\_type,  
.....  
column\_n column\_type);

### More SQL

### Complex Queries

The SQL AND & OR operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called as the conjunctive operators.

These operators provide a means to make multiple comparisons with different operators in the same SQL statement.

### The AND Operator

The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

### **Syntax**

The basic syntax of the AND operator with a WHERE clause is as follows –

SELECT column1, column2, columnN

FROM table\_name

WHERE [condition1] AND [condition2]...AND [conditionN];

You can combine N number of conditions using the AND operator. For an action to be taken by the SQL statement, whether it be a transaction or a query, all conditions separated by the AND must be TRUE.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**Example**

Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example, which would fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 and the age is less than 25 years –

```
SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS WHERE SALARY > 2000 AND age < 25;
```

This would produce the following result –

ID	NAME	SALARY
6	Komal	4500.00
7	Muffy	10000.00

**The OR Operator**

The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

**Syntax**

The basic syntax of the OR operator with a WHERE clause is as follows –

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

SELECT column1, column2, columnN

FROM table\_name

WHERE [condition1] OR [condition2]...OR [conditionN]

You can combine N number of conditions using the OR operator. For an action to be taken by the SQL statement, whether it be a transaction or query, the only any ONE of the conditions separated by the OR must be TRUE.

**Example**

Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code block has a query, which would fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 and the age is less than 25 years.

SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS WHERE SALARY > 2000 OR age < 25;

This would produce the following result –

ID	NAME	SALARY
3	kaushik	2000.00
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

```
| 7 | Muffy | 10000.00 |
+---+-----+-----+
```

**The NOT Operator**

EX: SELECT Id, Name, salary FROM Customers WHERE NOT address = 'MP'

```
+---+-----+---+-----+-----+
| ID | NAME | AGE | ADDRESS | SALARY |
+---+-----+---+-----+-----+
| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |
| 2 | Khilan | 25 | Delhi | 1500.00 |
| 3 | kaushik | 23 | Kota | 2000.00 |
| 4 | Chaitali | 25 | Mumbai | 6500.00 |
| 5 | Hardik | 27 | Bhopal | 8500.00 |
| 7 | Muffy | 24 | Indore | 10000.00 |
+---+-----+---+-----+-----+
```

**LIKE Clause**

The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

The percent sign (%)

The underscore (\_)

The percent sign represents zero, one or multiple characters. The underscore represents a single number or character. These symbols can be used in combinations.

**Syntax**

The basic syntax of % and \_ is as follows –

SELECT FROM table\_name WHERE column LIKE 'XXXX%'

Or

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**Example**

Following is an example, which would display all the records from the CUSTOMERS table, where the SALARY starts with 200.

SQL> SELECT \* FROM CUSTOMERS WHERE SALARY LIKE '200%';

This would produce the following result –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
3	kaushik	23	Kota	2000.00

**TOP, LIMIT****Syntax**

The basic syntax of the TOP clause with a SELECT statement would be as follows.

SELECT TOP number|percent column\_name(s)

FROM table\_name

WHERE [condition]

The following query is an example on the SQL server, which would fetch the top 3 records from the CUSTOMERS table.

SQL> SELECT TOP 3 \* FROM CUSTOMERS;

This would produce the following result –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
3	kaushik	23	Kota	2000.00

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

```
+---+-----+---+-----+-----+
| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |
| 2 | Khilan | 25 | Delhi    | 1500.00 |
| 3 | kaushik | 23 | Kota     | 2000.00 |
+---+-----+---+-----+-----+
```

If you are using MySQL server, then here is an equivalent example –

```
SQL> SELECT * FROM CUSTOMERS LIMIT 3;
```

This would produce the following result –

```
+---+-----+---+-----+-----+
| ID | NAME   | AGE | ADDRESS | SALARY |
+---+-----+---+-----+-----+
| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |
| 2 | Khilan | 25 | Delhi    | 1500.00 |
| 3 | kaushik | 23 | Kota     | 2000.00 |
+---+-----+---+-----+-----+
```

If you are using an Oracle server, then the following code block has an equivalent example.

```
SQL> SELECT * FROM CUSTOMERS WHERE ROWNUM <= 3;
```

This would produce the following result –

```
+---+-----+---+-----+-----+
| ID | NAME   | AGE | ADDRESS | SALARY |
+---+-----+---+-----+-----+
| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |
| 2 | Khilan | 25 | Delhi    | 1500.00 |
| 3 | kaushik | 23 | Kota     | 2000.00 |
+---+-----+---+-----+-----+
```

**ORDER BY Clause**

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.

**Syntax**

The basic syntax of the ORDER BY clause is as follows –

SELECT column-list

FROM table\_name

[WHERE condition]

[ORDER BY column1, column2, .. columnN] [ASC | DESC];

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort that column should be in the column-list.

**Example**

The following code block has an example, which would sort the result in an ascending order by the NAME and the SALARY –

```
SQL> SELECT * FROM CUSTOMERS ORDER BY NAME, SALARY;
```

This would produce the following result –

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
3	kaushik	23	Kota	2000.00
2	Khilan	25	Delhi	1500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00
1	Ramesh	32	Ahmedabad	2000.00



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

The following code block has an example, which would sort the result in the descending order by NAME.

```
SQL> SELECT * FROM CUSTOMERS ORDER BY NAME DESC;
```

This would produce the following result –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
7	Muffy	24	Indore	10000.00
6	Komal	22	MP	4500.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
4	Chaitali	25	Mumbai	6500.00

**Group By**

The SQL GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

**Syntax**

The basic syntax of a GROUP BY clause is shown in the following code block. The GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clause if one is used.

```
SELECT column1, column2
```

```
FROM table_name
```

```
WHERE [ conditions ]
```

```
GROUP BY column1, column2
```

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

ORDER BY column1, column2

Example

Now, let us look at a table where the CUSTOMERS table has the following records with duplicate names –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Ramesh	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	kaushik	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now again, if you want to know the total amount of salary on each customer, then the GROUP BY query would be as follows –

SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS GROUP BY NAME;

This would produce the following result –

NAME	SUM(SALARY)
Hardik	8500.00
kaushik	8500.00
Komal	4500.00
Muffy	10000.00
Ramesh	3500.00

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**SQL | Views**

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

**Creating Views**

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic **CREATE VIEW** syntax is as follows –

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

You can include multiple tables in your SELECT statement in a similar way as you use them in a normal SQL SELECT query.

**Example**

Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

Following is an example to create a view from the CUSTOMERS table. This view would be used to have customer name and age from the CUSTOMERS table.

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM CUSTOMERS;
```

Now, you can query CUSTOMERS\_VIEW in a similar way as you query an actual table. Following is an example for the same.

```
SQL > SELECT * FROM CUSTOMERS_VIEW;
```

This would produce the following result.

```
+-----+-----+
| name  | age |
+-----+-----+
| Ramesh | 32 |
| Khilan | 25 |
| kaushik | 23 |
| Chaitali | 25 |
| Hardik | 27 |
| Komal | 22 |
| Muffy | 24 |
+-----+-----+
```

**Updating a View****Syntax:**

UPDATE **view-name**

set value

WHERE condition;

```
SQL > UPDATE CUSTOMERS_VIEW
```

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

```
SET AGE = 35
```

```
WHERE name = 'Ramesh';
```

This would ultimately update the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

```
+---+-----+---+-----+-----+
| ID | NAME   | AGE | ADDRESS | SALARY |
+---+-----+---+-----+-----+
| 1 | Ramesh | 35 | Ahmedabad | 2000.00 |
| 2 | Khilan | 25 | Delhi    | 1500.00 |
| 3 | kaushik | 23 | Kota     | 2000.00 |
| 4 | Chaitali | 25 | Mumbai   | 6500.00 |
| 5 | Hardik | 27 | Bhopal   | 8500.00 |
| 6 | Komal | 22 | MP       | 4500.00 |
| 7 | Muffy | 24 | Indore   | 10000.00 |
+---+-----+---+-----+-----+
```

**Inserting Rows into a View**

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

**Deleting Rows into a View**

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE = 22.

```
SQL > DELETE FROM CUSTOMERS_VIEW
```

```
WHERE age = 22;
```

This would ultimately delete a row from the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

```
+---+-----+---+-----+-----+
| ID | NAME   | AGE | ADDRESS | SALARY |
+---+-----+---+-----+-----+
| 1 | Ramesh | 35 | Ahmedabad | 2000.00 |
| 2 | Khilan | 25 | Delhi    | 1500.00 |
| 3 | kaushik | 23 | Kota     | 2000.00 |
| 4 | Chaitali | 25 | Mumbai   | 6500.00 |
| 5 | Hardik | 27 | Bhopal   | 8500.00 |
| 7 | Muffy  | 24 | Indore   | 10000.00 |
+---+-----+---+-----+-----+
```

**Dropping Views**

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below –

```
DROP VIEW view_name;
```

Following is an example to drop the CUSTOMERS\_VIEW from the CUSTOMERS table.

