2019
MU

Strictly as per the New Revised Syllabus (Rev - 2016) of
Mumbai University w.e.f. academic year 2018-2019
(As per Choice Based Credit and Grading System)

Database Management System

Semester V - Computer Engineering (Code : CSC502) / Electronics Engineering (Department Level Elective-I) (Code : ELXDLO5011)

Same Subject, Same Author with New Publication

Mahesh Mali

With Solved Latest University Question Paper of Dec. 2018.

TechKnowledge™ Publications

Database Management System

Semester V - Computer Engineering (CSC502),
Electronics Engineering
(Department Level Elective-I) (Code - ELXDLO5011)
(Mumbai University)

Strictly as per the New Revised Syllabus (Rev- 2016) of Mumbai University w.e.f. academic year 2018-2019

Prof. Mahesh Mali
Ph.D. (Computer Engineering) (Pursuing),
M.E. (Computer Engineering), B.E. (Information Technology),
Oracle Certified PL/SQL Developer Associate (OCA),
SAS Certified Data Analyst.
Mumbai, Maharashtra, India.

TechKnowledge™ Publications

MO44A Price ₹ 225/-

(Book Code : MO44A)

Database Management System

Prof. Mahesh Mali

(Semester V - Computer Engineering, Electronics Engineering (Department Level Elective-I) (MU))

First Printed in India	:	April 2010 **(For Mumbai University)**
First Edition	:	August 2018
Second Revised Edition	:	July 2019 **(TechKnowledge Publications)**

This edition is for sale in India, Bangladesh, Bhutan, Maldives, Nepal, Pakistan, Sri Lanka and designated countries in South-East Asia. Sale and purchase of this book outside of these countries is unauthorized by the publisher.

Printed at : 37/2, Ashtvinayak Industrial Estate, Near Pari Company,
Narhe, Pune, Maharashtra State India,
Pune - 411041.

ISBN : 978-93-89424-23-2

Published by

TechKnowledge Publications

Head Office : B/5, First floor, Maniratna Complex,
Taware Colony, Aranyeshwar Corner,
Pune - 411 009. Maharashtra State, India
Ph : 91-20-24221234, 91-20-24225678.

[CSC502] (FID : MO44) (Book Code : MO44A)

(Book Code : MO44A)

Preface

Dear Students,

I am extremely happy to present this book to you. I have divided the subject into small chapters so that the topics can be arranged and understood properly. The topics within the chapters have been arranged in a proper sequence to ensure smooth flow and understanding of the subject.

I present this book in the loving memory of **Late Shri. Pradeepji Lunawat,** our source of inspiration and a strong foundation of **"TechKnowledge Publications".** He will always be remembered in our hearts and motivate us to achieve our new milestone.

I am thankful to Prof. Arunoday Kumar, Shri. Shital Bhandari & Shri. Chandroday Kumar for the encouragement and support that they have extended. I am also thankful to the staff members of TechKnowledge Publications and others for their efforts to make this book as good as it is. I have made every possible efforts to eliminate all the errors in this book. However if you find any, please let me know, because that will help me to improve the book quality further.

I am also thankful to my family members and friends for their patience and encouragement.

- Author

□□□

(Book Code : MO44A)

Syllabus

Database Management System

(Mumbai University)

Course Code	Course Name	Credits
CSC502	Database Management System	4

Course Objectives :

1. Learn and practice data modelling using the entity-relationship and developing database designs
2. Understand the use of Structured Query Language (SQL) and learn SQL syntax.
3. Apply normalization techniques to normalize the database.
4. Understand the needs of database processing and learn techniques for controlling the consequences of concurrent data access.

Course Outcomes : On successful completion of course learner will be able to:

1. Understand the fundamentals of a database systems.
2. Design and draw ER and EER diagram for the real life problem.
3. Convert conceptual model to relational model and formulate relational algebra queries.
4. Design and querying database using SQL.
5. Analyze and apply concepts of normalization to relational database design.
6. Understand the concept of transaction, concurrency and recovery.

(Book Code : MO44A)

Prerequisite : Basic knowledge of Data structure.

Module No.	Unit No.	Topics	Hrs.
1.0		**Introduction Database Concepts**	
	1.1	• Introduction, Characteristics of databases • File system v/s Database system • Users of Database system	4
	1.2	• Data Independence • DBMS system architecture. • Database Administrator **(Refer chapters 1 and 2)**	
2.0		**Entity–Relationship Data Model**	
	2.1	• The Entity-Relationship (ER) Model : Entity types : Weak and strong entity sets, Entity sets, Types of Attributes, Keys, Relationship constraints : Cardinality and Participation, Extended Entity-Relationship (EER) Model : Generalization, Specialization and Aggregation. **(Refer chapter 3)**	8
3.0		**Relational Model and relational Algebra**	
	3.1	• Introduction to the Relational Model, relational schema and concept of keys. • Mapping the ER and EER Model to the Relational Model	8
	3.2	• Relational Algebra – unary and set operations, Relational Algebra Queries. **(Refer chapters 4 and 5)**	
4.0		**Structured Query Language (SQL)**	
	4.1	• Overview of SQL • Data Definition Commands, Data Manipulation commands, Data Control commands, Transaction Control Commands.	12
	4.2	• Set and string operations, aggregate function - group by, having. • Views in SQL, joins, Nested and complex queries, Integrity constraints :- key constraints, Domain Constraints, Referential integrity, check constraints.	
	4.3	• Triggers **(Refer chapters 6, 7 and 8)**	

(Book Code : MO44A)

Module No.	Unit No.	Topics	Hrs.
5.0		**Relational - Database Design**	
	5.1	• Pitfalls in Relational-Database designs, Concept of normalization • Function Dependencies, First Normal Form, 2nd, 3rd, BCNF, multi valued dependencies, 4NF. **(Refer chapter 9)**	8
6.0		**Transactions Management and Concurrency**	
	6.1	• Transaction concept, Transaction states, ACID properties • Concurrent Executions, Serializability - Conflict and View, Concurrency Control : Lock-based, Timestamp-based protocols.	12
	6.2	• Recovery System : Failure Classification, Log based recovery, ARIES, Checkpoint, Shadow paging. • Deadlock handling **(Refer chapters 10, 11 and 12)**	

❑❑❑

MODULE NO. I

Syllabus : Introduction, Characteristics of databases, File system V/s Database system, Users of a Database system. Data Independence, DBMS system architecture, Database Administrator.

MODULE NO. II

Syllabus : The Entity-Relationship (ER) Model : Entity types - Weak and strong entity sets, Entity sets, Types of Attributes, Keys, Relationship constraints : Cardinality and Participation, Extended Entity-Relationship (EER) Model : Generalization, Specialization and Aggregation.

MODULE NO. III

Syllabus : Introduction to the Relational Model, relational schema and concept of keys, Mapping the ER and EER Model to the Relational Model. Relational Algebra – unary and set operations, Joins, Relational Algebra Queries.

MODULE NO. IV

Syllabus : Overview of SQL, Data Definition Commands, Data Manipulation commands, Data Control commands, Transaction Control commands. Integrity constraints : key constraints, Domain Constraints, Referential integrity, check constraints. Aggregate function-group by,having, Views in SQL, Nested and complex queries. Triggers.

MODULE NO. V

Syllabus : Pitfalls in Relational-Database designs, Concept of normalization, Functional Dependencies, First Normal Form, 2nd, 3rd, BCNF, multi valued dependencies, 4NF..

MODULE NO. VI

Syllabus : Transaction concept, Transaction states, ACID properties , Concurrent Executions, Serializability – Conflict and View. Concurrency Control : Lock-based, Timestamp-based protocols. Failure Classification, Log based recovery, ARIES, Checkpoint, Shadow paging, Deadlock handling.

□□□

Introduction Database Concepts

Syllabus

Introduction, Characteristics of databases, File system V/s Database system, Users of a Database system.

1.1 Introduction to DBMS

Q.	Explain the detailed concept of DBMS.	(5 Marks)
Q.	Write a short note on: Data.	(2 Marks)
Q.	Write a short note on: Database.	(2 Marks)
Q.	Discuss Database system.	(2 Marks)

- Many of us are very much familiar with the term called as data.
- We come across term "data" regularly in our day to day life.
- The name of a person, the price of a book, number of students in a college, pin code of a city, etc. are some examples of data.
- In our daily life, we may need to remember the bulk amount of data, which is quiet difficult for us due to memory constraints.

Example :

- We may be in a position to tell accurately the age, height, income, educational qualification, residential address, etc. of our close friends.
- But it could be very difficult for us to memorize all these information for a large number of individuals in an organisation.

1. Data

- The facts and figures that can be recorded in system and that have special meaning assigned to it is called as data.
- The system use to record these data can be a manual system (register) or it can be a computerized system.

Example :

- Data of a customer like name, telephone number, address and product purchased date etc.
- As need of data increases, there is need to develop a computer-based system for storing and managing data as a file system or information system.

2. Database

- A database is a collection of data items stored in one place and having some common base (Background) between them.
- For Example : A college database contains data such as teachers, students, books, canteen etc. college is common (Base) between all above data items.
- So, Data with a common base (Background) is called as Database .
- The database acts as a logical collection of relevant data. It is designed to offer an organized mechanism for storing, managing and retrieving stored information.

Student table

Sid	Name	Class	Major

Course Table

Cid	Name	Hours

Department Table

Did	Name

Marks Table

Sid	Cid	Marks	Grade

Fig. 1.1.1 : Sample Student Database

3. Database Management System (DBMS)

- A **Database Management System (DBMS)** is a collection of software or programs which help user in creation and maintenance of a database (set of information). Hence it is also known as a computerized record-keeping system.
- DBMS is the software system that helps in the process of defining, constructing, manipulating the database.
- Database management system has become an integral part of the information systems of many organizations as it is used to handle a huge amount of data.
- Computer-based **Information Systems (IS)** is capable of serving many complex tasks in

a coordinated manner. Such systems handle large volumes of data, multiple users and several applications in a centralized database environment.

- The heart of an Information System (IS) is database management system. This is because most **Information Systems (IS)** have to handle huge amounts of data. This core module of an Information System is also called as Database Management System (DBMS).

Examples :

- MS Access, Fox Pro by Microsoft.
- Oracle by Oracle corp.
- SQL Server By Microsoft.
- Ingres, DB2 by IBM.

1.2 Characteristics of DBMS

Q.	Explain the features of DBMS.	(6 Marks)
Q.	Explain various advantages of Databases.	(6 Marks)

- The database approach has many important characteristics due to which database has become an integral part of the software industry.

The various characteristics of the DBMS are as mentioned below

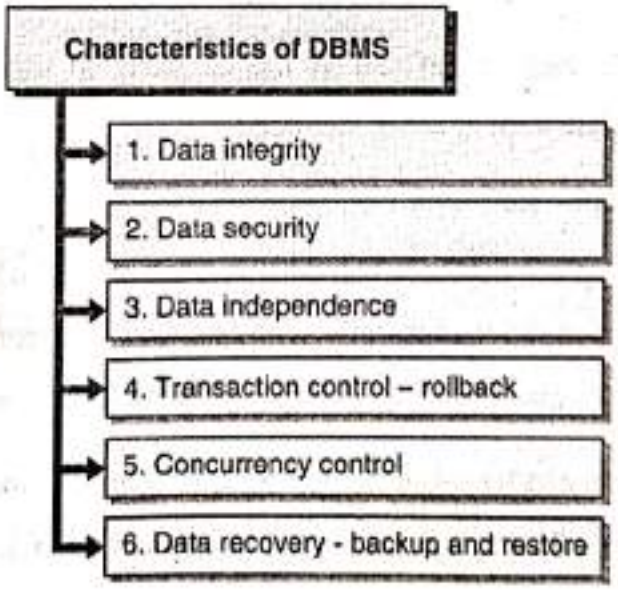


Fig. 1.2.1 : Characteristics of DBMS

1. Data integrity

- Integrity constraints provide a way of ensuring that changes made in the database by

authorized users that do not result in the loss of data consistency and correctness.

- Database integrity is concerned with the correctness and completeness of data in the database.
- This objective can never be guaranteed, one cannot ensure that every entry made in database is accurate.
- Some examples of incorrect data are as below :
  1. Student taking admission to branch which is not available in college.
  2. Employee assigned with non existing department.
  3. Sometime inconsistency introduced due to system failures.

2. Data security

- A DBMS system always has a separate system for security which is responsible for protecting database against accidental or intentional loss, destruction or misuse.
- Data in database should be given to only authorized users.
- Only authorized users should be allowed to modify data.
- Authorized users are able to access data any time he wants.

3. Data independence

Data Independence can be defined as the capacity to change data kept at one place without changing data kept at other locations.

4. Transaction control - Rollback

- The changes made in the database can be reverted back with help of rollback command.
- The changes can be saved successfully with the help of commit data command.

5. Concurrency control

- The data in database can be accessed by multiple users at same point of time.
- Such operations are allowed by sharing same data between multiple users.

6. Data recovery - Backup and Restore

- Database recovery is the process of restoring the database to original (correct) state after database failure.

- The main element of database recovery is the most recent database backup.
- If you maintain database backup efficiently, then database recovery is very straight forward process.

1.3 File System v/s Database System

Q.	Discuss the advantages of Database system over Files system.	(5 Marks)
Q.	Explain advantages of DBMS over file system.	(10 Marks)
Q.	List four significant differences between file processing system and database management system.	MU - Dec. 18, 10 Marks

1. Redundancy can be reduced

- As we are using relational approach for data organization, data is not stored in more than one location.
- Repetition of information can be avoided which in turn saves storage space.

2. Inconsistency can be avoided

The database assures that all the users access actual or true data present in the database.

3. Data can be shared

- Multiple users can login at a time into the database to access information.
- They can manipulate the database in a controlled environment.

Example :

In yahoo portal, many users are accessing data in database in a controlled manner.

4. Centralized control of data

With a **centralized control of data**, the database system may be designed for an overall optimal performance for entire organization.

5. Standards can be enforced

- Standards (rules and regulations for coding and designing) can be enforced on the database to regulate the access to the database.
- Primary Key constraint or foreign key constraint can be enforced on database which will be helpful for accessing data from database.

6. **Security restrictions can be applied**
- Security is the process of limiting access of database users to the database server itself.
- Data access is most important aspect for security and needs to be carefully planned.

7. **Integrity can be maintained**

Through integrity, one can ensure only accurate data is stored within the database.

8. **Data independence can be provided**
- None of the users need to know the technical aspects of the database to access it.
- They are physically as well as logically independent to access the database.

9. **New applications** may be developed using the existing database.

Database Management System	File Processing System
Computerized record – keeping system is used in DBMS	Collection of individual files accessed by applications programs is called File Processing System
DBMS allows flexible access to data	File – Processing System is designed to allow predetermined access to data
It co-ordinates both the physical and logical	It co-ordinates only the physical access to data
DBMS provides multiple user interface	Data is isolated in the file system
Unauthorized access is restricted in DBMS	Unauthorized access cannot be restricted
Redundancy can be controlled	Redundancy cannot be controlled

1.4 Database Users

Q. Discuss different database Users. (3 Marks)

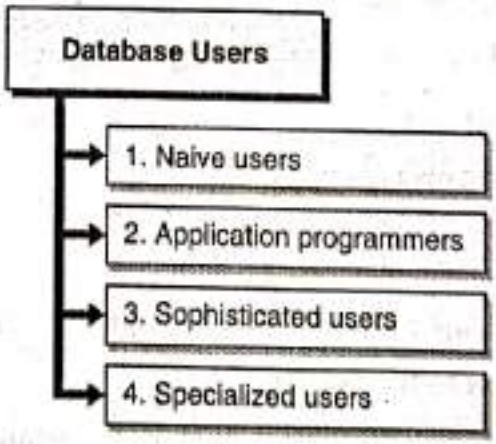


Fig. 1.4.1 : Database Users

1. Naive users

- Naive users are users who interact with the system using application programs that have been developed previously.
- For example, Student wants to pay fees Rs.50 then accountant will invoke a program called fees_payment(). This program asks the accountant for the amount of fees to be paid.
- The typical graphical user interface for naive users is a kind of form interface, where the user can fill in appropriate fields of the form.
- A given end user can access the database via one of the applications or can use an interface provided as an integral part of the database system software (such interfaces are also supported by means of applications, of course, but those applications are built-in, not user-written, e.g., query language processor)
- Naive users can read reports generated from the database.

2. Application programmers

- Application programmers responsible for writing application programs that use the database.
- Application programmers are developers or computer professionals who write application programs.
- Application programmers develop user interfaces using any preferred language.
- **Rapid Application Development (RAD)** tools are available nowadays that enable an application programmer to construct application without writing code.
- Some programming languages combine control structures with database language statements. Such languages, sometimes called fourth-generation languages.