```
DROP VIEW CUSTOMERS_VIEW;
```

**DATABASE DESIGN****Relational Database Design by ER- and EER-to Relational Mapping**

Design a relational database schema Based on a conceptual schema design  
 Seven-step algorithm to convert the basic  
 ER model constructs into relations Additional steps for EER model

**Mapping ER/EER model to relational database**

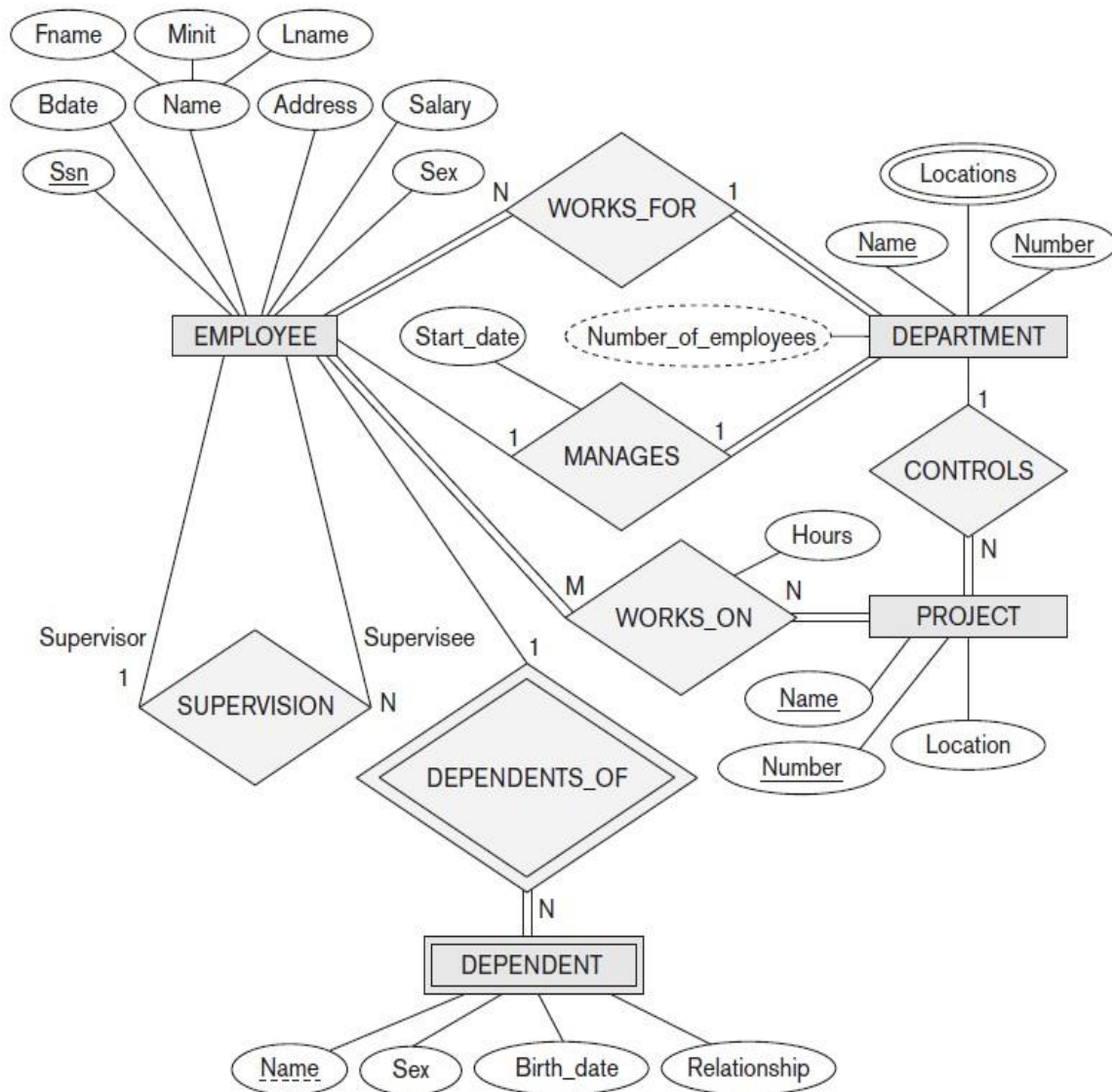
## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**Figure 9.1**

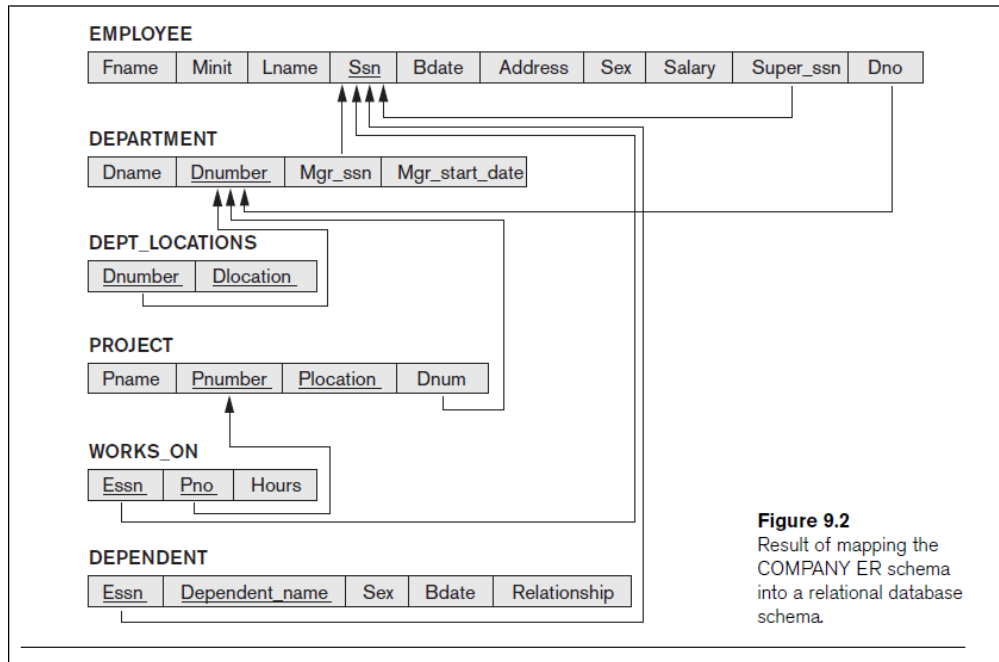
The ER conceptual schema diagram for the COMPANY database.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020



**Figure 9.2**  
Result of mapping the  
COMPANY ER schema  
into a relational database  
schema.

### ER-to-Relational Mapping Algorithm

#### COMPANY database example

Assume that the mapping will create tables with simple single-valued attributes

#### Step 1: Mapping of Regular Entity Types

For each regular entity type, create a relation  $R$  that includes all the simple attributes of  $E$  Called **entity relations**

- Each tuple represents an entity instance

#### Step 2: Mapping of Weak Entity Types

For each weak entity type, create a relation  $R$  and include all simple attributes of the entity type as attributes of  $R$

Include primary key attribute of owner as foreign key attributes of  $R$



## KARPAGAM ACADEMY OF HIGHER EDUCATION

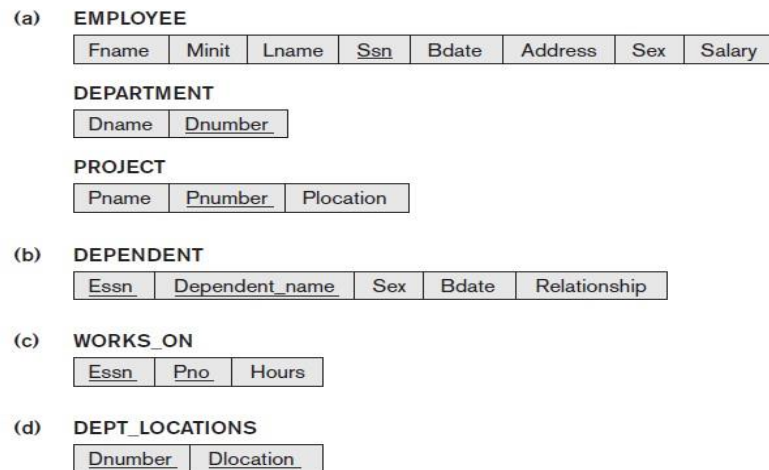
CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**Figure 9.3**

Illustration of some mapping steps.

- a. *Entity* relations after step 1.
- b. Additional *weak entity* relation after step 2.
- c. *Relationship* relation after step 5.
- d. Relation representing multivalued attribute after step 6.

**Step 3: Mapping of Binary 1:1 Relationship****Types**

For each binary 1:1 relationship type  
Identify relations that correspond to entity types participating in *R*

**Possible approaches:**

- Foreign key approach
- Merged relationship approach
- Crossreference or relationship relation approach

**Step 4: Mapping of Binary 1:N Relationship****Types**

For each regular binary 1:N relationship type

- Identify relation that represents participating entity type at *N*-side of relationship type
- Include primary key of other entity type as foreign key in *S*
- Include simple attributes of 1:N relationship type as attributes of *S*

**Alternative approach**

- Use the **relationship relation** (cross-reference) option as in the third option for binary 1:1 relationships

**Step 5: Mapping of Binary M:N Relationship****Types**

For each binary *M:N* relationship type

- Create a new relation *S*

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

- Include primary key of participating entity types as foreignkey attributes in S
- Include any simple attributes of  $M:N$  relationship type

**Step 6: Mapping of Multivalued Attributes**

For each multivalued attribute

- Create a new relation
- Primary key of  $R$  is the combination of  $A$  and  $K$
- If the multivalued attribute is composite, include its simple components

**Step 7: Mapping of  $N$ -ary Relationship Types**

For each  $n$ -ary relationship type  $R$

- Create a new relation  $S$  to represent  $R$
- Include primary keys of participating entity types as foreign keys
- Include any simple attributes as attributes

**Table 9.1** Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	<i>Entity</i> relation
1:1 or 1:N relationship type	Foreign key (or <i>relationship</i> relation)
$M:N$ relationship type	<i>Relationship</i> relation and <i>two</i> foreign keys
$n$ -ary relationship type	<i>Relationship</i> relation and $n$ foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

In a relational schema relationship, types are not represented explicitly  
Represented by having two attributes  $A$  and  $B$ : one a primary key and the other a foreign key

**Mapping EER Model Constructs to Relations****Extending ER-to-relational mapping Algorithm****Mapping of Specialization or Generalization**

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**Step 8: Options for Mapping Specialization or Generalization****Option 8A: Multiple relations—superclass and subclasses**

- For any specialization (total or partial, disjoint or overlapping)

**Option 8B: Multiple relations—subclass relations only**

- Subclasses are total
- Specialization has disjointness constraint

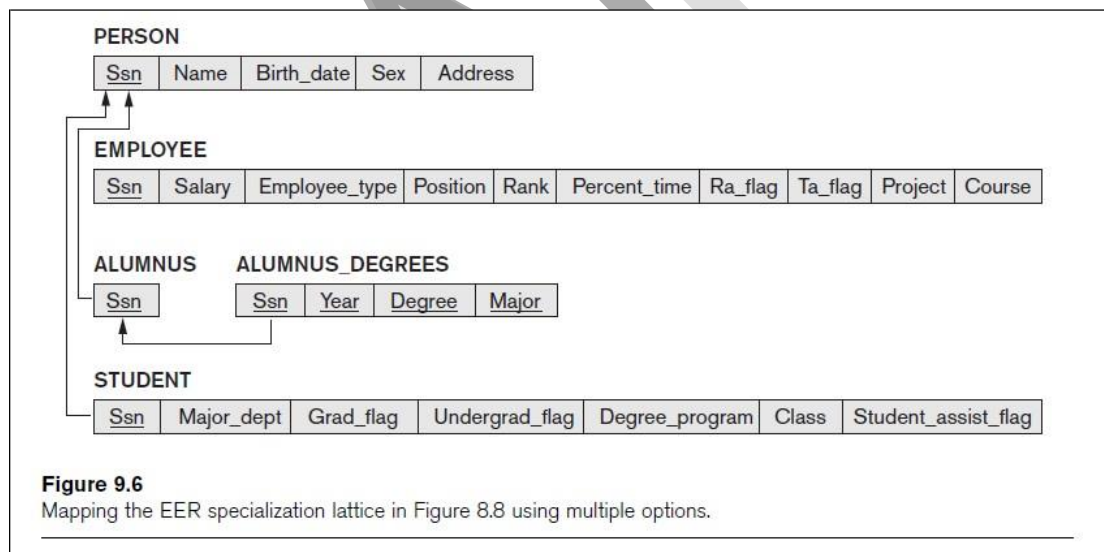
**Option 8C: Single relation with one type attribute**

- Type or discriminating attribute indicates subclass of tuple
- Subclasses are disjoint
- Potential for generating many NULL values if many specific attributes exist in the subclasses

**Option 8D: Single relation with multiple type attributes**

- Subclasses are overlapping
- Will also work for a disjoint specialization

Apply any of the options discussed in step 8 to a shared subclass

**Step 9: Mapping of Union Types (Categories)**

Defining superclasses have different keys

Specify a new key attribute

- **Surrogate key**

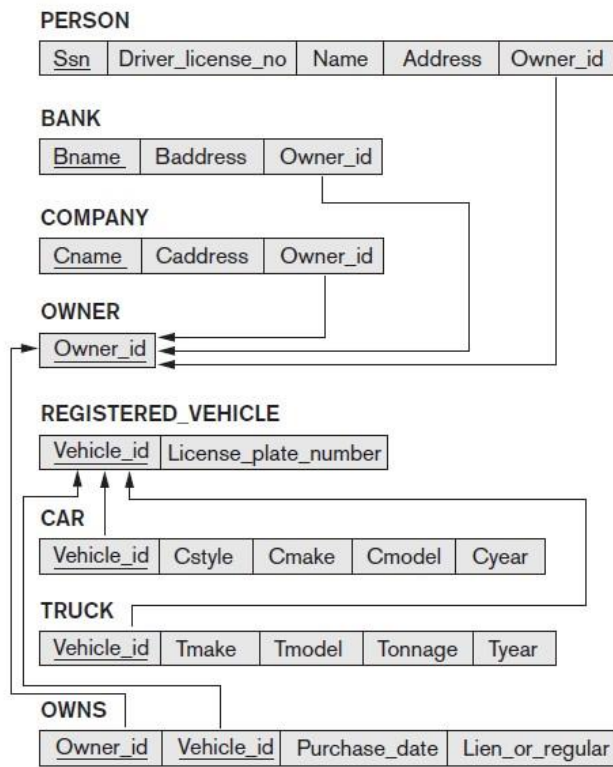
## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**Figure 9.7**

Mapping the EER categories (union types) in Figure 8.8 to relations.



### Summary

Map conceptual schema design in the ER model to a relational database schema

Algorithm for ER-to-relational mapping

Illustrated by examples from the COMPANY database

Include additional steps in the algorithm for mapping constructs from EER model into relational model

### What is functional dependency?

- **Functional Dependency** is a relationship that exists between multiple attributes of a relation.
- This concept is given by **E. F. Codd**.
- Functional dependency represents a formalism on the infrastructure of relation.
- It is a type of constraint existing between various attributes of a relation.
- It is used to define various normal forms.
- These dependencies are restrictions imposed on the data in database.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

- If P is a relation with A and B attributes, a functional dependency between these two attributes is represented as  $\{A \rightarrow B\}$ . It specifies that,

A	It is a determinant set.
B	It is a dependent attribute.
$\{A \rightarrow B\}$	A functionally determines B. B is a functionally dependent on A.

- Each value of A is associated precisely with one B value. A functional dependency is trivial if B is a subset of A.
- 'A' Functionality determines 'B'  $\{A \rightarrow B\}$  (Left hand side attributes determine the values of Right hand side attributes).

**For example:** <Employee> Table

EmpId

EmpName

- In the above <Employee> table, EmpName (employee name) is functionally dependent on EmpId (employee id) because the EmpId is unique for individual names.
- The EmpId identifies the employee specifically, but EmpName cannot distinguish the EmpId because more than one employee could have the same name.
- The functional dependency between attributes eliminates the repetition of information.
- It is related to a candidate key, which uniquely identifies a tuple and determines the value of all other attributes in the relation.

### Advantages of Functional Dependency

- Functional Dependency avoids data redundancy where same data should not be repeated at multiple locations in same database.
- It maintains the quality of data in database.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

- It allows clearly defined meanings and constraints of databases.
- It helps in identifying bad designs.
- It expresses the facts about the database design.

**Introduction to Axioms Rules**

- Armstrong's Axioms is a set of rules.
- It provides a simple technique for reasoning about functional dependencies.
- It was developed by William W. Armstrong in 1974.

**Various Axioms Rules****A. Primary Rules**

<b>Rule 1</b>	<b>Reflexivity</b> If A is a set of attributes and B is a subset of A, then A holds B. $\{A \rightarrow B\}$
<b>Rule 2</b>	<b>Augmentation</b> If A hold B and C is a set of attributes, then AC holds BC. $\{AC \rightarrow BC\}$ It means that attribute in dependencies does not change the basic dependencies.
<b>Rule 3</b>	<b>Transitivity</b> If A holds B and B holds C, then A holds C. If $\{A \rightarrow B\}$ and $\{B \rightarrow C\}$ , then $\{A \rightarrow C\}$ A holds B $\{A \rightarrow B\}$ means that A functionally determines B.

**B. Secondary Rules**

<b>Rule 1</b>	<b>Union</b> If A holds B and A holds C, then A holds BC. If $\{A \rightarrow B\}$ and $\{A \rightarrow C\}$ , then $\{A \rightarrow BC\}$
<b>Rule 2</b>	<b>Decomposition</b> If A holds BC and A holds B, then A holds C. If $\{A \rightarrow BC\}$ and $\{A \rightarrow B\}$ , then $\{A \rightarrow C\}$

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

<b>Rule 3</b>	<b>Pseudo Transitivity</b> If A holds B and BC holds D, then AC holds D. If $\{A \rightarrow B\}$ and $\{BC \rightarrow D\}$ , then $\{AC \rightarrow D\}$
---------------	--

Sometimes Functional Dependency Sets are not able to reduce if the set has following properties,

1. The Right-hand side set of functional dependency holds only one attribute.
2. The Left-hand side set of functional dependency cannot be reduced, it changes the entire content of the set.
3. Reducing any functional dependency may change the content of the set.

A set of functional dependencies with the above three properties are also called as **Canonical or Minimal**.

Trivial Functional Dependency

<b>Trivial</b>	If A holds B $\{A \rightarrow B\}$ , where A is a subset of B, then it is called a <b>Trivial Functional Dependency</b> . Trivial always holds Functional Dependency.
<b>Non-Trivial</b>	If A holds B $\{A \rightarrow B\}$ , where B is not a subset A, then it is called as a <b>Non-Trivial Functional Dependency</b> .
<b>Completely Non-Trivial</b>	If A holds B $\{A \rightarrow B\}$ , where $A \cap Y = \Phi$ , it is called as a <b>Completely Non-Trivial Functional Dependency</b> .

**Example:**

Consider relation E = (P, Q, R, S, T, U) having set of Functional Dependencies (FD).

$P \rightarrow Q$        $P \rightarrow R$

$QR \rightarrow S$        $Q \rightarrow T$

Calculate some members of Axioms are as follows,

1.  $P \rightarrow T$
2.  $PR \rightarrow S$
3.  $QR \rightarrow SU$
4.  $PR \rightarrow SU$

**Solution:**

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BSC CS  
COURSE CODE: 17CSU403

COURSE NAME: DATABASE MANAGEMENT SYSTEM  
UNIT: III  
BATCH-2017-2020

**1.  $P \rightarrow T$** 

In the above FD set,  $P \rightarrow Q$  and  $Q \rightarrow T$

**So, Using Transitive Rule: If  $\{A \rightarrow B\}$  and  $\{B \rightarrow C\}$ , then  $\{A \rightarrow C\}$**

$\therefore$  If  $P \rightarrow Q$  and  $Q \rightarrow T$ , then  $P \rightarrow T$ .

**$P \rightarrow T$**

**2.  $PR \rightarrow S$** 

In the above FD set,  $P \rightarrow Q$

As,  $QR \rightarrow S$

**So, Using Pseudo Transitivity Rule: If  $\{A \rightarrow B\}$  and  $\{BC \rightarrow D\}$ , then  $\{AC \rightarrow D\}$**

$\therefore$  If  $P \rightarrow Q$  and  $QR \rightarrow S$ , then  $PR \rightarrow S$ .