3. Sophisticated users

- Sophisticated users interact with application without writing programs by using a database query language.
- This query will be solved by query processor.
- **Online Analytical Processing (OLAP)** tools is used to view summaries of data in different ways which helps analysts (e.g. sales of region, city etc.) with OLAP analysts can use data mining tools, which help them find certain kinds of patterns in data.

4. Specialized users

- Creates the actual database and implements technical controls needed to enforce various policy decisions.
- Specialized users are sophisticated users who develop database applications.
- The DBA is also responsible for ensuring that the system operates with adequate performance and for providing a variety of other related technical services.

Review Questions

Q. 1 Write advantages of DBMS over a file system.

Q. 2 State five main advantages of DBMS.

Q. 3 What are the different types of database system users?

Q. 4 What are the disadvantages of file processing system which were removed by DBMS?

Q. 5 Why would you choose a database system instead of simply storing the data in operating system files?

❑❑❑

CHAPTER 2

Database Architecture

Module I

Syllabus

Data Independence, DBMS system architecture, Database Administrator.

2.1 Three-Levels Schema Architecture

Q.	Explain three-level architecture of DBMS.	(5 Marks)
Q.	State and explain various levels of database abstraction.	(5 Marks)
Q.	Explain physical, conceptual and view level abstraction of DBMS.	(5 Marks)

1. Introduction

- The goal of the three-schema architecture is to separate the front end (user applications interface) and the back end (physical database).
- The three-schema architecture is a tool with which the user can visualize the schema levels in a DBMS. Many DBMS systems do not separate the three levels completely, but support the three schema architecture to some extent.
- A description of data in terms of a data model is called a schema.
- The description of a database is called **database schema**, which is specified during database design and it is not expected to change frequently.

2. Database architecture

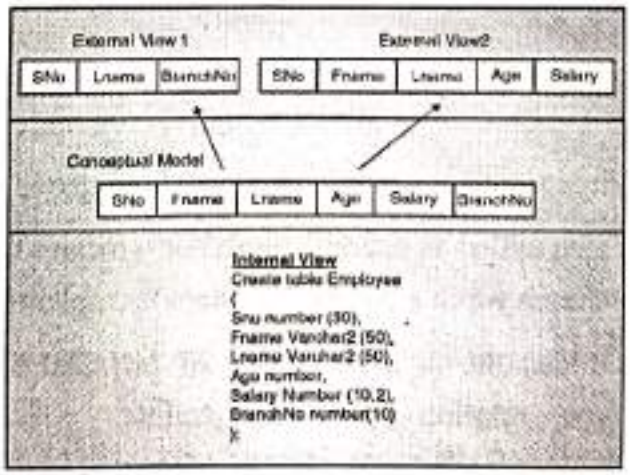


Fig. 2.1.1 : Database schema levels

(I) Internal Level (Physical Level)

- The internal level is very close to physical storage of data.
- This level describes the physical storage structure of the data in database.
- The internal (or physical) database is stored on secondary storage devices, mainly the magnetic disk.
- Describes the complete details of data storage and various available access methods for the database.
  - At its ground level, it is stored in the form of bits with the physical addresses on the secondary storage device.
  - At its highest level, it can be viewed in the form of files and simple data structures.

Internal view/ schema

- The internal view defines the various stored data types and specifies what type of indexes exist, how stored fields are represented and so on.
- The internal schema uses a physical data model.

Example :

```
Create table Employee
(   Sno         number (30),
    Fname       varchar2 (50),
    Lname       varchar2 (50),
    Age         number,
    Salary      number (10,2),
    BranchNo    number (10) );
```

(II) Conceptual level

- This level describes the structure of the whole database for a group of users.
- The conceptual model is also called as the data model or we can say data model is used to describe the conceptual schema when a database system is implemented.
- The conceptual schema hides the internal details of physical storage and targets on describing entities, data types, relationships and constraints.
- The conceptual schema contains all the information to build relevant external records. As

the conceptual model is derived from the physical model.

Conceptual view / schema

- The conceptual view is a representation of the entire content of the database.
- The conceptual view includes definitions of each of the various conceptual data types.

(III) External level (view level)

- The external level is the one closest to the user, i.e., it is related with the way data is viewed by individual end users.
- The external level includes a number of user views or external schemas.
- Each external schema 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.
- External views are the proper interface between the user and the database, as an individual user can hardly be expected to be interested in the entire database.
- The external model is derived from the conceptual model.

External view / schema

External schema consists of definitions of each of the various external data types in that external view.

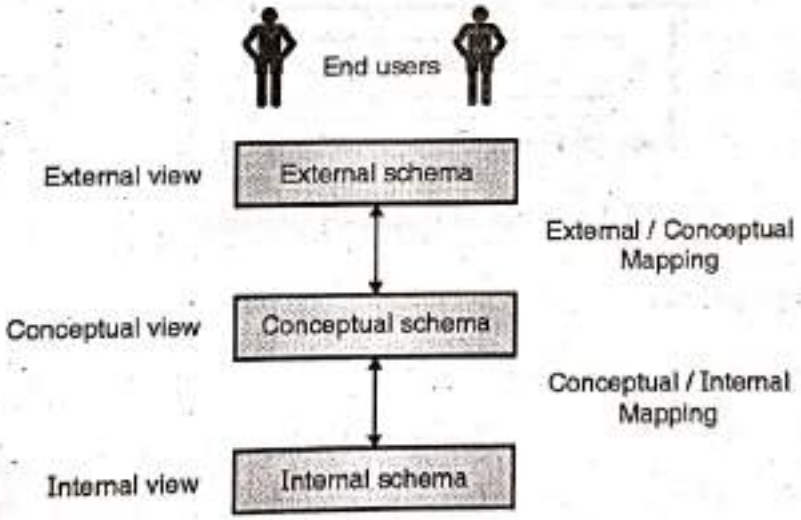


Fig. 2.1.2 : Three level schema architecture

(IV) Mapping

- The processes of transforming requests and results between various levels of architecture are called mappings.
- These mappings may be time-consuming, so small databases do not support external views.

- **External / conceptual mapping :** The DBMS must transform a request on an external schema into a request against the conceptual schema.
- **Conceptual / internal mapping :** A certain amount of mapping is necessary to transform requests between the conceptual and internal levels.

2.2 Data Independence

Q.	Define data Independence and explain types of data Independence.	(5 Marks)
Q.	Explain the term : Data Independence.	(2 Marks)
Q.	Write a short notes on : Data Independence.	(5 Marks)

Concept of data independence can be explained with help of 3 schema architecture. The three-schema architecture can make it easier to achieve true data independence.

Definition : ***Data Independence*** *can be defined as the capacity to change one level of schema without changing the schema at the next higher level.*

Types

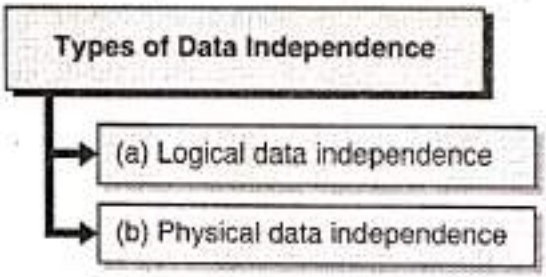


Fig. 2.2.1 : Types of Data Independence

(a) Logical data independence

- Logical data independence is a capacity to change the conceptual schema without having any changes to external schemas. (or application programs)
- Separating the external views from the conceptual view enables us to change the conceptual view without affecting the external views. This separation is sometimes called logical data independence.

Example :

- We may change the conceptual schema by removing a data item. In this case the external schemas that refer only to the remaining data should not be affected.

(b) Physical data independence

- Physical data independence is a capacity to change the internal schema without having any changes to conceptual schema.
- The separation of the conceptual view from the internal view enables us to provide a logical description of the database without the need to specify physical structures. This is often called physical data independence.

Example :

By creating additional access paths to improve the performance of retrieval. If the same data as before remains in the database, we should not have to change the conceptual schema.

2.3 Database Administrator (DBA)

Q. Write a short note on : Database Administrator. (6 Marks)
Q. Define DBA. MU - Dec. 18, 2 Marks

- The database administrator is responsible for the overall planning of the company's data resources, for the design of data, and for the day-to-day operational aspects of data management.
- A database administrator is a person responsible for the installation, configuration, upgradation, maintenance and monitoring databases in an organization.
- The overall planning of corporate data is the strategic aspect of the database administration function and involves company-wide planning of existing data and assessment of organization-wise data standards.

2.3.1 Roles of DBA

Q. Discuss the role of Database Administrator. (3 Marks)
Q. Discuss role of DBA. MU - Dec. 18, 3 Marks

- The DBA needs to performs many roles to keep the database up and running,
- System Administrator / Designer
- The database administrator need to manage DBMS software and server.
- He is also responsible for deciding on the storage and access methods.
- The DBA performs all data field updates or adding new fields into database.
- Database Developer / Programmer

- The DBA writes the programs to design database and to design the means of reorganizi databases periodically.
- The DBA also determines and implement database searching strategies.
- System Analyst
- The DBA needs to analyse the system performance and fine tune the DBMS activities.
- DBA needs to take care of system crashes by planning proper recovery procedures.
- He will also specify techniques for monitoring database performance.

2.3.2 Responsibilities of DBA

Q. Write a short note on : Responsibilities of Database Administrator. (3 Marks)

The various responsibilities of DBA are as follows,

- Designing overall Database schema
- The DBA is responsible for designing overall database schema (tables and fields). Al responsible for deciding on the data storage and access methods.
- Selecting and installing database software and hardware.
- The DBA selects the suitable DBMS software like Oracle, SQL Server or MySQL.
- Designing Authorization/Access Control
- The DBA will decide the user access levels and security checks for access and da manipulations.
- Designing Recovery Procedures
- In order to take care of system crashes DBA needs to design the system recove procedures and also specifying techniques for monitoring database performance.
- Operations Management
- The operations management of database administration deals with data problems aris on a day-to-day basis. Specifically, the responsibilities include
  - Investigation of errors found in the data.
  - Supervision of restart and recovery procedures in the event of a failure.
  - Supervision of reorganization of databases.
  - Initiation and control of all periodic dumps of data.

2.3.3 Skills Required for DBA

Q. List the important skills required to a Database Administrator (DBA). (3 Marks)

- The various programming and soft skills are required to DBA are as follows,
  - Good communication skills
  - Excellent knowledge of databases architecture and design and RDBMS
  - Knowledge of Structured Query Language (SQL).
- In addition, this aspect of database administration includes maintenance of data security, which involves maintaining security authorization tables, conducting periodic security audits, investigating all known security breaches.
- To carry out all these functions, it is crucial that the DBA has all the accurate information about the company's data readily on hand. For this purpose he maintains a *data dictionary*.
- The data dictionary contains definitions of all data items and structures, the various schemes, the relevant authorization and validation checks and the different mapping definitions.
- It should also have information about the source and destination of a data item and the flow of a data item as it is used by a system. This type of information is a great help to the DBA in maintaining centralized control of data.

2.4 Detailed DBMS Architecture

Q. Draw and explain database system structure. (6 Marks)
Q. Describe overall architecture of DBMS with diagram. (6 Marks)
Q. Explain Overall Architecture of DBMS in detail MU - Dec. 18, 10 Marks

- A database system can be separated into two different modules that deal with all operations of the overall system.

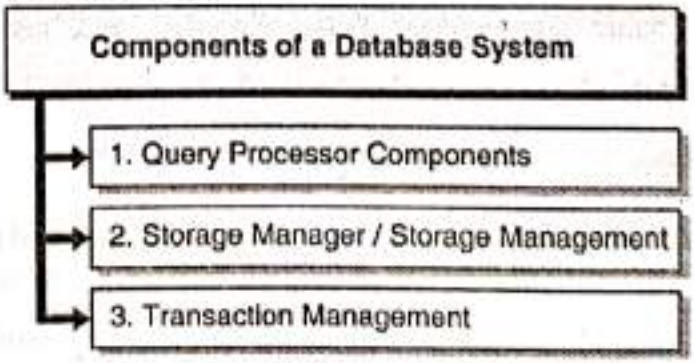


Fig. 2.4.1 : Components of a Database System

- The storage manager is important because databases typically require a huge amount of storage space.

2.4.1 Query Processor Components

Q. Write a short notes on : Query processor. (3 Marks)

1. Introduction

The query processor will accept query from user and solves it by accessing the database.

2. Parts of query processor

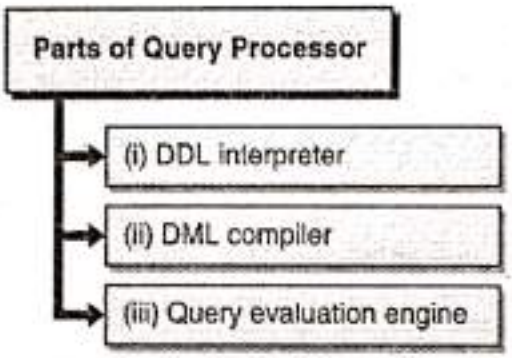


Fig. 2.4.2 : Parts of Query Processor

(i) DDL interpreter

This will interpret DDL statements and fetch the definitions in the data dictionary.

(ii) DML compiler

- This will translate DML statements in a query language into low level instructions that the query evaluation engine understands.
- A query can usually be translated into any of a number of alternative evaluation plans for same query result DML compiler will select best plan for query optimization.

(iii) Query evaluation engine

This engine will execute low-level instructions generated by the DML compiler on DBMS.

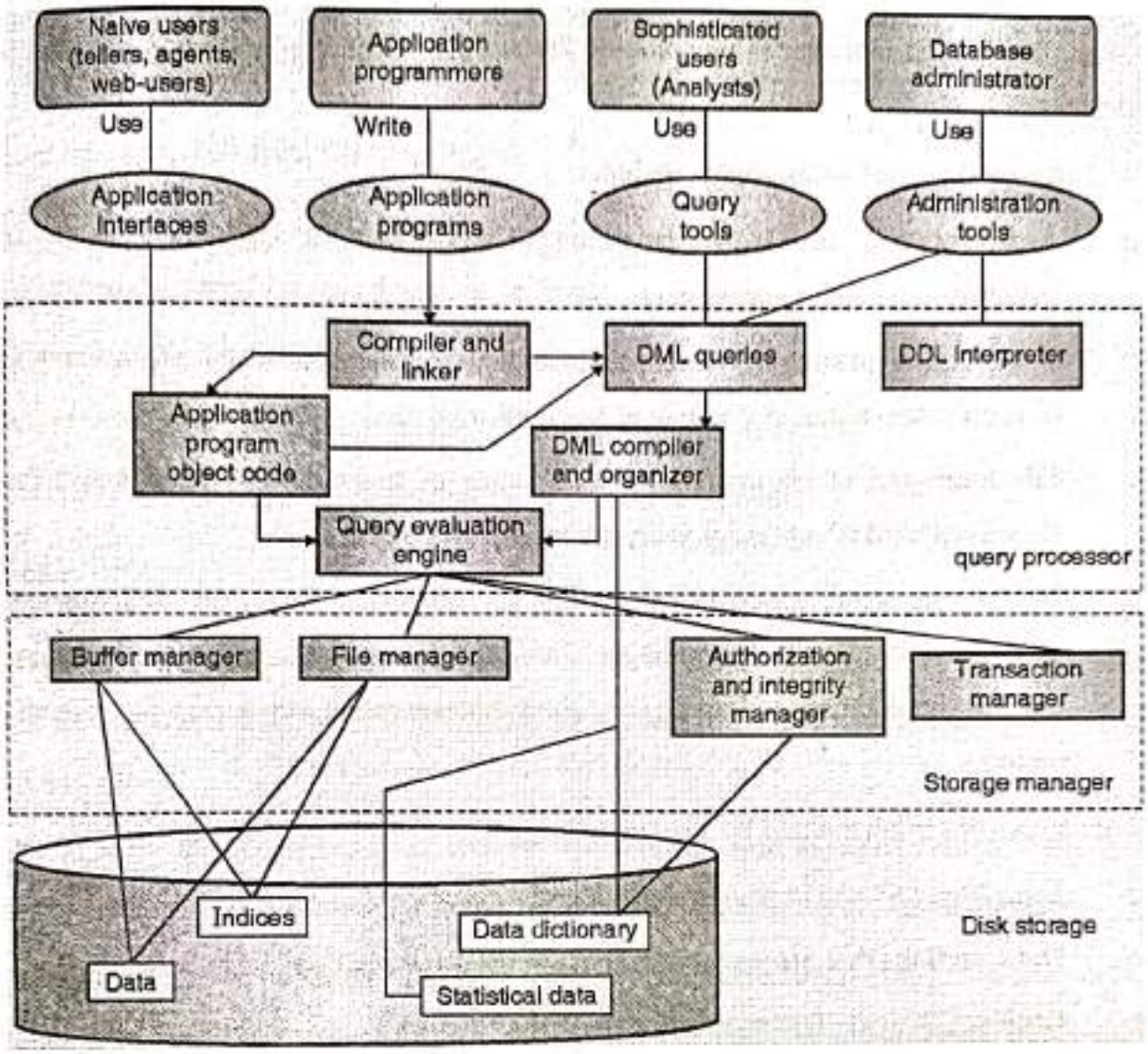


Fig. 2.4.3 : Components of DBMS

2.4.2 Storage Manager / Storage Management

Q. Write a short note on : Storage management. (3 Marks)

- **A storage manager** is a program module which acts like interface between the data stored in the database and the application programs and queries submitted to the system.
- The data is stored on the disk using the file system.
- The storage manager is programme which is responsible for the interaction with the file manager.
- The storage manager translates the various databases language statements into low level file system commands.

- Thus, the storage manager is responsible for storing, retrieving and updating data in the database.
- The storage manager components include :
  - **Authorization and integrity manager :** Checks for integrity constraints and authority of users to access data.
  - **Transaction manager,** which ensures that the database remains in a consistent (correct) state although there is system failures.
  - **File manager,** which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
  - **Buffer manager,** which is responsible for retrieving data from disk storage into main memory. The buffer manager is an important part of the database system, as it enables the database to handle data sizes that are much larger than the size of main memory.
- Data structures implemented by storage manager,
  - **Data files :** Stored in the database itself.
  - **Data dictionary :** Stores metadata about the structure of the database.
  - **Indices :** Provide fast access to data items.

2.4.3 Transaction Management

Q. Write a short note on : Transaction management. (3 Marks)

- A transaction is a series of small database operations that together form a single large operation.
- A transaction is started by issuing a BEGIN TRANSACTION command. Once this command is executed the DBMS starts monitoring the transaction.
- All operations executed after a BEGIN TRANSACTION command are treated as a single large operation.
- Application programs use transactions to execute sequences of operations when it is important that all the operations are successfully completed.
- Transaction management component will ensure the atomicity and durability properties.

2.5 Working of DBMS

- User requests data item from database.
- DBMS intercepts and interprets the request.
- Retrieves the data from the physical database.
- Constructs the record using physical/conceptual mapping.
- Records constructed using relevant conceptual/external mapping.
- Derives the required external record from conceptual record.

Example :

- Consider the situation in a library. Here, we have **data** corresponding to books, authors, suppliers, borrowers, etc. The total volume of data stored and handled in a library may be quite large.
- The Library DBMS may require several **operations**, such as issue, return or purchase of books; handle **queries** related to book information, borrowing information, etc.
- Moreover, there are different types of **users** who operate various stages or activities. For instance, a borrower may merely view certain information, whereas an issuer may be allowed to update the status of a book during issue or return.
- The Library staff may, on the other hand, add new books, their supplier, price and other information to the database.
- Each user category has a different **access right** on both, the data, as well as the processing capabilities.
- Multiple users may concurrently operate the library DBMS performing several tasks at the same time.
- They may even try to access the same data simultaneously. It is the job of a DBMS to handle the data and it's processing in an integrated, coordinated and consistent manner.

Review Questions

Q. 1 List the functions of a Database Administrator (DBA).

Q. 2 Write short note on : Responsibilities of database administrator.

Q. 3 Describe the overall architecture of DBMS with diagram.

Q. 4 Explain the following term : Data independence and its types.

Q. 5 Explain three-level architecture of DBMS.

Q. 6 Define data Independence and explain types of data Independence.

Q. 7 Draw and explain database system structure.

❑❑❑

CHAPTER 3

Entity Relationship Data Model

Module II

Syllabus

The Entity-Relationship (ER) Model : Entity types - Weak and strong entity sets, Entity sets, Types of Attributes, Keys, Relationship constraints : Cardinality and Participation, Extended Entity-Relationship (EER) Model : Generalization, Specialization and Aggregation.

3.1 Entity-Relationship (ER) Model

Q. What do you mean by ER model. **(2 Marks)**

Q. Explain Components of ER Model. **MU - Dec. 18, 5 Marks**

- In 1976, Scientist Chas hen developed the Entity-Relationship (ER) model which is a high-level conceptual data model.
- ER diagram is the first step of database design to specify the desired components of the database system and the relationships among those components.
- ER model define data elements and relationships among various data elements for a specified system.
- The ER data model is based on perception of real world data that consists of set of entities (data items) and relationship among these entities.
- ER Diagrams having components,
  - Entity
  - Attributes
  - Relationships

3.2 Entity set

Q. We can convert any entity set to a strong entity set by simply adding appropriates attributes. Why then, do we have weak entity sets? **(2 Marks)**

Q. What is strong entity? Explain with example. **(3 Marks)**

Q. What is weak entity ? Explain with example. **(3 Marks)**

(1) Introduction

- Entity is anything in real world which may have physical or logical existence.
- An **entity** is anything in real world with its physical existence. Example, Student, faculty, subject having independent physical existence.
- An entity may be an object with a physical existence or it may have logical existence. Example, Department, Section, subject may have logical existence.
- Each entity has its own properties which describes that entity such properties are known as **attributes.**

(2) Entity Set

Entity set is collection of entities of same type.

Example :

Student entity set contains all students in college database.

(3) Entity Type

- Entity set is collection of entities with same attributes.
- As in Student table, each row is an entity and have same attributes. In other words we can say a student table is an entity type
- The types of entities are,

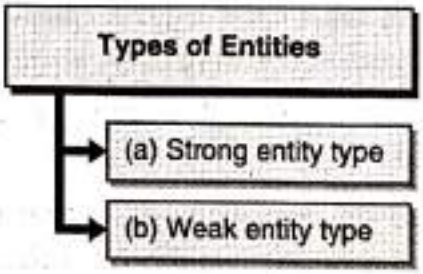


Fig. 3.2.1 : Types of entities

(a) Strong entity type

Definition : *Entity type which has its own key attributes by which we can identify specific entity uniquely is called as* ***strong entity type.***

Example :

- In case of Employee entity any specific employee can be identified by his Employee_id which is primary key of employee entity.
- In case of student in class each student identified by unique roll number which is his primary key.

- Strong entity type is represented by single rectangle.

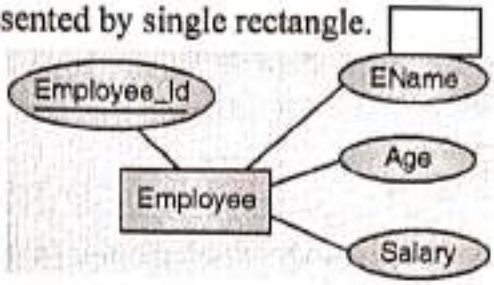


Fig. 3.2.2 : Employee entity

(b) Weak entity type

Definition : *Entity type which cannot form distinct key from their attributes and takes help from corresponding strong entity is called as* ***weak entity type.***

- These types of entities are dependent on strong entity for primary key.
- For some weak entities we assign virtual primary key. Such virtual primary key of weak entity is called as '**discriminator**'.
- Weak entity type is represented by double rectangle.

Example :

- In case of "Dependent" entity depend on employee entity for primary key.

Fig. 3.2.3 : Weak entity "dependent"

3.3 Attributes

Q.	Explain different types of attributes in ER Model.	(5 Marks)
Q.	Write a note on composite attributes.	(3 Marks)
Q.	Write a note on multivalued attributes.	(3 Marks)
Q.	Define Derived attribute.	(2 Marks)
Q.	Write a note on Null attributes	(2 Marks)
Q.	Write a note on Key attributes. Explain various type of keys in ER Diagram.	(6 Marks)

Introduction

- Each entity has its own properties which describes that entity such properties are known as attributes.
- The attribute value that describes each entity becomes a major part of data stored in database.
- Employee entity may be described by attributes name, age, phone etc.

Type	Notation
Attribute (Simple/Single valued/Stored)	

- A particular entity will have some value for each of its attributes.

Example :

- For an employee of with Employee_id 30, the name attribute value is 'Jayendra'
- The various types of attribute are used in ER diagrams,

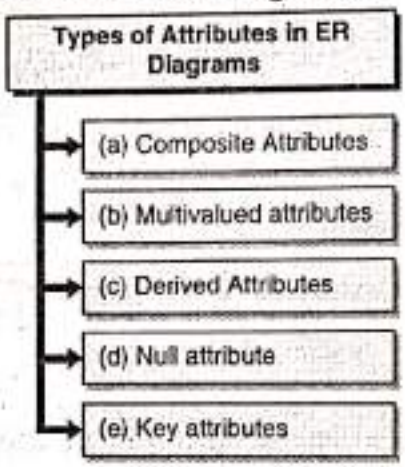


Fig. 3.3.1 : Types of attribute are used in ER diagrams

(a) Composite Attributes

- The attributes which can be divided in multiple subparts.

Type	Notation
Composite attribute	

- The divisible attributes are composite attributes.

Example :

The Name attribute of Student table can be divided into First_Name and Last_Name.

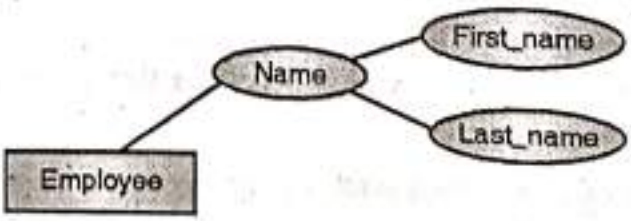


Fig. 3.3.2 : Composite attributes

(b) Multivalued Attributes

The attribute having more then one value for a same entity is called as multi-valued attribute.

Type	Notation
Multivalued Attribute	

Example :

A Single student can have multiple mobile numbers.

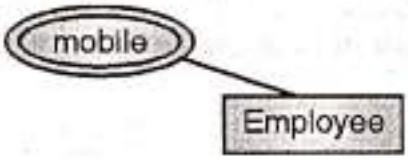
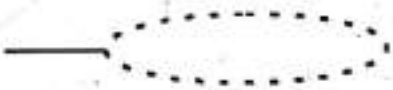


Fig. 3.3.2 Multi valued attributes

(c) Derived Attributes

Definition : *The value of some attribute can be derived from the value of related stored attribute such attributes are known as* ***derived attributes.***

Type	Notation
Derived attribute	

Example :

Employee tenure can be calculated from stored attribute 'Date_of_joining' of employee by subtracting it from today's date.

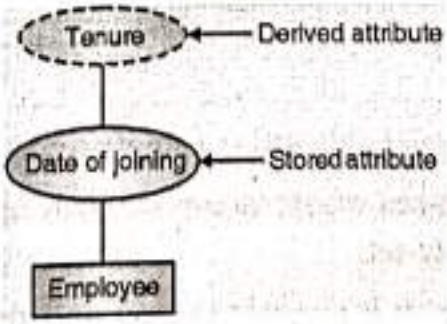


Fig. 3.3.4 : Derived attributes

(d) Null Attribute

- This attribute can take NULL value when entity does not have value for it.
- This is a special attribute the value of which is unknown, unassigned, not applicable or missing.

Example :

- The 'Net_Banking_Active_Bin' attribute gives weather particular customer having net banking facility activated or not activated.
- For bank which does not offer facility of net banking in customer table 'Net_Banking_Active_Bin' attribute is always null till Net banking facility is not activated as this attribute indicates Bank offers net banking facility or does not offers.
- These attributes can be used in future use or for unknown, unsigned, missing values of attribute.

(e) Key Attributes

This is an attribute of an entity which must have a unique value by which any row can be identified is called as key attribute of entity.

Example :

Emp_Id for employee.

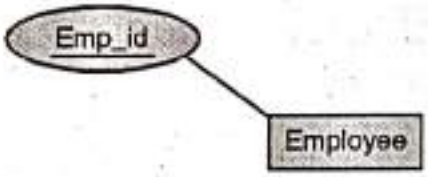


Fig. 3.3.5 : Key attributes

Type	Notation
Key attribute	

- The column value that uniquely identifies a single record in a table called as key of table.
- An attribute or set of attributes whose values uniquely identify each entity in an entity set is called a key for that entity set.
- ID is a key of student table. It is possible to have only one student with one ID (Say only one student **'Mahesh'** with ID = 1)

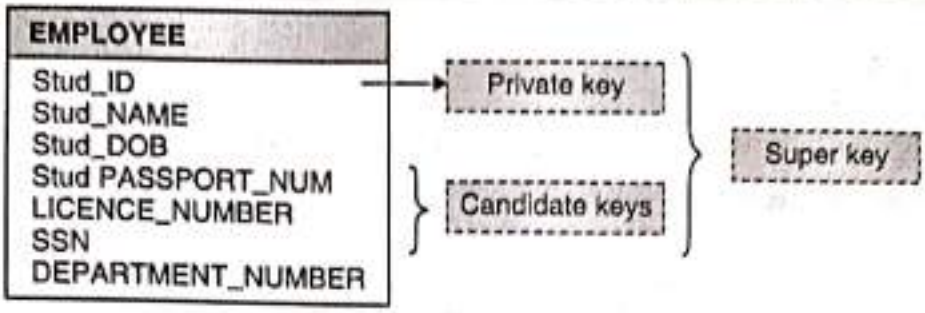


Fig. 3.3.6 : Key Concept

Types of Keys

The various types of keys in ER Diagrams are as follows,

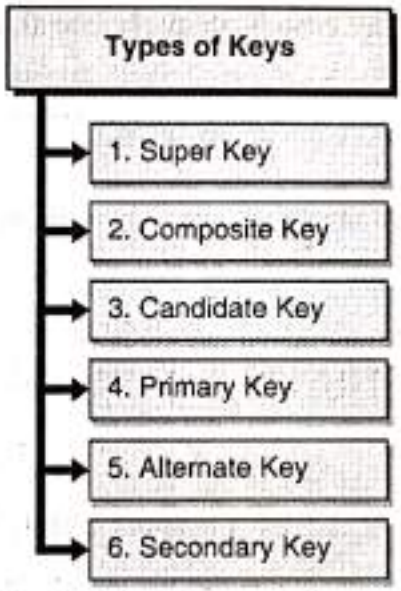


Fig. 3.3.7 : Types of keys

1. Super Key

- An attribute or set of attributes that uniquely identifies a single tuple in entity.
- There can be more than one super keys in single table

Example :

In Fig. 3.3.6 combination of (Stud_ID, Stud_name, Stud_Passport_Num Licence_ Number, ssn) acts like a super key.

2. Composite Key

Any key with more than one attributes that uniquely identifies a single tuple in entity.

Example :

In Fig. 3.3.6 a super key has more than one attribute so, it is a composite key.

3. Candidate Key

- A super key with minimum number of attributes is a candidate's key.
- No subset of candidate key van be key.

Example :

In Fig. 3.3.6 combination of (Stud_Passport_Num, Licence_Number,ssn) acts like a Candidates key.

4. Primary Key

A selected key of strong entity which uniquely identify tuple in entity is a primary key of that entity.

Example :

In Fig. 3.3.6 combination of (Stud_ID) acts like a Primary key.

5. Alternate Key

A Candidate key which is not selected as primary key.

Example :

In Fig. 3.3.6 candidate key (Stud_Passport_Num, Licence_Number, ssn) acts like alternate key

6. Secondary Key

- An attribute or set of attributes that is used to access a single tuple in entity.
- The secondary key not necessary to be unique

Example :

In Fig. 3.3.6 attribute (Stud_Passport_Num) can be used for accessing student's data, so it is acting like secondary key

Key Type	Definition
Super Key	An attribute or set of attributes that uniquely identifies a single tuple in entity.
Composite Key	Any key with more than one attributes that uniquely identifies a single tuple in entity.
Candidate Key	A super key with minimum number of attributes is a candidate's key.

Key Type	Definition
	No subset of candidate key van be key.
Primary key	A selected key of strong entity which uniquely identify tuple in entity is a primary key of that entity.
Alternate Key	A Candidate key which is not selected as primary key
Secondary Key	An attribute or set of attributes that used to access a single tuple in entity.

3.4 Relationships

Q. What is relationship set? (2 Marks)

1. Introduction

- A relationship is an association among one or more than one entities.
- We use diamond shape to show relationship.

Type	Notation
Relationship	

- It is recommended to arrange relationship to be read it from left to right or up to down.