**$PR \rightarrow S$**

**3.  $QR \rightarrow SU$** 

In above FD set,  $QR \rightarrow S$  and  $QR \rightarrow U$

**So, Using Union Rule: If  $\{A \rightarrow B\}$  and  $\{A \rightarrow C\}$ , then  $\{A \rightarrow BC\}$**

**$QR \rightarrow SU$**

**4.  $PR \rightarrow SU$** 

**So, Using Pseudo Transitivity Rule: If  $\{A \rightarrow B\}$  and  $\{BC \rightarrow D\}$ , then  $\{AC \rightarrow D\}$**

$\therefore$  If  $PR \rightarrow S$  and  $PR \rightarrow U$ , then  $PR \rightarrow SU$ .

**$PR \rightarrow SU$**

**What is decomposition?**

- Decomposition is the process of breaking down in parts or elements.
- It replaces a relation with a collection of smaller relations.
- It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.
- If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

Properties of Decomposition



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

**Following are the properties of Decomposition,**

1. Lossless Decomposition
2. Dependency Preservation
3. Lack of Data Redundancy

**1. Lossless Decomposition**

- Decomposition must be lossless. It means that the information should not get lost from the relation that is decomposed.
- It gives a guarantee that the join will result in the same relation as it was decomposed.

**Example:**

Let's take 'E' is the Relational Schema. With instance 'e' is decomposed into: E1, E2, E3

- In the above example, it means that, if natural joins of all the decomposition give the original relation, then it is said to be lossless join decomposition.

**Example: <Employee\_Department> Table**

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Bangalore	25000	D005	Human Resource

- Decompose the above relation into two relations to check whether a decomposition is lossless or lossy.
- Now, we have decomposed the relation that is Employee and Department.

Eid	Ename	Age	City	Salary
E001	ABC	29	Pune	20000

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

E002	PQR	30	Pune	30000
E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Bangalore	25000

**Relation 1 : <Employee> Table**

- Employee Schema contains (Eid, Ename, Age, City, Salary).

**Relation 2 : <Department> Table**

Deptid	Eid	DeptName
D001	E001	Finance
D002	E002	Production
D003	E003	Sales
D004	E004	Marketing
D005	E005	Human Resource

- Department Schema contains (Deptid, Eid, DeptName).
- So, the above decomposition is a Lossless Join Decomposition, because the two relations contains one common field that is 'Eid' and therefore join is possible.
- Now apply natural join on the decomposed relations.

**Employee ⋈ Department**

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: II BSC CS

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CSU403

UNIT: III

BATCH-2017-2020

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS : II.B.Sc CS

BATCH : 2016 - 2019

COURSE NAME : DATABASE MANAGEMENT SYSTEM

COURSE CODE : 16CSU403

**UNIT III**

E005	STU	32	Bangalore	25000	D005	Human Resource
------	-----	----	-----------	-------	------	----------------

Hence, the decomposition is Lossless Join Decomposition.

- If the <Employee> table contains (Eid, Ename, Age, City, Salary) and <Department> table contains (Deptid and DeptName), then it is not possible to join the two tables or relations, because there is no common column between them. And it becomes **Lossy Join Decomposition**.

**2. Dependency Preservation**

- Dependency is an important constraint on the database.
- Every dependency must be satisfied by at least one decomposed table.
- If  $\{A \rightarrow B\}$  holds, then two sets are functional dependent. And, it becomes more useful for checking the dependency easily if both sets in a same relation.
- This decomposition property can only be done by maintaining the functional dependency.
- In this property, it allows to check the updates without computing the natural join of the database structure.

**3. Lack of Data Redundancy**

- Lack of Data Redundancy is also known as a **Repetition of Information**.
- The proper decomposition should not suffer from any data redundancy.
- The careless decomposition may cause a problem with the data.
- The lack of data redundancy property may be achieved by Normalization process.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**

**BATCH-2017-2020**

**POSSIBLE QUESTIONS****2 MARK QUESTIONS**

1. What is a view?
2. Define referential integrity.
3. Mention the difference between drop and truncate command in SQL.
4. Define Triggers.
5. Define constraints
6. Write any seven attributes of student entity.
7. Define Concatenation operator.
8. Define functional dependencies.
9. Define candidate key with example.
10. Write syntax for any 2 set operators.
11. Define triggers with example.

**6 MARK QUESTIONS**

1. What is Functional Dependency? Explain types and properties of FD's.
2. Explain the following with example
  - i. i) INSERT
  - II) UPDATE
  - III) DROP
3. Briefly explain about views of data with proper example.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

4. Discuss Lossless decomposition with neat sketch
5. Explain with example.                      i) Complex Queries    ii) Views
6. How to Map ER/EER model to relational database?
7. Explain the advantages of decomposition? Discuss the problems faced in decomposition.
8. Explain with example i) Aggregation functions. li) UPDATE Statements in SQL

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**CLASS: II BSC CS**  
**COURSE CODE: 17CSU403**

**COURSE NAME: DATABASE MANAGEMENT SYSTEM**  
**UNIT: III**  
**BATCH-2017-2020**

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
**DEPARTMENT OF COMPUTER SCIENCE, CA & IT**  
**II BSc CS- Batch 2017-2020**  
**PART-A OBJECTIVE TYPE/ One mark -Online Examination**  
**DATABASE MANAGEMENT SYSTEM(17CSU403)**

**UNIT-III**

sno	Questions	opt1	opt2	opt3	opt4	Answer
1	What does SQL stand for?	Structured Query Language	Strong Question Language	Structured Question Language	Structure Query Language	Structured Query Language
2	Create,Alter,Drop commands are _____ language commands	DML	TCL	DCL	DDL	DDL
3	Insert,Select,Update,delete commands are _____ language commands	DML	TCL	DCL	DDL	DML
4	Selection Operation is used to ____ from a relation	Select the columns	Select the rows	Select the table	none.	Select the rows
5	_____ command is used to remove the table definition information	Delete	Remove	Destroy	Drop	Drop
6	_____ is used to modify the structure of an existing table.	Modify	Alter	Change	Recreate	Alter
7	View can be dropped using _____ command	Delete View	Remove View	Replace View	Drop View	Drop View
8	_____ columns are not allowed to contain null values	Primary Key	Candidate key	foreign Key	Unique Key	Primary Key
9	_____ are valuable to give the security to our original table	Original table	Duplicate table	Views	Tables	Views
10	_____ clause is used to modify a particular row.	Update	Modify	Alter	Where	Where
11	_____ 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	check	integrity Constraint
12	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
13	_____ is used to refer the primary key in another entity	Candidate key	referential key	foreign key	Primary key	foreign key
14	_____ value is unknown or not applicable	not null	zero	unknown	null	null
15	_____ statement is used to define a new table.	Create	Produce	Insert	Add	Create
16	Rows are inserted using the _____ command	Create	Insert	Add	Make	Insert
17	rows are deleted using the _____ command	Delete	drop	remove	alter	Delete
18	Modify the column values in an existing row using _____ command	Modify	Alter	Update	Change	Update
19	_____ command is used to remove the table definition information	Delete	Remove	Destroy	Drop	Drop
20	_____ is used to modify the structure of an existing table.	Modify	Alter	Change	Recreate	Alter
21	View can be dropped using _____ command	Delete View	Remove View	Replace View	Drop View	Drop View
22	Duplicates are eliminated by using _____ keyword.	Remove	Distinct	RM	Redundant	Distinct
23	_____ is a query that has another query embedded within it.	Query	Subquery	QBE	QUEL	Subquery
24	_____ to calculate the number of values in the Column.	Count	aggregate	Cal	Calculate	Count
25	_____ is used to calculate the sum of all values in the column	Total	Sum	Count	Collection	Sum
26	_____ is used to calculate the average of all values in the column	Total	Sum	Average	avg	avg
27	Which function is used to extract the maximum values in the relations?	max	maximum	excess	large	max
28	Which function is used to extract the maximum values in the relations?	Small	minimum	lower	min	min
29	Which clause is used, when we are using Group by clause, instead of where clause?	where	Having	Distinct	Group	Having
30	_____ is used when the column value is either unknown or inapplicable.	zero	all	Empty	null	null

31	keyword specifies that the join condition is equality on all common attributes and the where clause is not required	full outerjoin	left outer join	Natural	Right outerjoin	Natural
32	constraints for a single table.	Assertion	table constraints	default	union	table constraints
33	keyword is used to assign a default value with a domain.	Static	Default	Permanent	distinct	Default
34	Which constraint is used to check the ranges in the column values?	range	verify	check	condition	check
35	operator is another set comparison operator such as IN.	Exists	IN	avail	present	Exists
36	Which keyword is similar to NOT IN?	Exist	IN	Not EXIST	Except	Except
37	Which keyword is similar to IN?	Exist	IN	Not EXIST	Except	Exist
38	With SQL, how can you return the number of records in the "Persons" table?	COLUMNS(*) FROM Persons	COUNT() FROM Persons	COUNT(*) FROM Persons	COLUMNS() FROM Persons	COUNT(*) FROM Persons
39	How can you change "Hansen" into "Nilsen" in the "LastName" column in the Persons table?	SET LastName='Nilsen'	SET LastName='Hanse	SET LastName='Hansen	SET LastName='Nilsen	SET LastName='Nilsen
40	table named "Persons" sorted descending by "FirstName"?	Persons SORT 'FirstName' DESC	Persons ORDER BY FirstName	Persons ORDER FirstName DESC	Persons SORT BY 'FirstName'	Persons ORDER BY FirstName
41	Which SQL keyword is used to sort the result-set?	ORDER	SORT	SORT BY	ORDER BY	ORDER BY
42	With SQL, how do you select all the records from a table named "Persons" where the value of the column	SELECT * FROM Persons WHERE	SELECT * FROM Persons WHERE	SELECT * FROM Persons WHERE	SELECT * FROM Persons WHERE	SELECT * FROM Persons WHERE
43	You need to calculate the total of all salaries in the accounting department. Which group function should	SUM	COUNT	TOTAL	LARGEST	SUM
44	SELECT ROUND (45.953, -1), TRUNC (45.936, 2) FROM dual; which values are displayed?	46 and 45	46 and 45.93	50 and 45.93	50 and 45.9	46 and 45.93
45	Select operator is not a unary operator	TRUE	FALSE	not opeator	operator	FALSE
46	Project operator chooses subset of attributes or columns of a relation	TRUE	FALSE	not opeator	operator	TRUE
47	..... database is used as template for all databases created.	Master	Model	Tempdb	None of the above	Model
48	subsystem is to ensure that only valid values can be assigned to each data items. This is referred to as	Data Security	Domain access	Data Control	Domain Integrity	Domain Integrity