Example :

Employee works for Department.

Fig. 3.4.1 : ER Diagram for Works_for

2. Relationship Set

Collection of all relationship of same type is relationship set. The many employees are working for different departments so it is relationship set of Works_For relationship.

3. Degree of Relationship

The degree of relationship type is number of participating entity types in a particular relation.

Types of Relationship based on degree are,

- Unary Relationship
- Binary Relationship
- Ternary Relationship

3.5 Relationship Types based on Constraints

Q. Give various constraints of relationship. **(5 Marks)**

Q. Explain the terms total participation and Partial participation with example. **(5 Marks)**

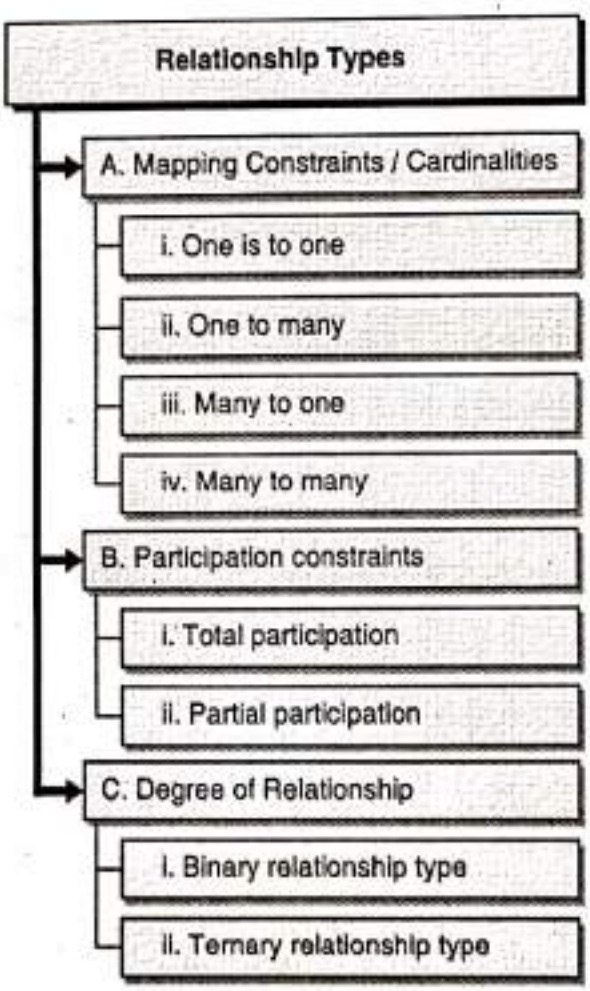


Fig. 3.5.1 : Relationship Types

(A) Mapping Constraints / Cardinalities

- Number of entities from each side participating in a relationship set.
- Cardinality expresses specific number of entity occurrence of related entity.

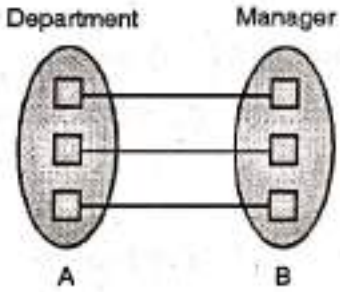
Types of mapping constraints

(i) One is to one

- In this type of constraint one tuple in entity is related with only one tuple in other entity.
- That is one row in table is related with only one row in other table.
- A associated with at most one entity in B, B associated with at most one entity in A.

Example :

- A associated with at most one entity in B, B associated with at most one entity in A.
- One department can have only one manager.
- Every row in Department table can be having relationship with only one row in Managers table.



(a) One to one mapping

(b) Representation in ER diagram

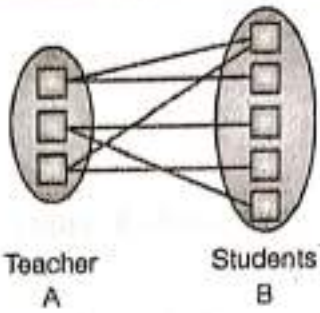
Fig. 3.5.2 : One to one mapping

(ii) One to many

- In this type of constraint one tuple in entity can be related with many tuples in other entity.
- A associated with any number of entities in B.
- B associated with at most one entity in A.

Example :

- One teacher may teach to many students.
- Every row in Teacher table can have relationship with many rows in Student table.



(a) One to many mapping

(b) Representation in ER diagram

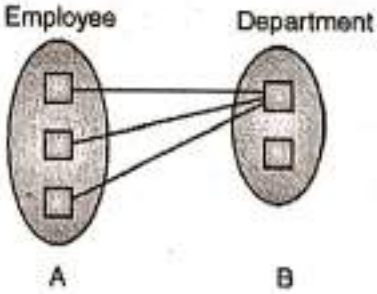
Fig. 3.5.3 : One to many mapping

(iii) Many to one

- In this type of constraint many tuple in entity can be related with only one tuple in other entity.
- A associated with at most one entity in B.
- B associated with any number of entities in entity A.

Example :

- Number of employee works for department.
- Multiple rows in Employees table can be related with only one row in Department table.



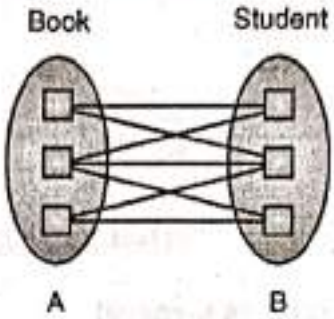
(a) Many to one mapping

(b) Representation in ER diagram

Fig. 3.5.4

(iv) Many to many

- In this type of constraint many tuple in entity can be related with multiple tuples in other entity.
- A associated with any number of entities in entity B.
- B associated with any number of entities in entity A.



(a) Many to many mapping

(b) Representation in ER diagram

Fig. 3.5.5

Example :

- Books in library issued by students.
- Multiple rows in Book table can be related with many rows in Student table.

(B) Participation constraints

(i) Total participation

- In case of total participation every object in an entity must participate in a relationship.
- The total participation is indicated by a dark line or double line between entity and relationship.

Example :

Every department must have a manager.

□□□□

Fig. 3.5.6 : Total participation

(ii) Partial participation

- In case of partial participation more than one object in an entity may participate in a relationship.
- The total participation is indicated by a single line between entity and relationship.

Example :

Employees works for department.

Fig. 3.5.7 : Partial participation

(C) Degree of Relationship (Binary Vs ternary)

- The degree of the relationship type is number of participating entity types.
- Types

(i) Binary relationship type

- A relationship of degree two.
- Example, Employees works for department.

Fig. 3.5.8 : Binary Relationship

(ii) Ternary relationship type

- A relationship of degree three.

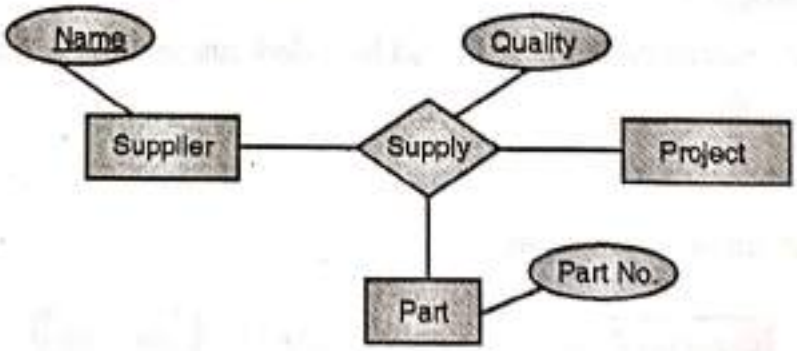


Fig. 3.5.9 : Ternary relationship

3.6 Extended Entity-Relationship (ER) Model

Q. Compare ER and EER models. (5 Marks)

- EER model includes all the modeling concept of ER model. In addition it also includes the concept of aggregation, specialization and generalization.
- A diagrammatic technique for displaying these concepts when they arise in EER schema are the resulting schema diagrams called as EER diagrams.

EER Features

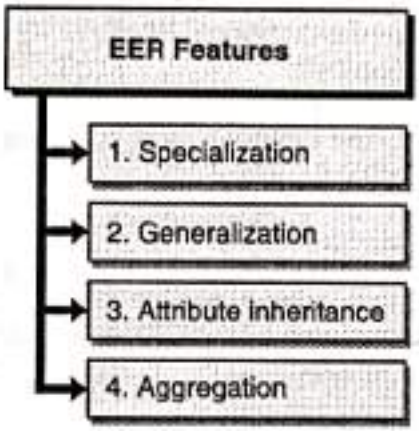


Fig. 3.6.1 : EER Features

3.6.1 Specialization

Q. Defines a set of sub class of an entity type. (5 Marks)
Q. Establish additional specific attributes with each subclass. (5 Marks)
Q. Establish additional specific relationship types between each subclass and other entity type or other subclass. (5 Marks)
Q. Write a short note on : Specialization. MU - Dec. 18, 5 Marks

- Top down approach of superclass / subclass relationship.
- Specialization is a process of defining a set of subclass of entity type, this entity type is called super **class of specialization.**
- The set of subclasses that forms a specialization is defined on the basis of some distinguishing characteristic of entity in super class.

Example :

Set of subclass (Saving_Account, Current_Account) are Specialization of super class Account.

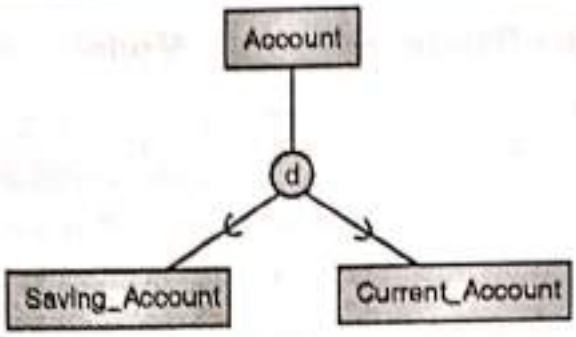


Fig. 3.6.2 : Specialization

Notation

- The subclass defined in a specialization is attached by lines to a circle which is connected to super class.
- The subset symbol on each line connecting a subclass to circle indicates the direction of super class / subclass relationship.

Specific attribute

An attribute applied only to entities of particular subclass is called as specific attribute.

3.6.2 Generalization

Q. Explain Generalization with the help of an example. **(5 Marks)**
Q. Write a short note on :Generalization. **MU - Dec. 18, 5 Marks**

This is reverse process of specialization or this is bottom up approach of Superclass/subclass relationship.

Definition

Definition : *Generalization* is a process in which we differentiate among several entity types identifying there common features and generalizing them to a single super class of which original entity type are special subclass.

Example :

Car and Bike all having several common attribute they can generalize to the super class vehicle.

Notation

- A diagrammatic notation to distinguish between generalization and specialization is used in some programming methodologies.

- Arrow pointing to generalized superclass represents generalization.
- Arrow pointing to generalized subclass represents specialization.

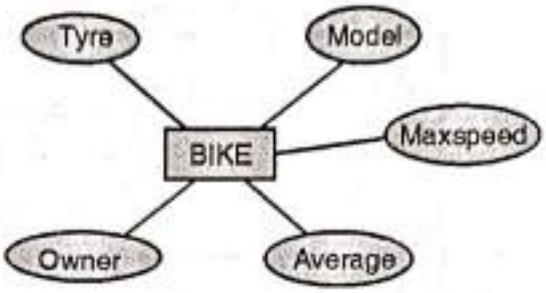


Fig. 3.6.3 : BIKE entity

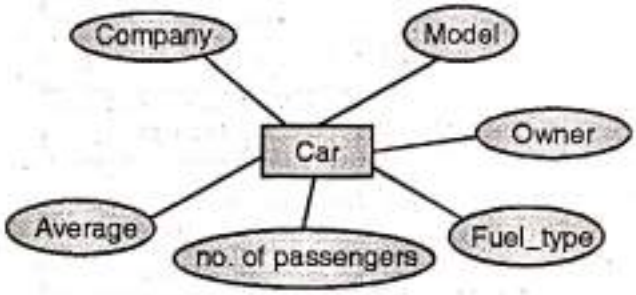


Fig. 3.6.4 : Car entity

3.6.3 Attribute inheritance

- The attributes of higher and lower level entities created by specialization and generalizations are attributes inheritance.
- Abstraction through which relationship (aggregation) is treated as higher level entities.

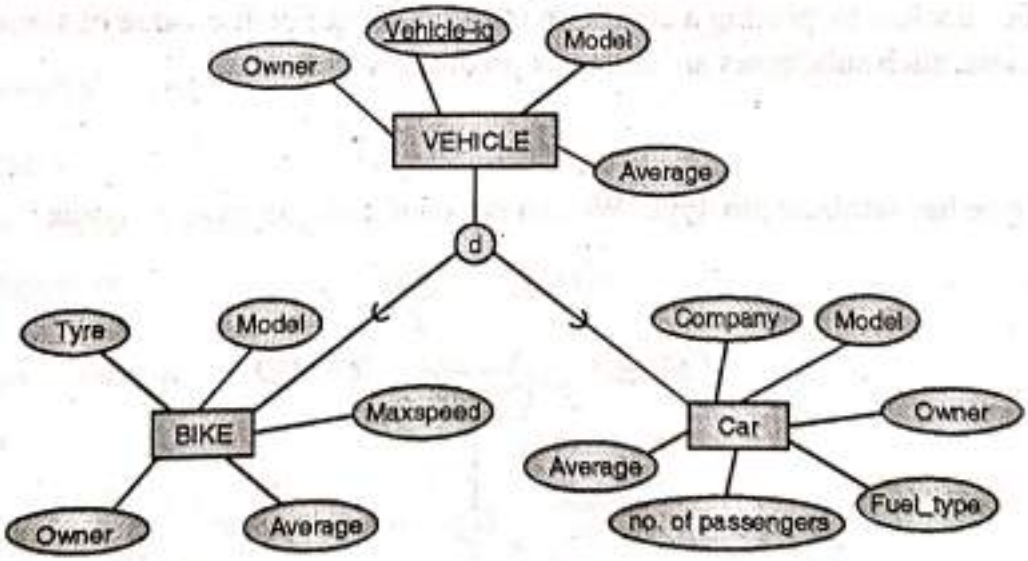


Fig. 3.6.5 : Generalized VEHICLE entity

3.6.4 Constraints and Characteristics of Specialization and Generalization

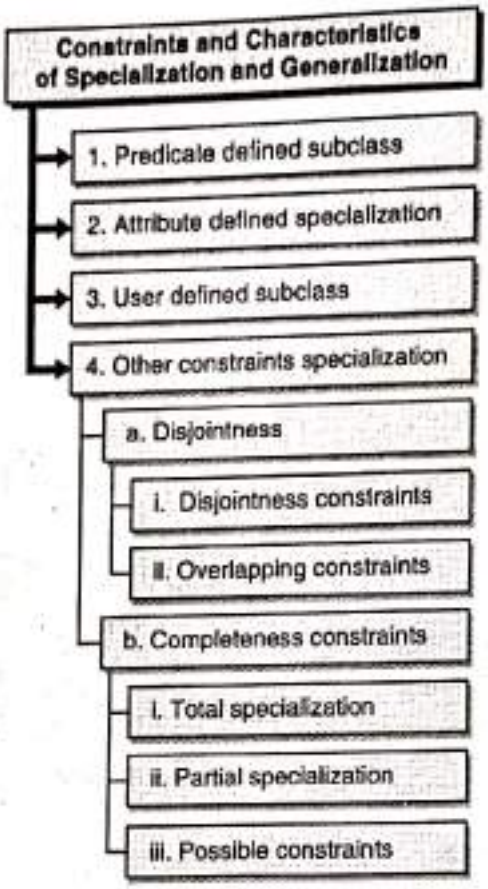


Fig. 3.6.6

(1) Predicate defined subclass

In specialization sometimes we can find exactly which entities will become member of specific subclass by placing a condition (or a predicate) on the value of some attributes of superclass, such subclasses are called as predicate subclasses.

Example :

Employee has attribute job_type. We can put condition job_type = 'typist'

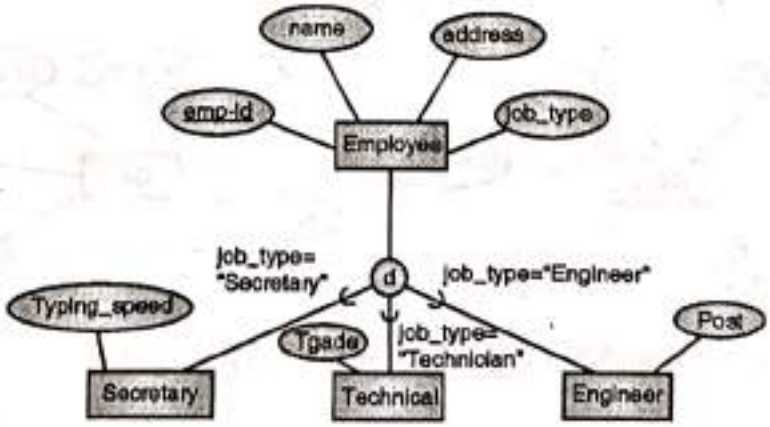


Fig. 3.6.7 : Predicate defined specialization

(2) Attribute defined specialization

- If all subclasses in a specialization have their membership condition on some attribute of superclass then the specialization itself is called as attribute defined specialization.
- This kind of attribute is called as defining attribute of specialization.

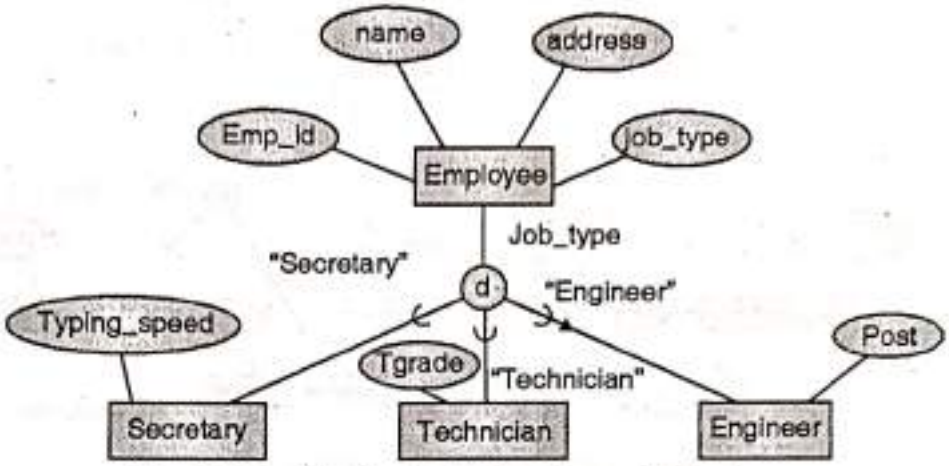


Fig. 3.6.8 : Attribute defined specialization

(3) User defined subclass

- When we do not have a predefined condition for determining membership to a subclass hence user needs to specify condition for such specialization then the subclass is known as user defined subclass.
- Membership in such a subclass is determined by database user, when database user applies the operations to add an entity to subclass.
- Hence membership is specialized for each entity by user and not by any condition that may be evaluated automatically.

(4) Other constraints specialization

(a) Disjointness

(i) Disjointness constraints

Disjointness constraints specify that the subclasses of specialization must be disjoint that means entity can be a member of the most one subclass of specialization.

Example :

- New employee can become member of only one subclass like Assistant, Technician or Engineer only.
- Attribute defined Specialization determines the disjointness constraints.
- If attribute is used to define the membership then predicate should be single valued.

Example :

- In employee entity Job_Type predicate that must be single valued.
- Disjoint subclass indicated by encircled d as shown in Fig. 3.6.9.

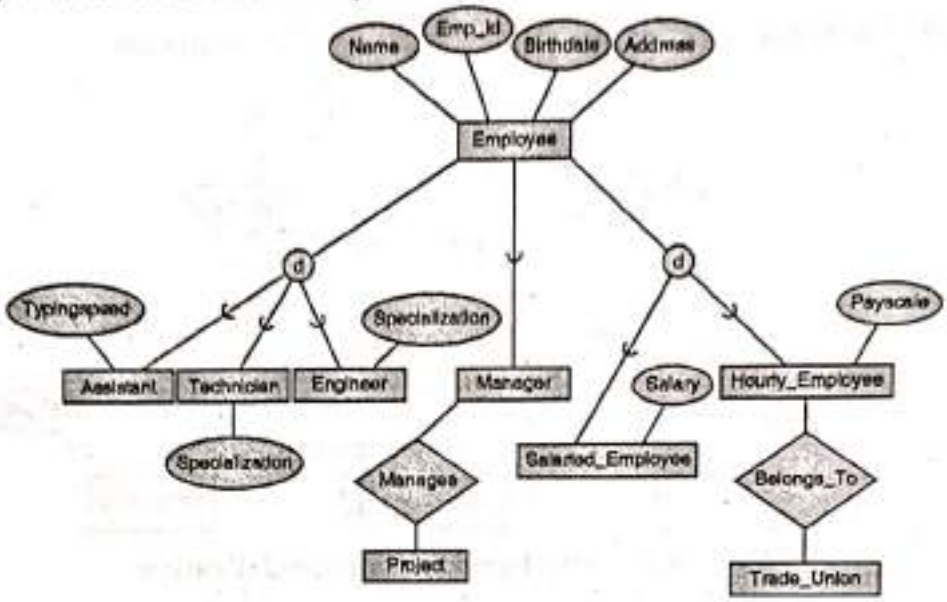


Fig. 3.6.9 : Disjoint constraints

(ii) Overlapping constraints

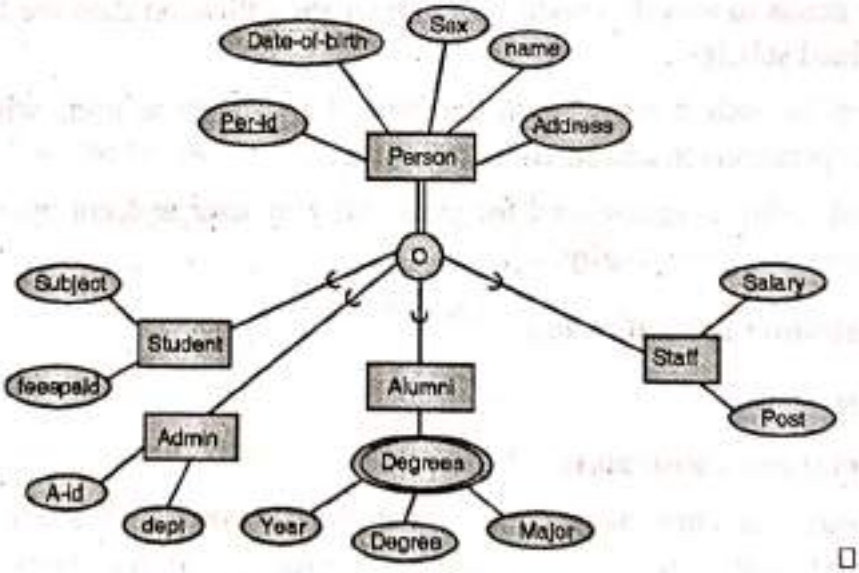


Fig. 3.6.10 : Overlapping constraint

- The subclass is not always required to be disjoint, then such set of entities can be overlaped i.e. entity may part of more the subclasses.

Example :

- Person can become member of subclasses like student, Admin or Staff. It is possible to have person belonging to more than one subclasses.
- Disjoint subclass is indicated by encircled 'O'.

(b) Completeness constraints

(i) Total specialization

- A total specialization constraints specifies that entity in super class may be member of at least one of the subclass in the specialization.
- A double line is used to represent total specialization in EER diagram.

Example :

An employee must belong to salaried employees or hourly employee.

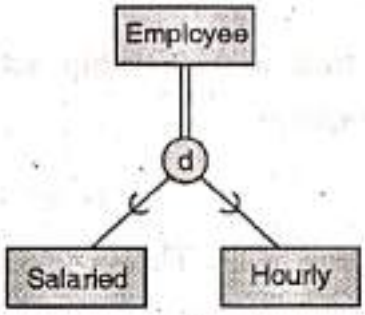


Fig. 3.6.10(a) : Total specialization

(ii) Partial specialization

- An entity can either belong to a subclass or not belongs to any subclass.
- A single line used to represent partial specialization in EER diagram.

Example :

An employee can be Engineer, Secretary or Technician.

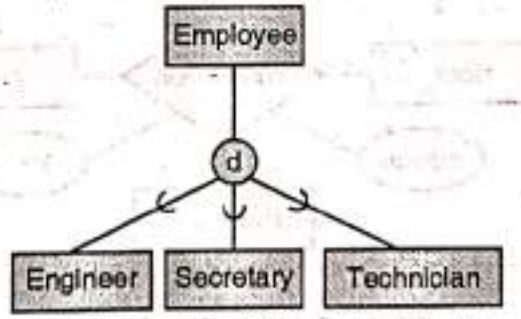


Fig. 3.6.10(b) : Partial specialization

(iii) Possible constraints

- Disjoint, total constraints
- Disjoint, partial constraints
- Overlapping, total constraints
- Overlapping, partial constraints

3.7 Aggregation

Q. Write short note on : Aggregation. **(5 Marks)**
Q. Explain aggregation with the help of an example. **(5 Marks)**

- Aggregation is meant to represent a relationship between a whole object and its component parts.
- It is used when we have to model a relationship involving entity sets and a relationship set.
- Aggregation allows us to treat a relationship set as an entity set for purpose of participation in (other) relationships.

Example :

- A Project is sponsored by a department. This is a simple relationship.
- An Employee monitors this sponsorship (and not project or department). This is aggregation. Monitors are mapped to the table like any other relationship set.

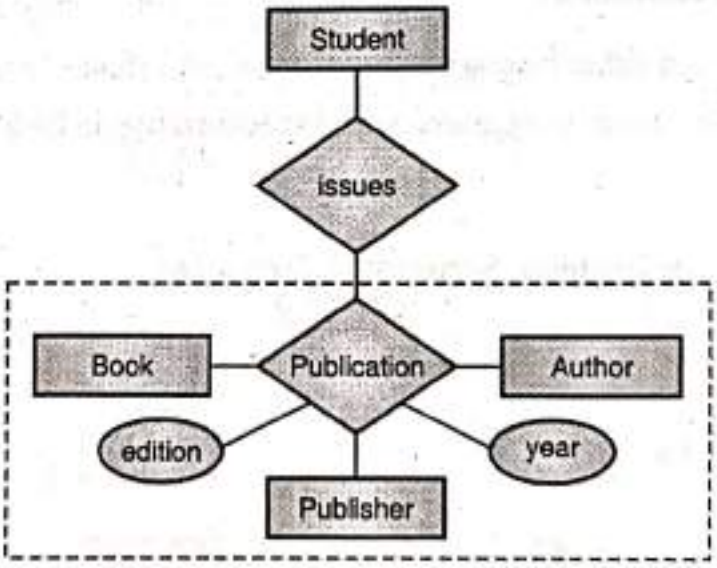


Fig. 3.7.1 : Aggregation

3.8 Solved ER Designing Examples

Example 3.8.1 : A publication may be a book or an article. Articles are published in Journals. Publication has title and location. Book having their title and category. Article includes title, Topic and date. Publication is written by Authors stores Name, address and mobile number. Publication also belongs to particular subject which has their names. **(10 Marks)**

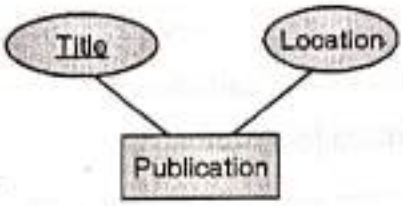
Soln. :

Step 1 : Identify entities

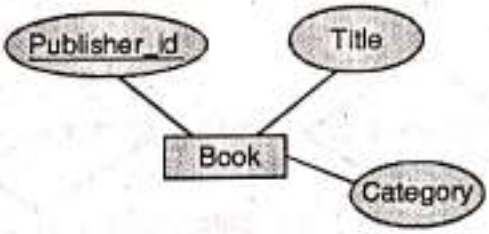
1. Publication
2. Book
3. Article
4. Journal
5. Subject
6. Author

Step 2 : Identify attributes

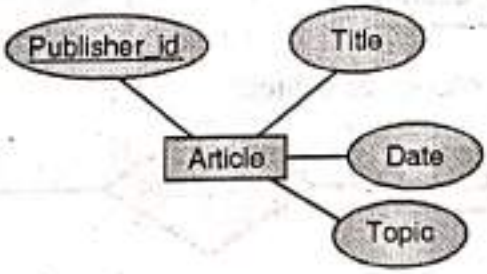
1. Publication (Title, Location)



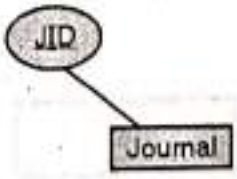
2. Book (Publisher_id, Title, Category)



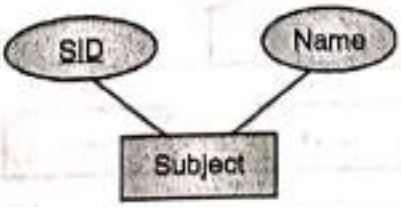
3. Article (Publisher_id, Title, Date, Topic)



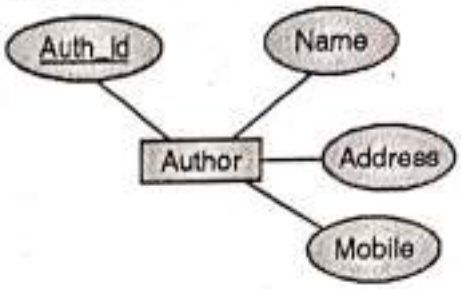
4. Journal (JID)



5. Subject (SID, Name)

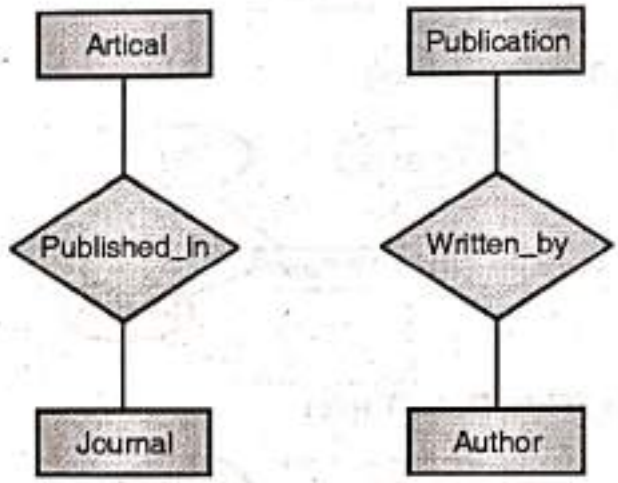


6. **Author (Auth_id, Name, Address, Mobile)**



Step 3 : Identify relationships

1. Articles are published in Journal.
2. Publication is written by Author.

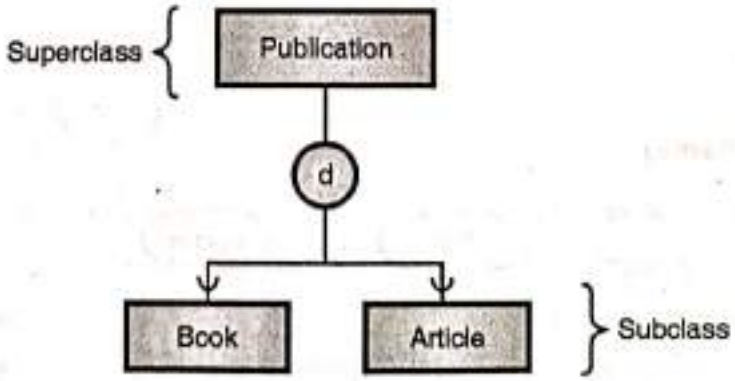


3. Publication belongs to a particular subject.

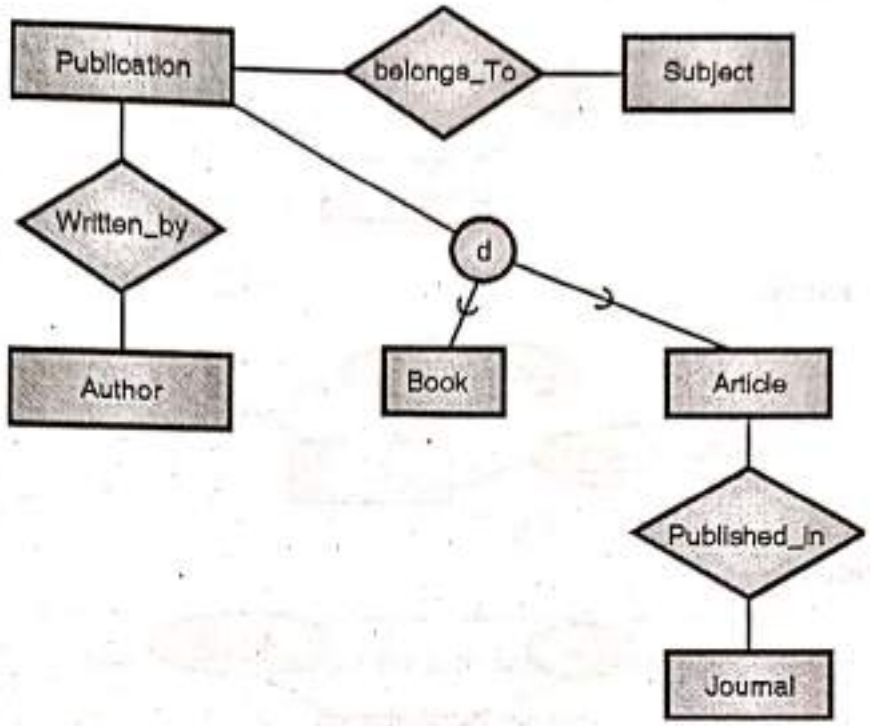
Step 4 : Identify inheritance relations

Publication can be

BOOK or ARTICLE.