49	..... operator is basically a join followed by a project on the attributes of first relation.	Join	Semi-Join	Full Join	Inner Join	Semi-Join
50	Which of the following is not a binary operator in relational algebra?	Join	Semi-Join	Assignment	Project	Project
51	DBMS ..... duplication and ensures the consistency and validity of the database.	Increases	Skips	Does not reduce	Reduces	Reduces
52	Which of the following is/are the DDL statements?	Create	Drop	Alter	All of the above	All of the above
53	In snapshot, ..... clause tells oracle how long to wait between refreshes.	Complete	Force	Next	Refresh	Refresh
54	allowed in columns and is the standard mechanism for enforcing database integrity.	Column	Constraint	Index	Trigger	Constraint
55	If B is a subset of A ,then A→B indicates	Reflexivity	Augmentation	Transitivity	Decomposition	Reflexivity
56	If A→B, then AC→BC indicates	Reflexivity	Augmentation	Transitivity	Decomposition	Augmentation
57	If A→B and B→C, then A→C indicates	Reflexivity	Augmentation	Transitivity	Decomposition	Transitivity
58	If A→BC, then A→B and A→C indicates	Reflexivity	Augmentation	Transitivity	Decomposition	Decomposition
59	If A→B and A→C, then A→BC indicates	Reflexivity	Augmentation	Transitivity	Union	Union
60	If A→B and C→D, then AC→BD indicates	Composition	Augmentation	Transitivity	Union	Composition



**UNIT-4**

**SYLLABUS**

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

**What is Database Design?**

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems

It helps produce database systems

1. That meet the requirements of the users
2. Have high performance.

The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

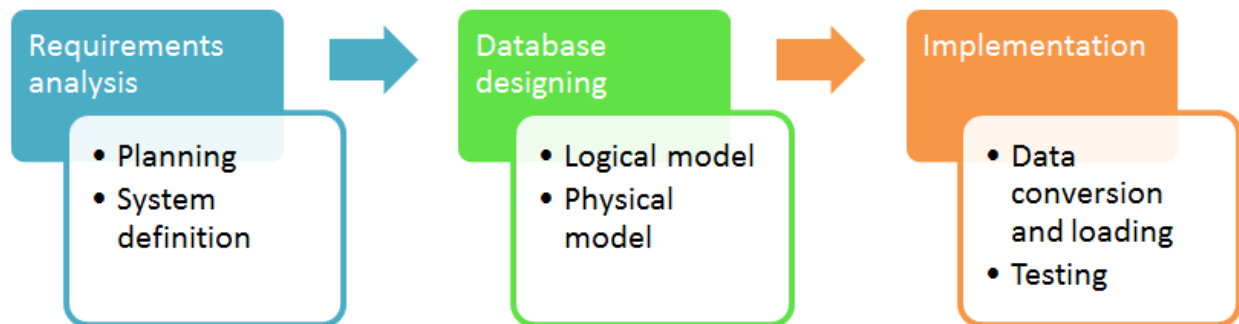
**Why Database Design is Important?**

Database designing is crucial to **high performance** database system.

Apart from improving the performance, properly designed database are easy to maintain, improve data consistency and are cost effective in terms of disk storage space.

Note , the genius of a database is in its design . Data operations using SQL is relatively simple

### Database development life cycle



The database development life cycle has a number of stages that are followed when developing database systems.

The steps in the development life cycle do not necessarily have to be followed religiously in a sequential manner.

On small database systems, the database system development life cycle is usually very simple and does not involve a lot of steps.

In order to fully appreciate the above diagram, let's look at the individual components listed in each step.

#### **Requirements analysis**

- **Planning** - This stage concerns with planning of entire Database Development Life Cycle. It takes into consideration the Information Systems strategy of the organization.
- **System definition** - This stage defines the scope and boundaries of the proposed database system.

#### **Database designing**

- **Logical model** - This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.
- **Physical model** - This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

#### **Implementation**

- **Data conversion and loading** - this stage is concerned with importing and converting data from the old system into the new database.

- **Testing** - this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

### Two Types of Database Techniques

1. **Normalization**
2. **ER Modeling**

Let's study them one by one

### What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples

#### What is Normalization?

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.

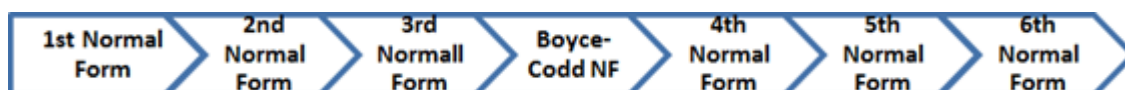
It divides larger tables to smaller tables and links them using relationships.

In this tutorial, you will learn-

- Database Normal Forms
- 1NF Rules
- What is a KEY?
- What is Composite Key
- 2NF Rules
- Database - Foreign Key
- What are transitive functional dependencies?
- 3NF Rules
- Boyce-Codd Normal Form (BCNF)

The inventor of the relational model Edgar Codd proposed the theory of normalization with the introduction of First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined with Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.

Theory of Data Normalization in SQL is still being developed further. For example, there are discussions even on 6<sup>th</sup> Normal Form. **However, in most practical applications, normalization achieves its best in 3<sup>rd</sup> Normal Form.** The evolution of Normalization theories is illustrated below-



**Database Normalization Examples -**

Assume a video library maintains a database of movies rented out. Without any normalization, all information is stored in one table as shown below.

Full Names	Physical Address	Movies rented	Salutation	Category
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.	Action, Action
Robert Phil	3 <sup>rd</sup> Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.	Romance, Romance
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.	Action

**Database Normal Forms**

Now let's move into 1<sup>st</sup> Normal Forms

**1NF (First Normal Form) Rules**

- Each table cell should contain a single value.
- Each record needs to be unique.

The above table in 1NF-

**1NF Example**

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3 <sup>rd</sup> Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3 <sup>rd</sup> Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

**What is a KEY?**

A KEY is a value used to identify a record in a table uniquely. A KEY could be a single column or combination of multiple columns

Note: Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

What is a Primary Key?



**Primary Key**

A primary is a single column value used to identify a database record uniquely.

It has following attributes

- A primary key cannot be NULL
- A primary key value must be unique
- The primary key values cannot be changed
- The primary key must be given a value when a new record is inserted.

### What is Composite Key?

A composite key is a primary key composed of multiple columns used to identify a record uniquely

In our database, we have two people with the same name Robert Phil, but they live in different places.

<b>Composite Key</b>			
Robert Phil	3 <sup>rd</sup> Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

**Names are common. Hence you need name as well Address to uniquely identify a record.**

Hence, we require both Full Name and Address to identify a record uniquely. That is a composite key.

Let's move into second normal form 2NF

### 2NF (Second Normal Form) Rules

- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key

It is clear that we can't move forward to make our simple database in 2<sup>nd</sup> Normalization form unless we partition the table above.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 <sup>rd</sup> Street 34	Mr.
3	Robert Phil	5 <sup>th</sup> Avenue	Mr.

Table 1

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Table 2

We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

We have introduced a new column called Membership\_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

### **Database - Foreign Key**

In Table 2, Membership\_ID is the Foreign Key

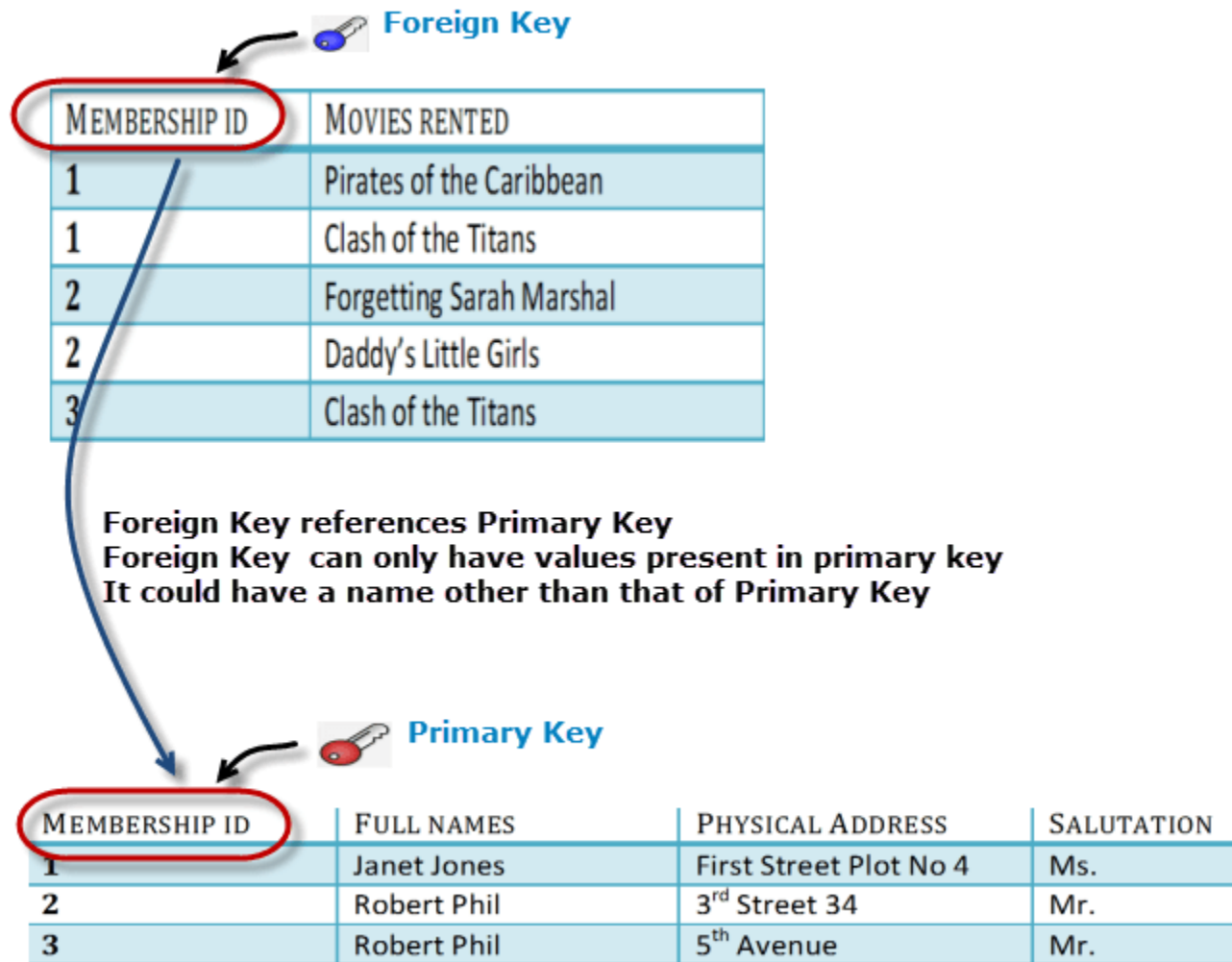
MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans



Foreign Key

Foreign Key references the primary key of another Table! It helps connect your Tables

- A foreign key can have a different name from its primary key
- It ensures rows in one table have corresponding rows in another
- Unlike the Primary key, they do not have to be unique. Most often they aren't
- Foreign keys can be null even though primary keys can not



Why do you need a foreign key?

Suppose an idiot inserts a record in Table B such as

You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.



Insert a record in Table 2 where Member ID = 101

MEMBERSHIP ID	MOVIES RENTED
101	Mission Impossible

But Membership ID 101 is not present in Table 1

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 <sup>rd</sup> Street 34	Mr.
3	Robert Phil	5 <sup>th</sup> Avenue	Mr.

Database will throw an **ERROR**. This helps in referential integrity

The above problem can be overcome by declaring membership id from Table2 as foreign key of membership id from Table1

Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

### What are transitive functional dependencies?

A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change

Consider the table 1. Changing the non-key column Full Name may change Salutation.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 <sup>rd</sup> Street 34	Mr.
3	Robert Phil	5 <sup>th</sup> Avenue	Mr.

Change in Name → May Change Salutation

Let's move into 3NF

### 3NF (Third Normal Form) Rules

- Rule 1- Be in 2NF
- Rule 2- Has no transitive functional dependencies

To move our 2NF table into 3NF, we again need to again divide our table.

### 3NF Example

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No4	2
2	Robert Phil	3 <sup>rd</sup> Street 34	1
3	Robert Phil	5 <sup>th</sup> Avenue	1

TABLE 1

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Table 2

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

Table 3

We have again divided our tables and created a new table which stores Salutations.

There are no transitive functional dependencies, and hence our table is in 3NF

In Table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3

Now our little example is at a level that cannot further be decomposed to attain higher forms of normalization. In fact, it is already in higher normalization forms. Separate efforts for moving into next levels of normalizing data are normally needed in complex databases. However, we will be discussing next levels of normalizations in brief in the following.

### **Boyce-Codd Normal Form (BCNF)**

Even when a database is in 3<sup>rd</sup> Normal Form, still there would be anomalies resulted if it has more than one **Candidate Key**.

Sometimes BCNF is also referred as **3.5 Normal Form**.

### **4NF (Fourth Normal Form) Rules**

If no database table instance contains two or more, independent and multivalued data describing the relevant entity, then it is in 4<sup>th</sup> Normal Form.

### **5NF (Fifth Normal Form) Rules**

A table is in 5<sup>th</sup> Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

### **6NF (Sixth Normal Form) Proposed**

6<sup>th</sup> Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6<sup>th</sup> Normal Form in the near future...

That's all to Normalization!!!

### **Summary**

- Database designing is critical to the successful implementation of a database management system that meets the data requirements of an enterprise system.
- Normalization helps produce database systems that are cost-effective and have better security models.
- Functional dependencies are a very important component of the normalize data process

- Most database systems are normalized database up to the third normal forms.
- A primary uniquely identifies are record in a Table and cannot be null
- A foreign key helps connect table and references a primary key