Step 5 : Merging all above relations we will get final ER model



Example 3.8.2 : Construct on E-R diagram for a car-insurance company that has a set of customers each of whom owns one or more cars. Each car has associated with it zero to any number of recorded accidents. **(10 Marks)**

Solution :

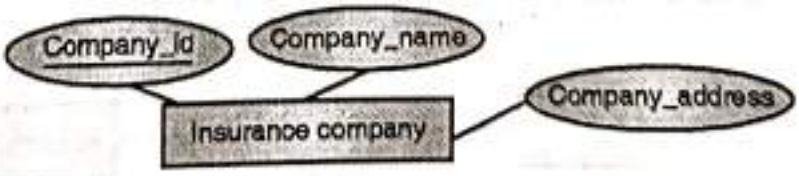
(1) Identify all entities

(a) Insurance company (b) Customer

(c) Car (d) Accidents

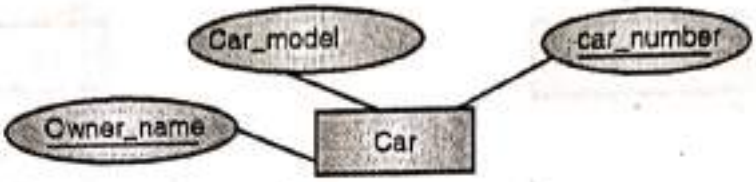
(2) Identify all attributes

(a) Company entity



(b) Customer entity

(c) Car entity



(d) Accidents

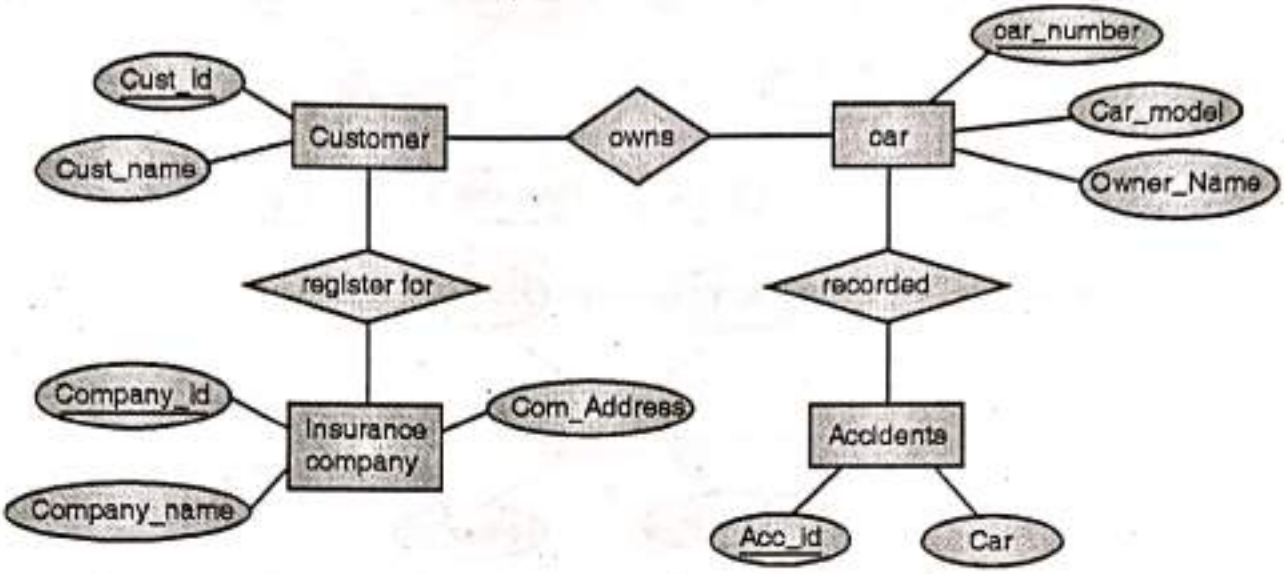
(3) Identify all relationship

a) Car insurance company has a set of customers

b) Customer owns one or more car

c) Each car associated with zero or any number of accidents

(4) Construct ER diagram by merging all above relationships



Example 3.8.3 : Construct an ER diagram for a hospital with a set of patients and the set of medical doctors associated with each patient a record of various text and examination conducted. **(10 Marks)**

Solution :

(1) Identify Entities

(i) Hospital (ii) Patient

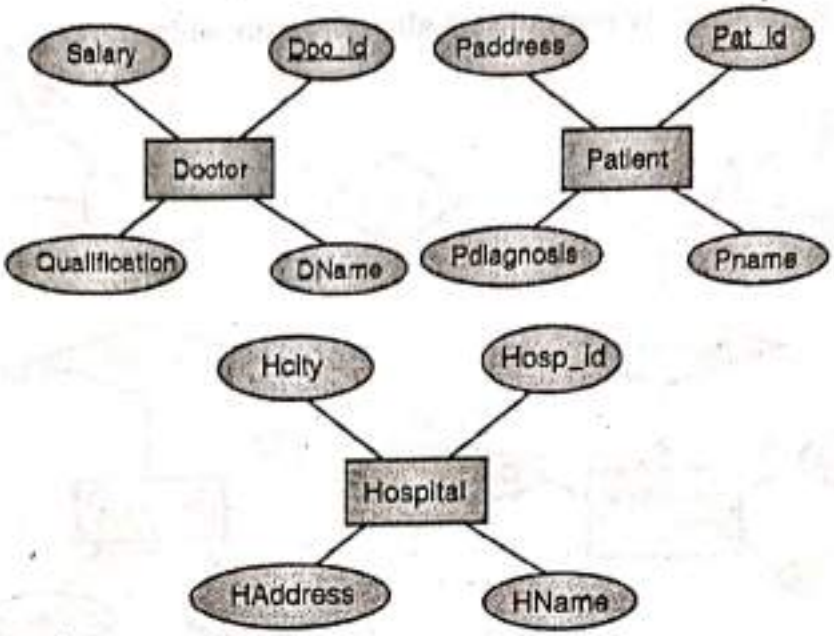
(iii) Doctors (iv) Medical-record (Record of various test and examination conducted)

(2) Identify Attributes

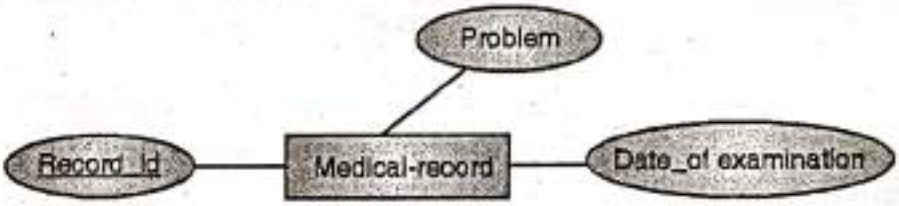
(i) Hospital (Hosp_id, HName, HAddress, Hcity)

(ii) Patient (Pat_id, Pname, Pdiagnosis, Padress)

(iii) Doctor (Doc_id, DName, Qualification, salary)



(iv) Medical_Record (Record_id, Date_of examination, Problem)



(3) Identify relationships

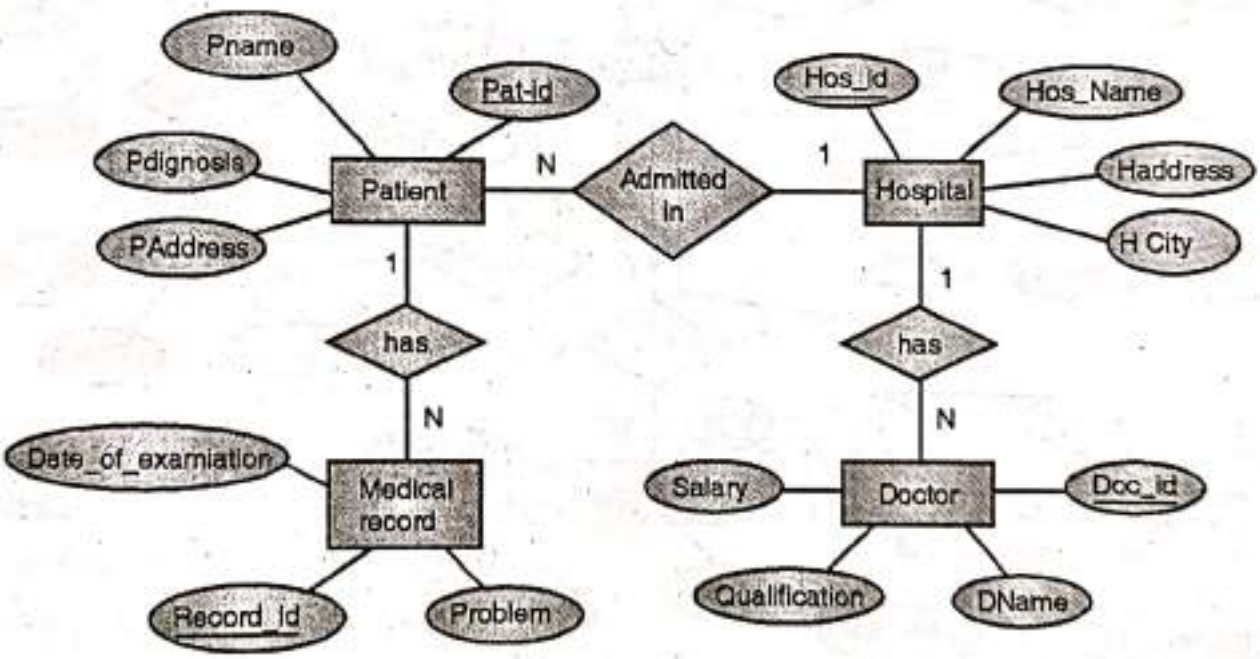
(a) Hospital has a set of patients

(b) Hospital has a set of doctors

(c) Doctors are associated with each patient

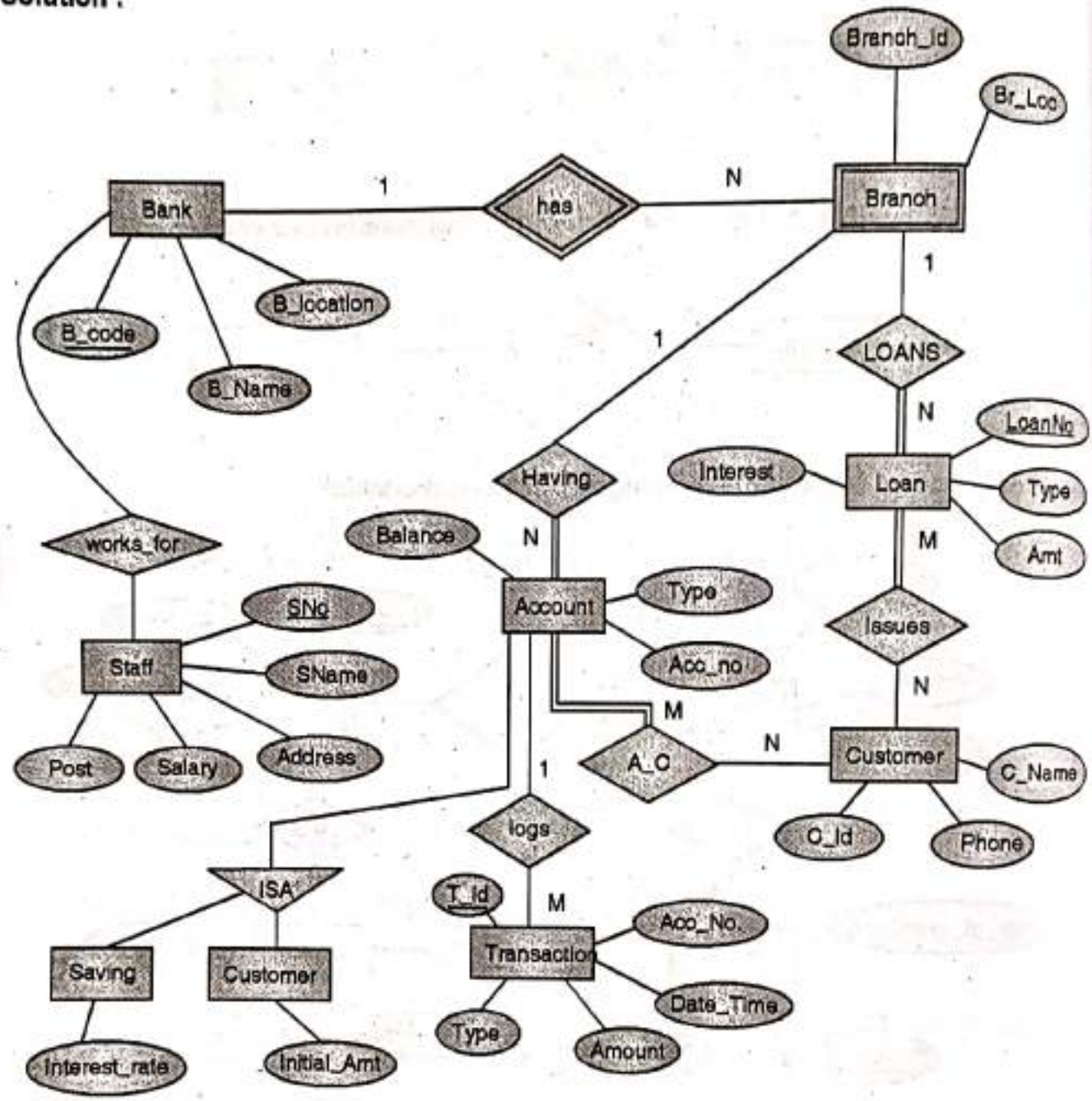
(d) Each patient has record of various test and examination conducted

(4) Construct ER Model from merging all above relationship



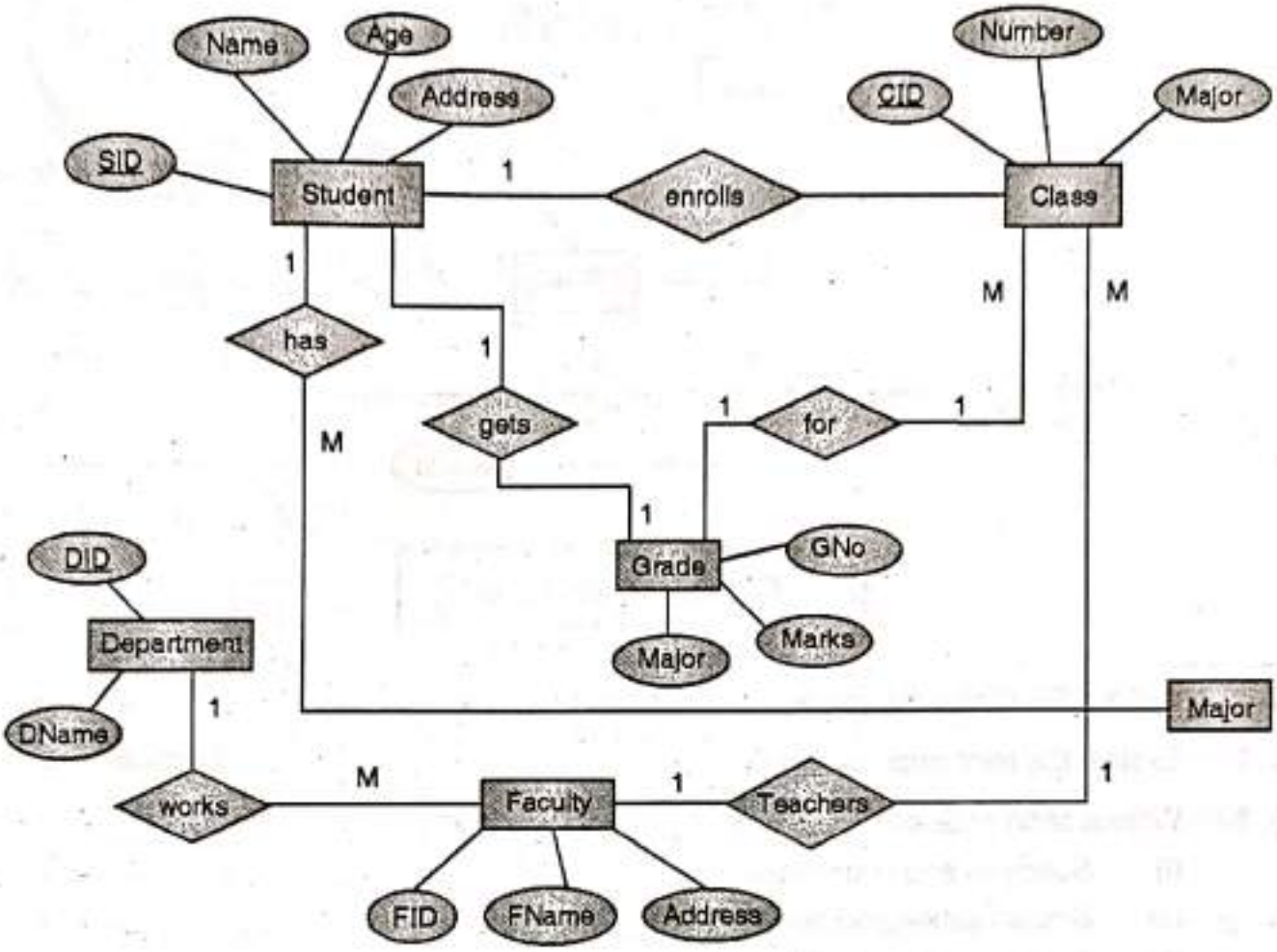
Example 3.8.4 : Draw ER Diagram for banking enterprise. **(10 Marks)**