### **Transaction Processing**

Earlier you have learned about the functions that a Database Management System (DBMS) should offer database users. Among these there are 3 closely related functions that are intended to ensure that the database is reliable and remains in a steady state, namely transaction support, concurrency controlling and recovery services. This reliability and consistency must be maintained in the presence of failures of both hardware and software components and when several users are accessing the database. In this chapter, you will be concentrating on a transaction and ACID property of DBMS.

What is Transaction?

A transaction is an action, or series of actions that are being performed by a single user or application program, which reads or updates the contents of the database.

A transaction can be defined as a logical unit of work on the database. This may be an entire program, a piece of a program or a single command (like the SQL commands such as INSERT or UPDATE) and it may engage in any number of operations on the database. In the database context, the execution of an application program can be thought of as one or more transactions with non-database processing taking place in between.

#### **Example of a Transaction in DBMS**

A simple example of a transaction will be dealing with the bank accounts of two users let say Karlos and Ray. A simple transaction of moving an amount of 5000 from Karlos to Ray engages many low-level jobs. As the amount of Rs. 5000 gets transferred from the Karlos's account to Ray's account, a series of tasks gets performed in the background of the screen.

This very simple and small transaction includes several steps: decrease Karlos's bank account from 5000:

Open\_Acc (Karlos)

OldBal = Karlos.bal

NewBal = OldBal - 5000

Ram.bal = NewBal

CloseAccount(Karlos)

Simply you can say, the transaction involves many tasks, such as opening the account of Karlos, reading the old balance, decreasing the specific amount 5000 from that account, saving new balance to an account of Karlos and finally closing the transaction session.

For adding amount 5000 in Ray's account the same sort of tasks needs to be done:

OpenAccount(Ray)

Old\_Bal = Ray.bal

NewBal = OldBal + 1000

Ahmed.bal = NewBal

CloseAccount(B)

### Properties of Transaction

There are properties that all transactions should follow and possess. The four basic are in combination termed as ACID properties. ACID properties and its concepts of a transaction are put forwarded by Haerder and Reuter in the year 1983. The ACID has a full form and is as follows:

- **Atomicity:** The 'all or nothing' property. A transaction is an indivisible entity that is either performed in its entirety or will not get performed at all. This is the responsibility or duty of the recovery subsystem of the DBMS to ensure atomicity.
- **Consistency:** A transaction must alter the database from one steady state to another steady state. This is the responsibility of both the DBMS and the application developers to

make certain consistency. The DBMS can ensure consistency by putting into effect all the constraints that have been particularly on the database schema such as integrity and enterprise constraints.

- **Isolation:** Transactions that are executing independently of one another is the primary concept followed by isolation. In other words, the frictional effects of incomplete transactions should not be visible or come into notice to other transactions going on simultaneously. It is the responsibility of the concurrency control sub-system to ensure adapting the isolation.
- **Durability:** The effects of a successfully accomplished transaction are permanently recorded in the database and must not get lost or vanished due to a subsequent failure. So this becomes the responsibility of the recovery sub-system to ensure durability.

### **Serializability**

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

- **Schedule** – A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
- **Serial Schedule** – It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

In a multi-transaction environment, serial schedules are considered as a benchmark. The execution sequence of an instruction in a transaction cannot be changed, but two transactions can have their instructions executed in a random fashion. This execution does no harm if two transactions are mutually independent and working on different segments of data; but in case these two transactions are working on the same data, then the results may vary. This ever-varying result may bring the database to an inconsistent state.

To resolve this problem, we allow parallel execution of a transaction schedule, if its transactions are either serializable or have some equivalence relation among them.

### **Equivalence Schedules**

An equivalence schedule can be of the following types –

### Result Equivalence

If two schedules produce the same result after execution, they are said to be result equivalent. They may yield the same result for some value and different results for another set of values. That's why this equivalence is not generally considered significant.

### View Equivalence

Two schedules would be view equivalence if the transactions in both the schedules perform similar actions in a similar manner.

For example –

- If T reads the initial data in S1, then it also reads the initial data in S2.
- If T reads the value written by J in S1, then it also reads the value written by J in S2.
- If T performs the final write on the data value in S1, then it also performs the final write on the data value in S2.

### Conflict Equivalence

Two schedules would be conflicting if they have the following properties –

- Both belong to separate transactions.
- Both accesses the same data item.
- At least one of them is "write" operation.

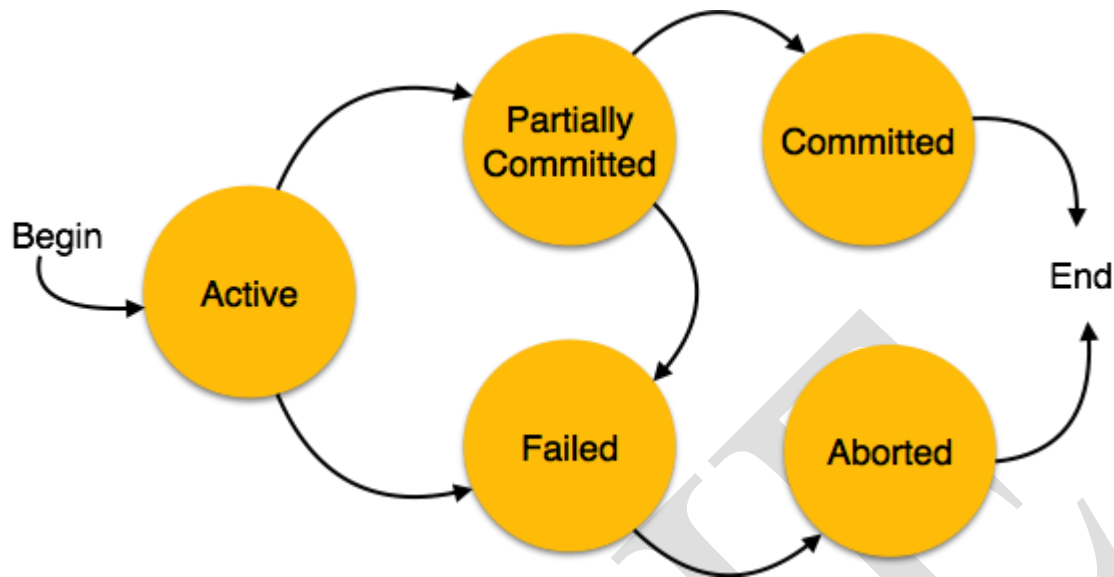
Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if –

- Both the schedules contain the same set of Transactions.
- The order of conflicting pairs of operation is maintained in both the schedules.

**Note** – View equivalent schedules are view serializable and conflict equivalent schedules are conflict serializable. All conflict serializable schedules are view serializable too.

### States of Transactions

A transaction in a database can be in one of the following states –



- **Active** – In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** – When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** – A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** – If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts –
  - Re-start the transaction
  - Kill the transaction
- **Committed** – If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

## concurrency control



In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories –

- Lock based protocols
- Time stamp based protocols

### **Lock-based Protocols**

Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it. Locks are of two kinds –

- **Binary Locks** – A lock on a data item can be in two states; it is either locked or unlocked.
- **Shared/exclusive** – This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

There are four types of lock protocols available –

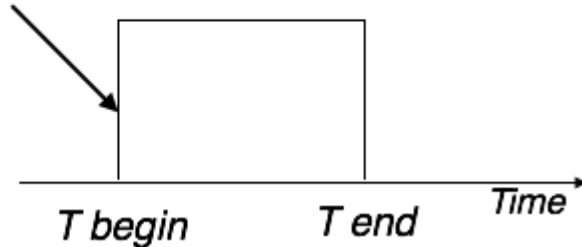
### **Simplistic Lock Protocol**

Simplistic lock-based protocols allow transactions to obtain a lock on every object before a 'write' operation is performed. Transactions may unlock the data item after completing the 'write' operation.

### **Pre-claiming Lock Protocol**

Pre-claiming protocols evaluate their operations and create a list of data items on which they need locks. Before initiating an execution, the transaction requests the system for all the locks it needs beforehand. If all the locks are granted, the transaction executes and releases all the locks when all its operations are over. If all the locks are not granted, the transaction rolls back and waits until all the locks are granted.

Lock acquisition  
phase

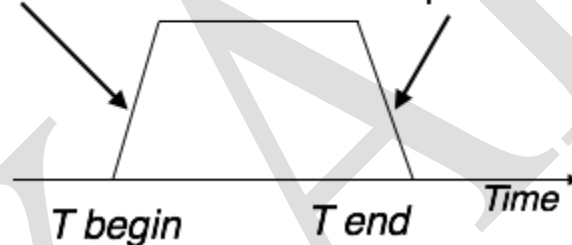


### Two-Phase Locking 2PL

This locking protocol divides the execution phase of a transaction into three parts. In the first part, when the transaction starts executing, it seeks permission for the locks it requires. The second part is where the transaction acquires all the locks. As soon as the transaction releases its first lock, the third phase starts. In this phase, the transaction cannot demand any new locks; it only releases the acquired locks.

Lock acquisition  
phase

releasing  
phase

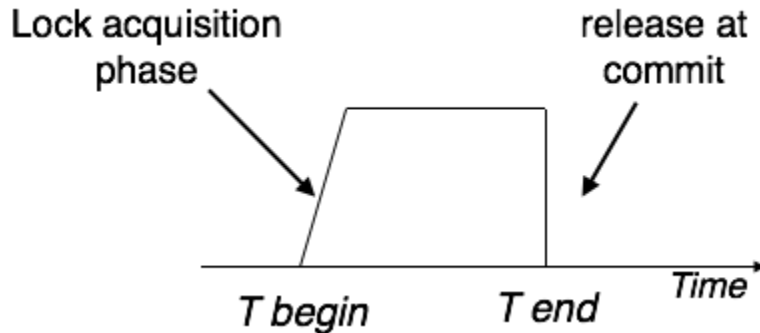


Two-phase locking has two phases, one is **growing**, where all the locks are being acquired by the transaction; and the second phase is **shrinking**, where the locks held by the transaction are being released.

To claim an exclusive (write) lock, a transaction must first acquire a shared (read) lock and then upgrade it to an exclusive lock.

### Strict Two-Phase Locking

The first phase of Strict-2PL is same as 2PL. After acquiring all the locks in the first phase, the transaction continues to execute normally. But in contrast to 2PL, Strict-2PL does not release a lock after using it. Strict-2PL holds all the locks until the commit point and releases all the locks at a time.



Strict-2PL does not have cascading abort as 2PL does.

### **Timestamp-based Protocols**

The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp.

Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.

Every transaction has a timestamp associated with it, and the ordering is determined by the age of the transaction. A transaction created at 0002 clock time would be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one.

In addition, every data item is given the latest read and write-timestamp. This lets the system know when the last 'read and write' operation was performed on the data item.

### **Timestamp Ordering Protocol**

The timestamp-ordering protocol ensures serializability among transactions in their conflicting read and write operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

- The timestamp of transaction  $T_i$  is denoted as  $TS(T_i)$ .
- Read time-stamp of data-item  $X$  is denoted by  $R\text{-timestamp}(X)$ .
- Write time-stamp of data-item  $X$  is denoted by  $W\text{-timestamp}(X)$ .

Timestamp ordering protocol works as follows –

- **If a transaction  $T_i$  issues a read( $X$ ) operation –**

- If  $TS(T_i) < W\text{-timestamp}(X)$ 
  - Operation rejected.
- If  $TS(T_i) \geq W\text{-timestamp}(X)$ 
  - Operation executed.
- All data-item timestamps updated.
- **If a transaction  $T_i$  issues a write( $X$ ) operation –**
  - If  $TS(T_i) < R\text{-timestamp}(X)$ 
    - Operation rejected.
  - If  $TS(T_i) < W\text{-timestamp}(X)$ 
    - Operation rejected and  $T_i$  rolled back.
  - Otherwise, operation executed.

#### Thomas' Write Rule

This rule states if  $TS(T_i) < W\text{-timestamp}(X)$ , then the operation is rejected and  $T_i$  is rolled back.

Time-stamp ordering rules can be modified to make the schedule view serializable.

Instead of making  $T_i$  rolled back, the 'write' operation itself is ignored.

#### Possible Questions

##### Part – B (2 Mark)

1. What are the advantages of normalization?
2. List the ACID properties in DBMS.
3. Define concurrency control.
4. How to convert a table from 1NF to 2NF?
5. Mention the difference between drop and truncate command in SQL.

**Part – C (6 Mark)**

1. Illustrate with syntax and example of first THREE Normal forms
2. List and explain ACID properties with example
3. State BCNF. How does it differ from 3NF?
4. Explain the time stamp based protocol for concurrency control in a DBMS.
5. What is transaction? Mention the desirable properties of a transaction.
6. State BCNF. How does it differ from 3NF?
7. Discuss any three Normal Forms with proper example.
8. Explain in detail about timestamp based concurrency control techniques.
9. Draw transaction state diagram and describe each state that a transaction goes through during its execution.
10. Explain briefly about 3NF, 4NF with suitable examples?

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
**DEPARTMENT OF COMPUTER SCIENCE, CA & IT**  
**II BSc CS- Batch 2017-2020**  
**PART-A OBJECTIVE TYPE/ One mark -Online Examination**  
**DATABASE MANAGEMENT SYSTEM(17CSU403)**

**UNIT-V**

sno	Questions	opt1	opt2	opt3	opt4
1	_____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
2	BCNF stands for _____	Boyce Codd Normal Form	Boy Codd Normal Form	Basic Codd Normal Form	Basic Codd Normalization Form
3	Boye Codd Normal Form (BCNF) is needed when	two non-key attributes are dependent	there is more than one possible composite key	there are two or more possible composite keys	there are two possible keys and they are
4	A relation is said to be in BCNF when	overlapping composite keys	it has no composite key	it has no multivalued dependencies	it has no overlapping composite keys
5	A 3 NF relation is converted to BCNF by	removing composite keys	removing multivalued dependencies	attributes of overlapping dependencies	non-key attributes are dependent
6	BCNF is needed because	otherwise tuples may be duplicated	when a data is deleted tuples may be deleted	updating is otherwise difficult	when there is dependent attributes in
7	Fourth normal form (4 NF) relations are needed when	there are multivalued dependencies	there are more than one composite keys	there are two or more overlapping composite keys	there are multivalued dependencies
8	A 3 NF relation is split into 4 NF	by removing overlapping composite keys	by splitting into relations which do not have complete key	removing multivalued dependencies	by putting dependent non-key attributes in
9	A third Normal Form (3 NF) relation should	be in 2 NF	not have complete key	not be 1 NF	have non-key attributes
10	The process of normalization	is automatic using a computer	requires one to understand dependencies	is manual and requires semantic information	is finding the key of a relation
11	A relation is said to be in 1NF if	there is no duplication of data	there are no composite attributes in	there are only a few composite attributes	all attributes are of uniform type
12	The number of normal forms which has been proposed and discussed in the book are	3	4	5	6
13	A relation which is in a higher normal form	implies that it also qualifies to be in lower	does not necessarily satisfy the	is included in the lower normal form	is independent of lower normal forms
14	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
15	Given the following relation vendor order (vendor no, order no, vendor name, qty supplied, price/unit) the second normal form	vendor (vendor no, vendor name)	vendor (vendor no, vendor name)	vendor (vendor no, vendor name) order (order no, qty)	vendor (vendor no, vendor name, qty)
16	_____ technique is used to reduce the redundancy	Closure set	Decomposition	Normalization	Null Values
17	3NF is also referred to as _____	LCNF	BCNF	information-preserving	desirable form
18	Projection-join normal form also known as	1NF	2NF	3NF	5NF
19	_____ 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
20	A relation is in _____ if and only if the nonkey attributes are mutually independent	3NF	1NF	2NF	5NF
21	_____ attribute does not participate in the primary key of the relation concerned	key attribute	Non-key attribute	Variable	none
22	A relation is in _____ if and only if, in every legal value of that relation, every tuple contains exactly one value for each attribute	2NF	3NF	1NF	4NF
23	_____ technique is used to eliminate the redundancy	Null Values	Decomposition	Normalizations	Concatenation
24	A relation 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
25	A relation 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
26	A relation is _____ if and only if the only determinants are candidate keys.	1NF	2NF	3NF	4NF

27	In the functional dependency, left-hand side indicates _____	determinants	Dependencies	trivial	non-trivial
28	In the functional dependency, right-hand side indicates _____	determinants	Dependencies	trivial	non-trivial
29	_____ property enables us to enforce any constraint on the original relation by simply enforcing same constraints on each of the _____ package constructs are declared in the package specification and defined in package body.	dependency preservation	lossless-join	normal forms	decomposition
30	_____ package constructs are not declared in the package specification but defined in package body.	public	private	internal	external
31	_____ package constructs are not declared in the package specification but defined in package body.	public	private	internal	external
32	Global _____ package item are declared in package _____ for external users to use it.	defintion	specification	body	declaration
33	Variables declared in package specification are initialised to _____ by default	zero	space	null	one
34	_____ which command is used to invoke a procedure in a package	call	execute	invoke	/
35	_____ is the process of taking a normalized database and modifying table structures to allow controlled redundancy for increased	Denormalization	normalization	functional dependency	Decomposition
36	in select statement Duplicates are eliminated by using _____ keyword.	Remove	Distinct	RM	Redundant
37	Group function is otherwise known as _____	Collection	Aggregate	function	Count
38	string Pattern matching is done through _____ operator.	Comparison	arithmetic	Logical	like
39	Which keyword is used to check if an element is in a given set?	not	in	not exist	except
40	Any two tables that are Union-Compatible if they have the same number of _____	Columns	rows	Columns and rows	null values
41	_____ Which clause is used, when we are using Group by clause, instead of where clause?	where	Having	Distinct	Group
42	_____ clause is used to group the values under particular characteristics	Order By	Group by	Count	Sum
43	which of the following is not a set operator	union	minus	intersect	plus
44	check constraint is _____ level constraint	table	column	row	database
45	a view is updatable when it is based upon _____ tables	two	one	three	any numberof
46	_____ clause is used in to make a sequence repeat the series.	cycle	nocycle	cache	start with
47	what is the maximum value for a descending sequence	-1	0	null	10^27
48	_____ is used and maintained automatically by oracle server	view	table	index	sequence
49	The procedure has only _____ priveledge	alter	delete	execute	select
50	_____ queries as execution time, rather than at the development stage.	PL/SQL	SQL*Plus	SQL	Dynamic SQL
51	The RDBMS terminology for a row is	tuple	relation	attribute	degree
52	To change column value in a table the ..... command can be used	create	insert	alter	update
53	A set of possible data values is called	attribute	degree	tuple	domain
54	..... is critical in formulating database design.	row column order	number of tables	functional dependency	normalizing
55	A ..... represents the number of entities to which another entity can be associated	mapping cardinality	table	schema	information
56	Which two files are used during operation of the DBMS	languages and utilities	query language	Data dictionary and transaction log	and query language

57	A ..... is a set of column that identifies every row in a table	composite key	candidate key	foreign key	super key
58	that data is organized and stored in two-dimensional tables called	Fields	Records	Relations	Keys
59	defines valid values that are stored in a column or data type.	View	Rule	Index	Default
60	insert into <table_name> values <list of values>	TRUE	FALSE	Table	None



Answer
lossless-join
Boyce-Codd Normal Form
there are two or more possible composite overlapping keys and one attribute of a composite key
it has no overlapping composite keys which have related attributes
dependent attributes or overlapping composite keys are
not in separate relation when there is dependent
attributes in two possible composite keys, one of the
there are multivalued dependencies between attributes in composite key
by removing overlapping composite keys
be in 2 NF
requires one to understand dependency between attributes
there are no composite attributes in the relation
6
implies that it also qualifies to be in lower normal form
there is only one value of y
vendor (vendor no, vendor name)
order (order no, qty supplied, price/unit) vendor order (vendor
Normalization
BCNF
5NF
Non-trivial
3NF
Non-key attribute
1NF
Normalizations
2NF
3NF
2NF

determinants
Dependencies
dependency preservation
public
private
specification
null
execute
Denormalization
Distinct
Aggregate
like
in
Columns
Having
Order By
plus
column
one
cycle
-1
index
execute
Dynamic SQL
tuple
update
domain
functional dependency
mapping cardinality
Data dictionary and transaction log

super key
Relations
Index
TRUE

**UNIT-5**

**SYLLABUS**

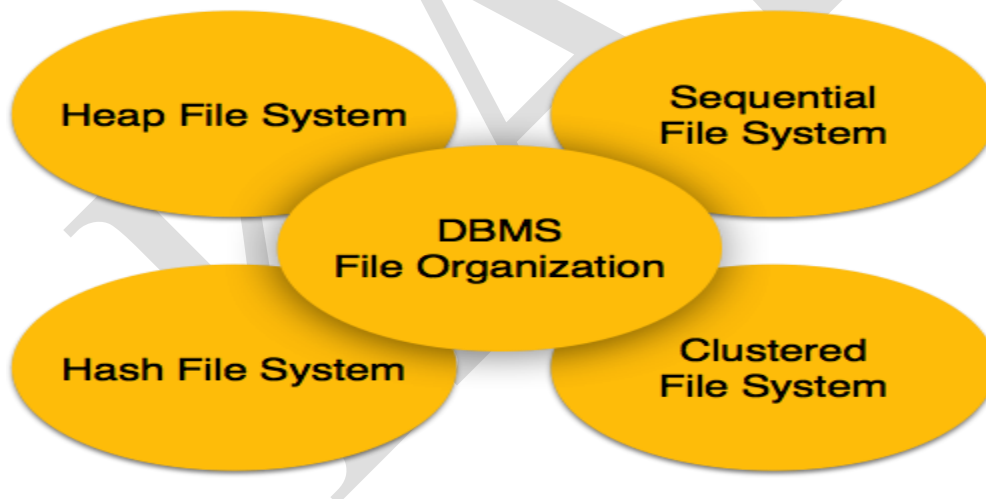
**File Structure and Indexing** (8 Lectures) Operations on files, File of Unordered and ordered records, overview of File organizations, Indexing structures for files( Primary index, secondary index, clustering index), Multilevel indexing using B and B+ trees

**File Structure**

Relative data and information is stored collectively in file formats. A file is a sequence of records stored in binary format. A disk drive is formatted into several blocks that can store records. File records are mapped onto those disk blocks.

**File Organization**

File Organization defines how file records are mapped onto disk blocks. We have four types of File Organization to organize file records –



**Heap File Organization**

When a file is created using Heap File Organization, the Operating System allocates memory area to that file without any further accounting details. File records can be placed anywhere in that memory area. It is the

responsibility of the software to manage the records. Heap File does not support any ordering, sequencing, or indexing on its own.

### **Sequential File Organization**

Every file record contains a data field (attribute) to uniquely identify that record. In sequential file organization, records are placed in the file in some sequential order based on the unique key field or search key. Practically, it is not possible to store all the records sequentially in physical form.