Solution :



Example 3.8.5 : Draw ER Diagram for University database consisting four Entities Student, Department, Class and Faculty. Student has a unique Id, the student can enroll for multiple classes and has a most one major. Faculty must belong to department and faculty can teach multiple classes. Each class is taught by only faculty. Every student will get grade for the class he/she has enrolled. **(10 Marks)**

Solution :



Example 3.8.6 : Draw an ER diagram for the education database that contains the information about an in house company education training scheme. The relevant relations are :

Course (course-no, title)

Offering (course-no, off-no, off-date, location)

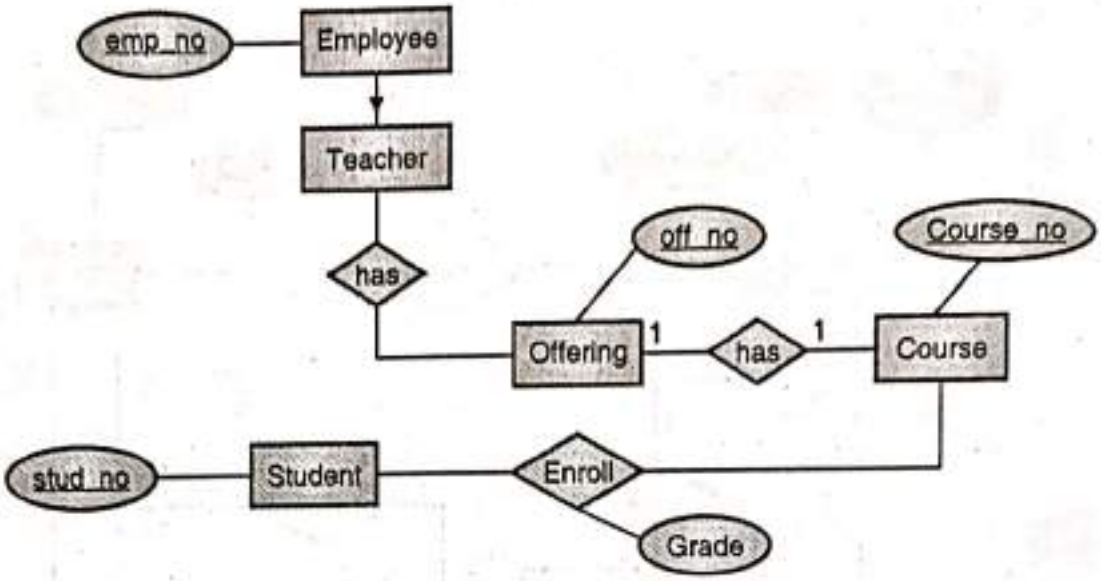
Teacher (course-no, off-no, emp-no)

Enrolment (course-no, off-no, stud-no, grade)

Employee (emp-no, emp-name, job)

Student (stud-no, stud-name, ph-no) **(10 Marks)**

Solution. :



Review Questions

Q. 1 Explain ER Diagrams and its components.

Q. 2 Explain the term aggregation.

Q. 3 Write a short note on :

(i) Subclass and superclass

(ii) Specialization and generalization

(iii) Type Inheritance

(iv) Weak entity set

Q. 4 Describe various constraints of specialization and generalization.

Q. 5 Write short note on : Total participation, partial participation.

Q. 6 Define degree.

Q. 7 Write short note on : Extended e-r features.

Q. 8 What is strong entity and weak entity ? Explain with example.

Q. 9 Define Derived attribute.

Q. 10 Explain generalization and specialization.

Q. 11 Explain Total participation and Partial participation.

Q. 12 Explain different types of attributes in ER Model.

CHAPTER 4

Relational Data Model

Module III

Syllabus

Introduction to the Relational Model, relational schema and concept of keys, Mapping the ER and EER Model to the Relational Model.

4.1 Relational Model

Q. Define Relation. **(2 Marks)**
Q. Define Attributes. **(2 Marks)**

1. Introduction

- The relational model first proposed by E. F. Codd hence he is known as father of relational model.
- Relational database was an attempt to simplify database structure by making use of tables and columns.
- Tables are known as **relations**, columns are known as **attributes** and rows (or records) are known as **tuples**.

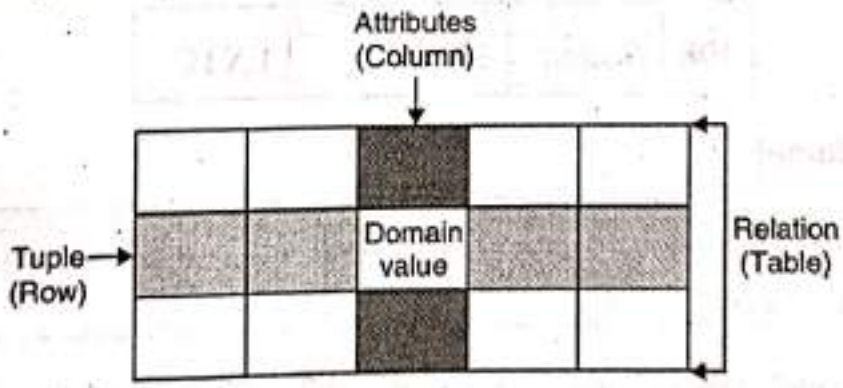


Fig. 4.1.1 : Relational Algebra Notations

2. Relation (Table)

Definition : *Relations are a logical structure which is a collection of tables consisting horizontal rows also called as tuples and vertical columns also called as* ***Attributes.***

- The table containing rows and columns represents entity in relational model, it is called as Relation.
- This concept doesn't represent how the data is stored in the physical memory of computer system.
- The relation can contain data about a single entity or relationship between two entities.

Characteristics of Relation

1. A table composed of rows and columns.
2. Each table in a database has its unique *table name*.
3. Each table row (tuple) represents a single entity occurrence within the entity set.
4. All values in a same column must conform to the same format of data.
5. Each table must have a single attribute or set of attributes that uniquely identifies each row.

Example :

The student relation can be shown as given below,

Id	Name	Age	Class	Branch
105	Mahesh	25	BE	IT
106	Suhas	28	FE	CS
107	Jay	29	SE	CS
108	Sachin	30	TE	EXTC

3. Attributes (Column)

Definition : *Relation has its own properties which describes that relation (table) such properties are known as attributes.*

- In relational model, the column in relation (table) or field of data is also called as Attribute.
- Every table must have at least one column in it.
- The single attribute will contains the similar type of data of all entities in relation.

Name
Mahesh
Suhas
Jay
Sachin

- It is not possible to have multiple columns with same column name in the same relation. But it is possible to have multiple columns with same column name in two different relations.
- The SQL standard does not specify any maximum number of columns in a table.

Example :

The Name attribute in above student relation will contains the name of all student entities in student relation.

4. Tuple (Row / Records)

- A single row in relational table which contains all the information about a single entity is called as Tuple.
- The single row in relation (Table) is called as Tuple.
- Each horizontal row of the student table represents a Student tuple.
- A table can have any number of rows in it.

Stud_Id	Name	Age	Std	Div
105	Mahesh	25	BE	A

Example :

The above tuple contains all data about the Id 105, student entity in student relation.

5. Domain (Data Value)

- The intersection column and row in a relational table which represents data of entity is called as Domain.
- Every column in a table has a set of data values that are allowed for that column which is called as Domain.
- In a relational table a domain can have a single value or no (Null) value.
- The single domain will contains the specific data of single entities in relation.

Name
Mahesh

Example :

The Name attribute of tuple id 105 will contains the name of all student with id 105 in student relation.

4.2 Relational Database Schema

Q. Write a note on Relational Database Schema. (2 Marks)
Q. Compare Relational Database Schema and Relational Schema with example. (2 Marks)

(1) Introduction

- The term schema refers to the organization or structure of data in relational database.
- The relational schema describes structure of relation (i.e. table) and relational database schema explains the structure of relational database.

(2) Relational Schema

Relational schema consists of a number of attributes associated with relation.

Example :

EMPLOYEE	(Ssn, Ename, Bdate, Address, Dnumber)
DEPARTMENT	(Dnumber, Dname, Dmgr_ssn)
DEPT_LOCATIONS	(Dnumber, Dlocation)
PROJECT	(Pnumber, Pname, Plocation, Dnum)
WORKS_FOR	(Ssn, Pnumber, Hours)

Fig. 4.2.1 : Sample Relational Schema

(3) Relational Database Schema

Relational database schema consists of a number of relation schemas associated with that database.

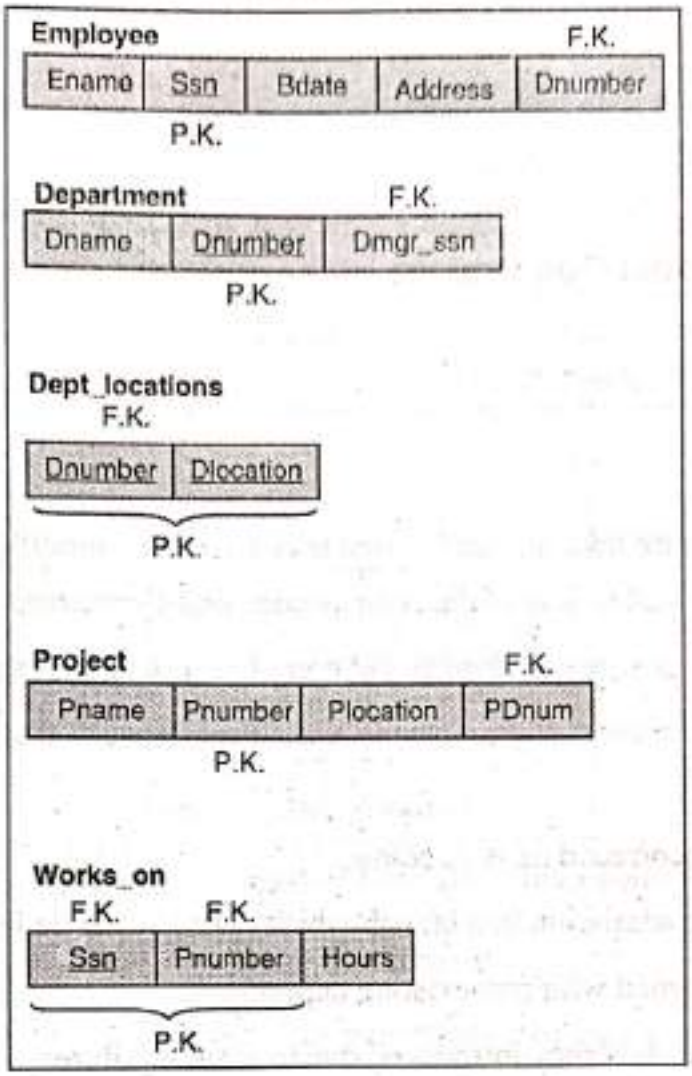


Fig. 4.2.2 : Sample Relational Database Schema

(4) Comparison

- The attributes are grouped to form a relation schema by mapping a conceptual data model i.e. ER or EER data model to relational schema.
- The relational mapping will identify entity types and relationship types and their respective attributes.
- The relation schema consists of a number of attributes in relation while the relational database schema consists of a number of relation schemas in the corresponding database.

Relational Schema (Table Structure)	Relational Database Schema (Database Structure)
$R_1 (A_1, A_2, .., A_n)$ $R_2 (B_1, B_2, .., B_n)$ ... $R_m (C_1, C_2, .., C_n)$	$R_1, R_2, ..., R_n$

Where,

$R_1, R_2, \ldots R_n$ = Relation or Tables

A, B, C = Attributes or Columns

4.3 Relational Model Constraints

Q. Explain different integrity constraints. (10 Marks)

(1) Introduction

- Constraints makes sure that only authorised user will make modifications to database and changes should not lead to loss of data consistency and correctness.
- These concepts makes sure correctness and completeness of data stored in the database.
- This objective can never be guaranteed, one cannot ensure that every entry made in database is accurate.
- Some example of incorrect data is as below,
  1. Student taking admission to a branch which is not available in the college.
  2. Employee assigned with non existing department.
  3. Sometimes inconsistency introduced due to system failures.

(2) Types of Relational Constraints

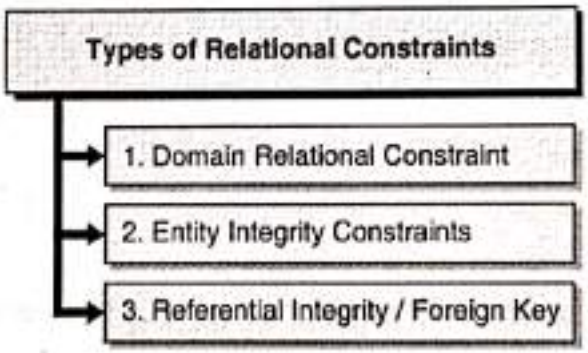


Fig. 4.3.1 : Types of Relational Constraint

4.4 Domain Relational Constraint

Q. Explain different integrity constraints. (10 Marks)

Q. Write any short note on Constrains in SQL. MU - Dec. 18, 10 Marks

1. Introduction

- Domain constraints allow us to test whether the values inserted into the database are correct or not.
- The CREATE TABLE Command may also include domain constraints which can check integrity of database.
- These domain constraints are the most basic form of integrity constraint.

2. Types of Domain Constraints

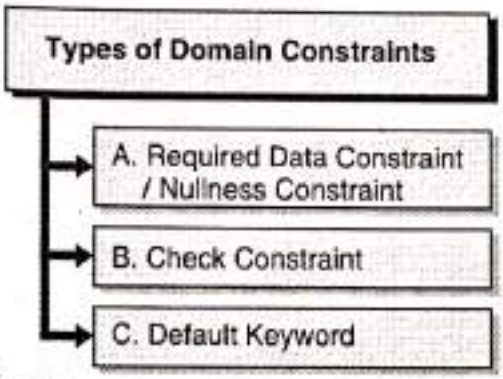


Fig. 4.4.1 : Types of Domain Constraints

A. Required Data Constraint / Nullness Constraint

- The database may have some attributes mandatory like user registration must have user email address.
- These attributes (columns) in a database are not allowed to contain NULL values or blanks.

Example :

In the student table, student must have an associated student name.

```
Student_Name varchar(100) NOT NULL
```

- Therefore now, the Student_Name column in the STUDENT table is a required data column. It is not possible to insert Null value in Student_Name column of Student table.
- The DBMS can prevent user from inserting NULL values in any column with help of such constraints.

B. Check Constraint

- The **check constraint** is used to ensure that attribute value satisfies specific condition as specified by data requirements or user.
- Suppose in Student Table, gender of student can be male or female only.

- The DBMS can prevent user from entering incorrect or other data in database table.

Example :

Table with student entity having gender which can be M or F.