### **Hash File Organization**

Hash File Organization uses Hash function computation on some fields of the records. The output of the hash function determines the location of disk block where the records are to be placed.

### **Clustered File Organization**

Clustered file organization is not considered good for large databases. In this mechanism, related records from one or more relations are kept in the same disk block, that is, the ordering of records is not based on primary key or search key.

### **File Operations**

Operations on database files can be broadly classified into two categories –

- **Update Operations**
- **Retrieval Operations**

Update operations change the data values by insertion, deletion, or update. Retrieval operations, on the other hand, do not alter the data but retrieve them after optional conditional filtering. In both types of operations, selection plays a significant role. Other than creation and deletion of a file, there could be several operations, which can be done on files.

- **Open** – A file can be opened in one of the two modes, **read mode** or **write mode**. In read mode, the operating system does not allow anyone to alter data. In other words, data is read only. Files opened in read mode can be shared among several entities. Write mode allows data modification. Files opened in write mode can be read but cannot be shared.

- **Locate** – Every file has a file pointer, which tells the current position where the data is to be read or written. This pointer can be adjusted accordingly. Using find (seek) operation, it can be moved forward or backward.
- **Read** – By default, when files are opened in read mode, the file pointer points to the beginning of the file. There are options where the user can tell the operating system where to locate the file pointer at the time of opening a file. The very next data to the file pointer is read.
- **Write** – User can select to open a file in write mode, which enables them to edit its contents. It can be deletion, insertion, or modification. The file pointer can be located at the time of opening or can be dynamically changed if the operating system allows to do so.
- **Close** – This is the most important operation from the operating system's point of view. When a request to close a file is generated, the operating system
  - removes all the locks (if in shared mode),
  - saves the data (if altered) to the secondary storage media, and
  - releases all the buffers and file handlers associated with the file.

The organization of data inside a file plays a major role here. The process to locate the file pointer to a desired record inside a file varies based on whether the records are arranged sequentially or clustered.

#### Files of Unordered Records (Heap Files)

In this simplest and most basic type of organization, records are placed in the file in the order in which they are inserted, so new records are inserted at the end of the file. Such an organization is called a **heap** or **pile file**. This organization is often used with additional access paths, such as the secondary indexes discussed in Chapter 18. It is also used to collect and store data records for future use.

Inserting a new record is *very efficient*. The last disk block of the file is copied into a buffer, the new record is added, and the block is then **rewritten** back to disk. The address of the last file block is kept in the file header. However, searching for a record using any search condition involves a **linear search** through the file block by block—an expensive procedure. If only one record satisfies the search condition, then, on the average, a program will read into memory and search half the file blocks before it finds the record. For a file of  $b$  blocks, this requires searching

( $b/2$ ) blocks, on average. If no records or several records satisfy the search condition, the program must read and search all  $b$  blocks in the file.

To delete a record, a program must first find its block, copy the block into a buffer, delete the record from the buffer, and finally **rewrite the block** back to the disk. This leaves unused space in the disk block. Deleting a large number of records in this way results in wasted storage space. Another technique used for record deletion is to have an extra byte or bit, called a **deletion marker**, stored with each record. A record is deleted by setting the deletion marker to a certain value. A different value for the marker indicates a valid (not deleted) record. Search programs consider only valid records in a block when conducting their search. Both of these deletion techniques require periodic **reorganization** of the file to reclaim the unused space of deleted records. During reorganization, the file blocks are accessed consecutively, and records are packed by removing deleted records. After such a reorganization, the blocks are filled to capacity once more. Another possibility is to use the space of deleted records when inserting new records, although this requires extra bookkeeping to keep track of empty locations.

We can use either spanned or unspanned organization for an unordered file, and it may be used with either fixed-length or variable-length records. Modifying a variable-length record may require deleting the old record and inserting a modified record because the modified record may not fit in its old space on disk.

To read all records in order of the values of some field, we create a sorted copy of the file. Sorting is an expensive operation for a large disk file, and special techniques for **external sorting** are used

For a file of unordered *fixed-length records* using *unspanned blocks* and *contiguous allocation*, it is straightforward to access any record by its **position** in the file. If the file records are numbered  $0, 1, 2, \dots, r-1$  and the records in each block are numbered  $0, 1, \dots, bfr-1$ , where  $bfr$  is the blocking factor, then the  $i$ th record of the file is located in block  $(i/bfr)$  and is the  $(i \bmod bfr)$ th record in that block. Such a file is often called a **relative** or **direct file** because records can easily be accessed directly by their relative positions. Accessing a record by its position does not help locate a record based on a search condition; however, it facilitates the construction of access paths on the file, such as the indexes.

### **Sequential File Organization**

1. A **sequential file** is designed for efficient processing of records in **sorted order** on some **search key**.
  - Records are chained together by pointers to permit fast retrieval in search key order.
  - Pointer points to next record in order.
  - Records are stored physically in search key order (or as close to this as possible).
  - This minimizes number of block accesses.
  - Figure 10.15 shows an example, with *bname* as the search key.
  
2. It is difficult to maintain physical sequential order as records are inserted and deleted.
  - Deletion can be managed with the pointer chains.
  - Insertion poses problems if no space where new record should go.
  - If space, use it, else put new record in an **overflow block**.
  - Adjust pointers accordingly.
  - Figure 10.16 shows the previous example after an insertion.
  - Problem: we now have some records out of physical sequential order.
  - If very few records in overflow blocks, this will work well.
  - If order is lost, reorganize the file.
  - Reorganizations are expensive and done when system load is low.
  
3. If insertions rarely occur, we could keep the file in physically sorted order and reorganize when insertion occurs. In this case, the pointer fields are no longer required.

### **Indexing**

We know that data is stored in the form of records. Every record has a key field, which helps it to be recognized uniquely.

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to what we see in books.

Indexing is defined based on its indexing attributes. Indexing can be of the following types –

- **Primary Index** – Primary index is defined on an ordered data file. The data file is ordered on a **key field**. The key field is generally the primary key of the relation.



- **Secondary Index** – Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
- **Clustering Index** – Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

Ordered Indexing is of two types –

- Dense Index
- Sparse Index

### Dense Index

In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index records contain search key value and a pointer to the actual record on the disk.

China	● →	China	Beijing	3,705,386
Canada	● →	Canada	Ottawa	3,855,081
Russia	● →	Russia	Moscow	6,592,735
USA	● →	USA	Washington	3,718,691

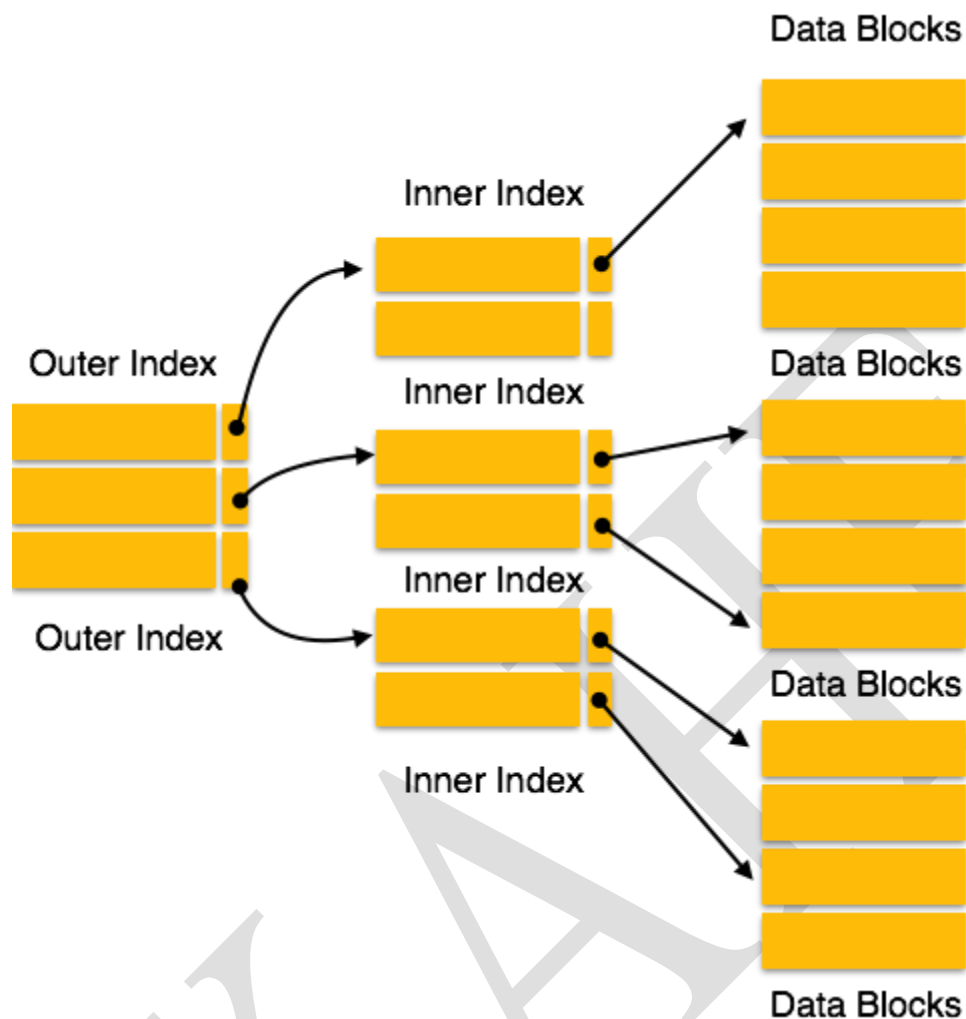
### Sparse Index

In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk. To search a record, we first proceed by index record and reach at the actual location of the data. If the data we are looking for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.

China	●	→	China	Beijing	3,705,386
Russia	●	↘	Canada	Ottawa	3,855,081
USA	●	↘	Russia	Moscow	6,592,735
		↘	USA	Washington	3,718,691

### Multilevel Index

Index records comprise search-key values and data pointers. Multilevel index is stored on the disk along with the actual database files. As the size of the database grows, so does the size of the indices. There is an immense need to keep the index records in the main memory so as to speed up the search operations. If single-level index is used, then a large size index cannot be kept in memory which leads to multiple disk accesses.



Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.

### What are B-trees?

- B-trees are balanced search trees: height =  $O(\log(n))$  for the worst case.
- They were designed to work well on Direct Access secondary storage devices (magnetic disks).
- Similar to red-black trees, but show better performance on disk I/O operations.
- B-trees (and variants like B+ and B\* trees ) are widely used in database systems.

Data structures on secondary storage:

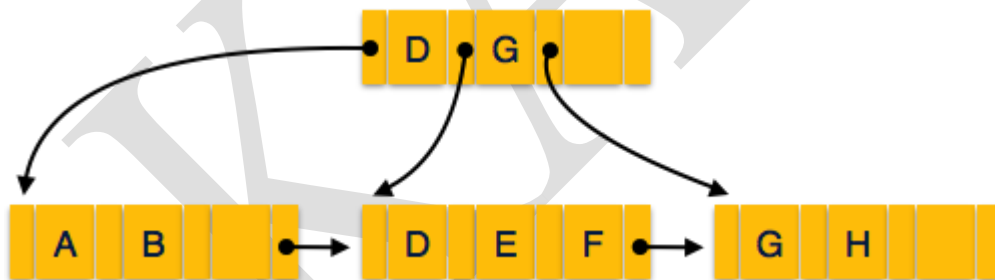
- Memory capacity in a computer system consists broadly on 2 parts:
  1. Primary memory: uses memory chips.
  2. Secondary storage: based on magnetic disks.
- Magnetic disks are cheaper and have higher capacity.
- But they are much slower because they have moving parts. B-trees try to read as much information as possible in every disk access operation.

### **B<sup>+</sup> Tree**

A B<sup>+</sup> tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B<sup>+</sup> tree denote actual data pointers. B<sup>+</sup> tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B<sup>+</sup> tree can support random access as well as sequential access.

### **Structure of B<sup>+</sup> Tree**

Every leaf node is at equal distance from the root node. A B<sup>+</sup> tree is of the order **n** where **n** is fixed for every B<sup>+</sup> tree.



### **Internal nodes –**

- Internal (non-leaf) nodes contain at least  $\lceil n/2 \rceil$  pointers, except the root node.
- At most, an internal node can contain **n** pointers.

### **Leaf nodes –**

- Leaf nodes contain at least  $\lceil n/2 \rceil$  record pointers and  $\lceil n/2 \rceil$  key values.

- At most, a leaf node can contain **n** record pointers and **n** key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

### **B<sup>+</sup> Tree Insertion**

- B<sup>+</sup> trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows –
  - Split node into two parts.
  - Partition at  $i = \lfloor (m+1)/2 \rfloor$ .
  - First **i** entries are stored in one node.
  - Rest of the entries (i+1 onwards) are moved to a new node.
  - **i<sup>th</sup>** key is duplicated at the parent of the leaf.
- If a non-leaf node overflows –
  - Split node into two parts.
  - Partition the node at  $i = \lfloor (m+1)/2 \rfloor$ .
  - Entries up to **i** are kept in one node.
  - Rest of the entries are moved to a new node.

### **B<sup>+</sup> Tree Deletion**

- B<sup>+</sup> tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
  - If it is an internal node, delete and replace with the entry from the left position.
- After deletion, underflow is tested,
  - If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
  - Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then

- Merge the node with left and right to it.

## **Possible Questions**

### **Part – B (2 Mark)**

1. Define Clustered Indexing with example.
2. What is Primary Index. Give Example?
3. What is Secondary Index?
4. What is multilevel indexing?
5. What is Ordered Files?

### **Part – C (6 Mark)**

1. Discuss Indexing structures with neat sketch.
2. Describe how they are organized inside a file?
3. Explain overview of File organizations
4. What is clustering index? Give Example.
5. Explain with example i) Primary index                      ii) clustering index.
6. Discuss File Structure in DBMS with example.
7. What is an index? Explain its role in improving database access
8. Explain in detail about B+ trees.

9. Mention various types of records. Describe how they are organized inside a file?
10. Discuss in detail about cluster and Multilevel indexes.

KAHE



**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
**DEPARTMENT OF COMPUTER SCIENCE, CA & IT**  
**II BSc CS- Batch 2017-2020**  
**PART-A OBJECTIVE TYPE/ One mark Online Examination**  
**DATABASE MANAGEMENT SYSTEM(17CSU403)**

**UNIT-V**

S.No	Questions	opt1	opt2	opt3	opt4
1	Index which has an entry for some of key value is classified as	linear index	dense index	non dense index	cluster index
2	Primary indexes, secondary indexes and cluster indexes are all types of	ordered indexes	unordered indexes	linear indexes	relative search indexes
3	In multilevel indexes, primary index created for its first level is classified as	zero level of multilevel index	third level of multilevel index	second level of multilevel index	first level of multilevel index
4	Indexes which specifies address of records on disk with a physical pointer are classified as	structural index	hashing index	physical index	logical index
5	Example of non dense index is	ternary index	secondary index	primary index	clustering index
6	The method of access which uses key transformation is known as	direct	hash	random	sequential
7	The physical location of a record is determined by a mathematical formula that transforms	B-Tree File	Hashed File	Indexed File	Sequential file
8	What is the purpose of index in sql server	query performance	To provide an index to a record	To perform fast searches	All of the mentioned
9	How many types of indexes are there in sql server?	1	2	3	4
10	How non clustered index point to the data?	It never points to anything	It points to a data row	pointing data rows containing	None of the mentioned
11	Which one is true about clustered index?	is not associated with table	is built by default on unique key	not built on unique key	None of the mentioned
12	What is true about indexes?	the performance even if the table	for sql server engines to work	harder for sql server engines to	None of the mentioned
13	Does index take space in the disk ?	as and when required	Yes, Indexes are stored on disk	Indexes are never stored on disk	Indexes take no space
14	What are composite indexes ?	are composed by database for its	index is a combination of	can never be created	None of the mentioned
15	columns for a record together, each column is stored separately with all other rows in an index.	Clustered	Non clustered	Column store	Row store
16	all the columns requested in the query without performing further lookup into the clustered index.	Clustered	Non Clustered	Covering	B-Tree
17	A(n) _____ can be used to preserve the integrity of a document or a message.	Message digest	Message summary	Encrypted message	None of the mentioned
18	A hash function must meet _____ criteria.	Two	Three	Four	Five
19	Index which has an entry for some of key value is classified as	linear index	dense index	non dense index	cluster index
20	In data file, first record of any of block is called	anchor record	dense record	non dense record	none of above
21	File which has secondary index for its every field is classified as	fully inverted file	fully indexed file	secondary indexed file	primary indexed file
22	First field in primary index having same data type as in ordering field is considered as	indexed key	ternary key	secondary key	primary key
23	In multilevel indexes, primary index created for its second level is classified as	second level of multilevel index	first level of multilevel index	zero level of multilevel index	third level of multilevel index
24	The SQL database language includes statements for:	Database definition.	Database manipulation.	Database control.	All of the above.
25	A command to remove a relation from an SQL database	Delete table table name	Drop table table name	Erase table table name	Alter table table name
26	Which SQL Query is use to remove a table and all its data from the database?	Create Table	Alter Table	Drop Table	None of these
27	A type of query that is placed within a WHERE or HAVING clause of another query is called	Super query	Sub query	Master query	Multi-query
28	Aggregate functions are functions that take a _____ as input and return a single value.	Collection of values	Single value	Aggregate value	Both a & b



29	Which of the following should be used to find the mean of the salary ?	Mean(salary)	Avg(salary)	Sum(salary)	Count(salary)
30	All aggregate functions except _____ ignore null values in their input collection.	Count(attribute)	Count(*)	Avg	Sum
31	A Boolean data type that can take values true, false, and _____.	1	0	Null	Unknown
32	If we do want to eliminate duplicates, we use the keyword in the aggregate expression.	Distinct	Count	Avg	Primary key
33	The _____ connective tests for set membership, where the set is a collection of values produced by a select clause.	Or	Not in	In	and
34	The _____ connective tests for the absence of set membership.	Or	Not in	In	and
35	We can test for the nonexistence of tuples in a subquery by using the _____ construct.	Not exist	Not exists	Exists	Exist
36	Dates must be specified in the format	mm/dd/yy	yyyy/mm/dd	dd/mm/yy	yy/dd/mm
37	Which of the following is used to store movie and image files ?	Clob	Blob	Dlob	None of the above
38	Entities are identified from the word statement of a problem by	picking words which are adjectives	picking words which are nouns	picking words which are verbs	picking words which are pronouns
39	Relationships are identified from the word statement of a problem by	picking words which are adjectives	picking words which are nouns	picking words which are verbs	picking words which are pronouns
40	One entity may be	related to only one other entity	related to itself	related to only two other entities	related to many other entities
41	By relation cardinality we mean	number of items in a relationship	number of relationships in which an entity	number of items in an entity	number of related occurrences for each of the two
42	If an entity appears in only one relationship then it is	a 1:1 relationship	a 1:N relationship	a N:1 relationship	a N:M relationship
43	If an entity appears in N relationships then it is	a 1:1 relationship	a 1:N relationship	a N:1 relationship	a N:M relationship
44	If an entity appears in not more than 5 relationships then it is a	1:1 relationship	1:5 relationship	5:1 relationship	5:5 relationship
45	A relation is	an entity	a relationship	members of a relationship set	members of an entity set or a relationship set
46	Rows of a relation are called	tuples	a relation row	a data structure	an entity
47	The database schema is written in	HLL	DML	DDL	DCL
48	The way a particular application views the data from the database that the application uses is a	module	relational model	schema	sub schema
49	The relational model feature is that there	is no need for primary key data	data independence	relationships among records.	are tables with many dimensions
50	Which of the following are the properties of entities?	Groups	Table	Attributes	Switchboards
51	Which database level is closest to the users?	External	Internal	Physical	Conceptual
52	Which are the two ways in which entities can participate in a relationship?	Passive and active	Total and partial	Simple and Complex	All of the above
53	..... data type can store unstructured data	RAW	CHAR	NUMERIC	VARCHAR
54	insert into <table_name> (column list) values <list of values>	TRUE	FALSE	Table	None
55	..... first proposed the process of normalization in DBMS.	Edgar. W	Edgar F. Codd	Edward Stephen	Edward Codd
56	Which of the following is not comparison operator?	<>	<	=<	>=
57	An outstanding functionality of SQL is its support for automatic ..... to the target data.	programming	functioning	navigation	notification
58	.....specifies a search condition for a group or an aggregate	GROUP BY Clause	HAVING Clause	FROM Clause	WHERE Clause
59	Drop Table cannot be used to drop a table referenced by a ..... constraint.	Local Key	Primary Key	Composite Key	Foreign Key

60	The user defined data type can be created using	Create datatype	Create data	Create definetype	Create type
----	---	-----------------	-------------	-------------------	-------------

Answer
non dense index
ordered indexes
second level of multilevel index
physical index
clustering index
hash
Hashed File
All of the mentioned
2
pointing data rows containing
is built by default on
for sql server engines to work
Yes, Indexes are stored on disk
index is a combination of
Column store
Covering
Encrypted message
Three
non dense index
anchor record
fully inverted file
primary key
third level of multilevel index
All of the above.
Drop table table name
Drop Table
Sub query
Collection of values

Avg(salary)
Count(*)
Unknown
Distinct
In
Not in
Not exists
yyyy/mm/dd
Blob
picking words which are nouns
picking words which are pronouns
related to many other entities
number of related occurrences for
a 1:1 relationship
a N:M relationship
1:5 relationship
an entity
tuples
DDL
sub schema
data independence
Attributes
External
Total and partial
RAW
TRUE
Edgar F. Codd
=<
notification
HAVING Clause
Foreign Key

Create type