Hence, attribute gender can take only two values either 'M' or 'F'.

```
Student_Gender varchar(1) CHECK (gender IN ('M','F'))
```

C. Default Keyword

- Default keyword is used to add a default specified value, if attribute value is not provided by user.
- It avoids the addition of NULL value to the database by inserting default value as specified by the developer while creating a table.

Example :

Table with customer entity having name and gender.

- If name is not added for customer that will be taken as 'Unknown' if we specify DEFAULT value of NAME column to 'UNKNOWN'

```
Student_Name varchar(50) DEFAULT 'UNKNOWN'
```

4.5 Entity Integrity Constraints

Q. Explain different integrity constraints. (10 Marks)
Q. Explain the term : Primary Key. (2 Marks)

1. Introduction

- Entity constraints allow us to test whether the tuple (entity) inserted into the database are correct or not.
- The create table Command may also include entity constraints which can primary key of table.

2. Types of Entity Constraints

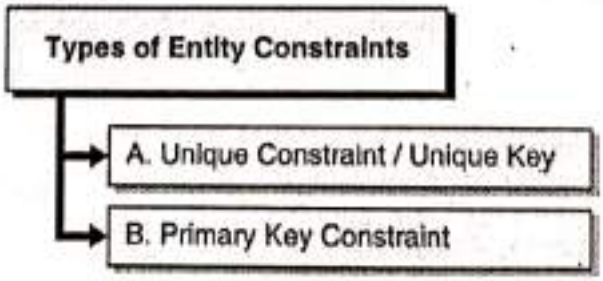


Fig. 4.5.1 : Types of Entity Constraints

A. Unique Constraint / Unique Key

- In case of unique constraint no two tuples can have equal value for same attributes.
- This constraint says that attributes forms candidates key, which allows one Null value which is unique by itself.
- This UNIQUE constraint can be applicable to user defined domain declaration also.

Example :

```
EMAIL varchar(30) UNIQUE
```

B. Primary Key Constraint

- A table in a relational database has one column or combination of some columns whose values uniquely identifies a single row in the table. This column or combination of columns is called the **primary key** of the table.
- Primary key attribute is same as unique key constraint with NOT NULL constraints (Unique constraint+ Not Null constraint).
- The main difference in unique constraint and primary key constraint is that one null value is allowed in unique constraint which can be treated as unique value while nulls are not allowed in primary key constraint.
- For example, each row of the STUDENT table has a unique set of values in its STUDENT_ID column, which uniquely identifies the student represented by that row.
- Duplicate values are not allowed in primary key column, because they cause problems in distinguishing one entity from another (entity may be an employee).
- The DBMS can prevent user from inserting same data values in a column again and again.

```
STUDENT_ID char(10) PRIMARY KEY
```

4.6 Referential Integrity / Foreign Key

Q. Discuss what is meant by term : Referential Integrity. (3 Marks)

1. Introduction

- A value appearing in a one relation (table) for a given set of attributes also appears for another set of attributes in another relation (table). This is called referential integrity.
- The referential integrity constraint is specified between two tables to maintain the consistency among tuples in the two tables.
- The tuple in one relation refers only to an existing tuple in another relation.

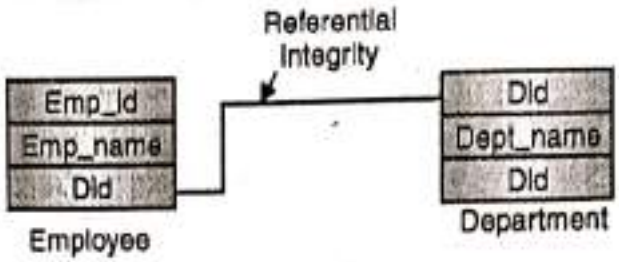


Fig. 4.6.1 : Referential integrity

Employee Table		
Emp_Id	**Emp_name**	**Did**
1	Sachin	20
2	Suhas	10
3	Jay	20
4	Om	10

Department Table	
Did	**Dept_name**
10	HR
20	TIS
30	L&D

- In the above example Employee table has Did as foreign key reference to Did column in Department table this is called as referential integrity.
- Here we are forcing the database to check the value of Did column from the department table while inserting any value in Employee table. This helps to maintain data consistency.

2. Foreign key violations in SQL

- If any row in EMP table is added with 'Did' value which is not there in department table the insert statement will give foreign key violation error.
- In above tables we will refer Department as parent table (as it is containing Primary key) and Employee table as Child table (as it is containing Foreign Key).
- There are 4 problems causes the foreign key violations,

(a) Adding new tuple to Child Table (Add Child)

- If we try to add an employee with Did 70 to employee table (Child Table), it will return foreign key violation error. As Did 70 is not there in Department table (Parent table).

```
INSERT INTO Employee
VALUES (11,'Devid', 70);
```

Output

ORA-02291 : integrity constraint (Employee.FK_Employee) violated - parent key not found

Adding new employee		
Emp_Id	Emp_name	Did
11	Devid	70

- This functionality helps to maintain data consistency in database.

(b) Updating tuple from Child Table

- If we try to update an employee Emp_Id = 2 with Did as 70 to employee table (Child Table), it will return foreign key violation error. As Did 70 is not there in Department table (Parent table).

```
UPDATE Employee
SET Did = 70
WHERE Emp_Id=2;
```

Output

ORA-02291 : Integrity constraint (Employee.FK_Employee) violated - parent key not found.

- This functionality helps to maintain data consistency in database.

(c) Deleting tuple from Parent Table

- If we try to delete department of Did = 10 from Department table (Parent table), it will return foreign key violation error. As there are few employees working in department with Did =10.

```
DELETE Department
WHERE Did=10;
```

Output

ORA-02292: integrity constraint (Employee.FK_Employee) violated - child record found

- This functionality will creates limitation for deletion of parent record if it has some associated child records.

(d) Updating tuple from Child Table

- If we try to update department of Did = 10 with Did = 70, it will return foreign key violation error. As there are few employees still working in department with Did =10.

```
UPDATE Department
SET Did = 70
WHERE Did = 10;
```

Output

ORA-02292 : integrity constraint (Employee.FK_Employee) violated - child record found

- This functionality will creates limitation for updating parent record if it has some associated child records.

3. Delete-Update (DU) rules to solve problem of foreign key violation

- If any row in EMP table is added with 'Did' value which is not there in department table the insert statement will give foreign key violation error.
- This rule can be enforced as given below,

```
Create Table Employee
(
    Eid varchar (50) Primary Key,
    ....
        Did varchar (50) foreign key references department (Did)
        On delete CASCADE
        On update CASCADE
    )
```

(a) NO ACTION / RESTRICT

- This clause will discards the delete or update operation on the parent table.
- In this case the database engine will not allow user to delete the row and using FK violation error.
- The RESTRICT rule will not allow you to delete a row from the parent table although as there is corresponding row present in child table.
- Foreign key (Did) references department :
  - On delete RESTRICT
  - On Update RESTRICT
- The database engine will give the error and the delete action on the row in the parent table is ignored.
- Deletion of department is not allowed as there is some employees are present in that department.

(b) CASCADE

- Corresponding rows are deleted from the referencing table (Child table), if that row is deleted from the parent table.
- Foreign key (Did) references department :
  - On delete CASCADE
  - On Update CASCADE
- If a department is deleted then all the employee records that refers to the deleted department are also been deleted.

(c) SET NULL

- Foreign key data value is set to NULL, if the corresponding row in the parent table is deleted.
- For this constraint to execute, the foreign key columns must be nullable.
- Foreign key (Did) references department :
  - On delete SET NULL
  - On Update SET NULL
- Insert Null value of did in the place of deleted did in employee table.

(d) SET DEFAULT

- Foreign key data values refer to non-existing foreign key are set to their default values.
- For this constraint to execute, all foreign key columns must have default definitions.
- Foreign key (Did) references department:
  - On delete SET DEFAULT
  - On Update SET DEFAULT
- If a column is null able, and there is no explicit default value set, NULL becomes the implicit default value of the column.
- Insert any default value of 'did' (which exists in the departing table) in the place of deleted 'Did'.

4. Self-referential relations

- A foreign key constraint can reference some other columns in same tables.
- It form self-referential relations.

For Example, EMPLOYEE (employee_id, employee_name, manager_id)

- Because the manager is also employee, there is a foreign key relationship from the **manager_id** column to the **employee_id** column in same table.

4.7 Concept of Keys

Q. Explain concept of keys relational Table. (3 Marks)

- The column value that uniquely identifies a single record in a table called as key of table.
- An attribute or set of attributes whose values uniquely identify each entity in an entity set is called a key for that entity set.
- ID is a key of student table. It is possible to have only one student with a one ID (Say only one student '**Mahesh**' with ID = 1)

Key type	Definition
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row. (Explained in details in previous section Entity Integrity)
Secondary key	An attribute (or combination of attributes) used strictly for data retrieval purposes.
Foreign key	An attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null. (Explained in details in previous section Referential Integrity)

4.8 Mapping Entities to Tables

Q. Explain the steps of an algorithm for ER to relational mapping. (10 Marks)
Q. Explain the algorithm to map ER and EER model to relational model in detail. (10 Marks)

(1) Regular entity types

- **Tables :** Regular entity sets can be represented as table in relational model.
- **Columns :** Attributes of entity set can be converted to the columns (attributes) of the tables in relational model.

Example :

Regular entity employee mapped as employee table in object model like 'Stud_id', 'Stud_Addr' etc. are shown as table columns.

ER model

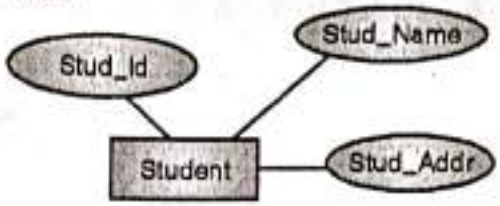


Fig. 4.8.1 : Regular entity

Stud_Id	Stud_Name	Stud_Addr
1	Snehal	Mumbai
2	Pratiksha	Mumbai
3	Supriya	Mumbai
4	Tanmay	Goa

(2) Weak entity types

For each weak entity type with owner entity, create a table and include all simple attributes of weak entity type as columns of table, including foreign key attributes as the primary key attribute of the table that correspond to the owner entity type.

Example :

Dependents (Weak entity) in Employee (Owner entity).

ER Model

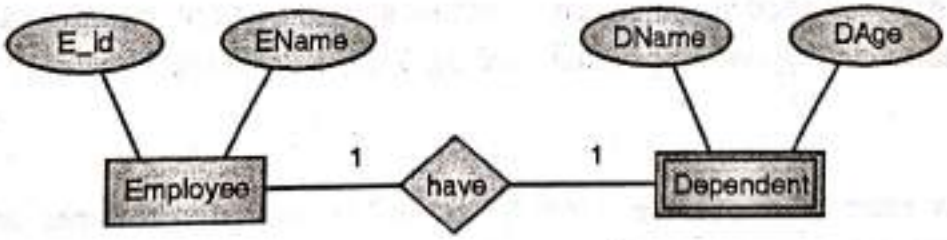


Fig. 4.8.2 : Weak entity

E_Id	Ename	DName	DAge
1	Sachin	Jyoti	23
2	Suhas	Manju	22
3	Jayendra	Tanya	27

4.9 Mapping Attributes to Columns of Table

Q. Explain how to convert attribute in ER to relational Table. (5 Marks)

(a) Simple attributes

Simple attribute can be directly converted to a column (Attribute) in relational model.

Example :

Employee 'Age' can be directly converted to column.

Fig. 4.9.1 : Simple attribute

Employee table

Eid	Age
1	23
2	24
3	43
4	28

(b) Composite attributes

These attributes need to be stored as set of simple component attributes (Columns) in relational model by avoiding actual attribute ('name' in below example).

Example :

In below example composite attribute 'Name' is converted to three columns in object model by avoiding actual attribute 'Name'.

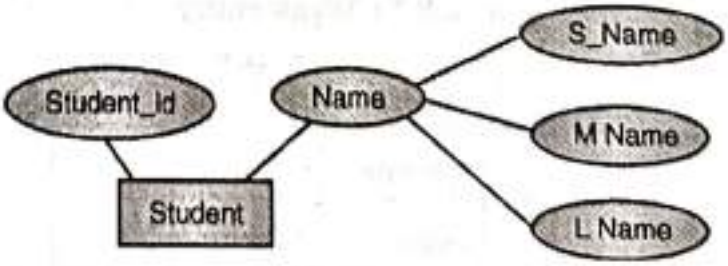


Fig. 4.9.2 : Composite attribute

Student table

Student_id	S_Name	MName	LName
1	Harshad	Rupali	Malar
2	Bipin	Anand	Shinde
3	Aanand	Ganesh	Panchal
4	Tushar	Bipin	Pimple

(c) Multi valued attributes

Multi valued attributes are mapped as a relation which includes combination of the primary key of table and multi valued attribute as a composite primary key as shown in Fig. 4.9.3.

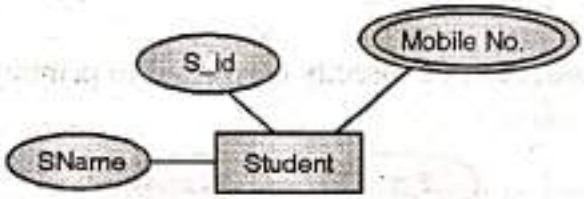


Fig. 4.9.3 : Multi valued attribute

Mobile table	
Sid	**Mobile No.**
1	9891295492
1	9821959241
2	8080918456
3	8095124890
3	9773112456

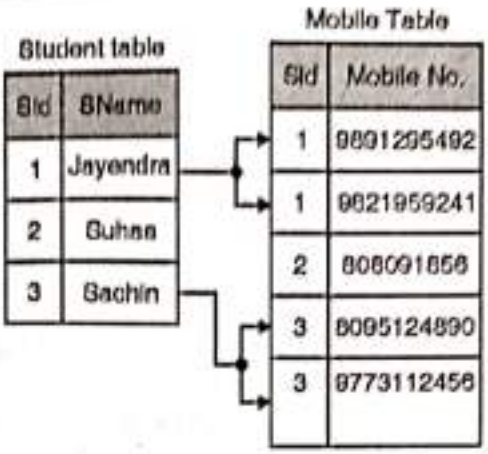


Fig. 4.9.4

(d) Derived attributes

There is no need to store such attribute in relational model. It will be calculated from stored attribute.

(e) Key attributes

Key attribute in ER Model can be directly converted to primary key attribute of relational model.

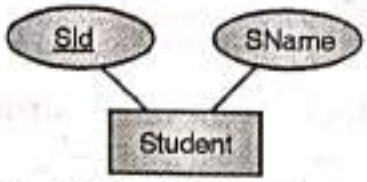


Fig. 4.9.5 : Key attributes

Student table

Sid	SName
1	Deepak
2	Vaibhav
3	Yogita
4	Bency

4.10 Mapping Relationships

Q. Explain how to convert various type of relations in ER to relational Table. (5 Marks)

(a) Foreign key approach

If binary relationship type does not possess many attributes then we can map such relation using foreign key.

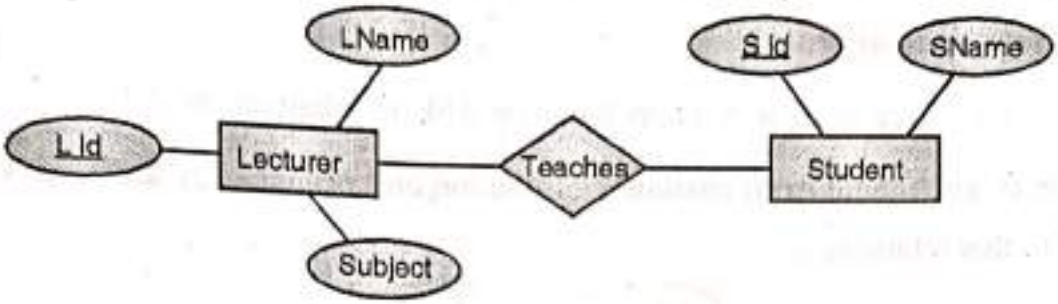


Fig. 4.10.1 : Foreign key approach

Lecturer table		
Lid	**LName**	**Subject**
1	Omprakash	IP
2	Yogesh	INS
3	Amit	PM

Student table		
Sid	**SName**	**Lid**
1	Bency	1
2	Deepak	1
3	Yogita	1
4	Snehal	2
5	Pratiksha	2

Lid is foreign key in student table while primary key in lecturer table.

(b) Merged relationship approach

When participation is total it is possible to merge relation and involved entities as a single relation and then map it to a table.

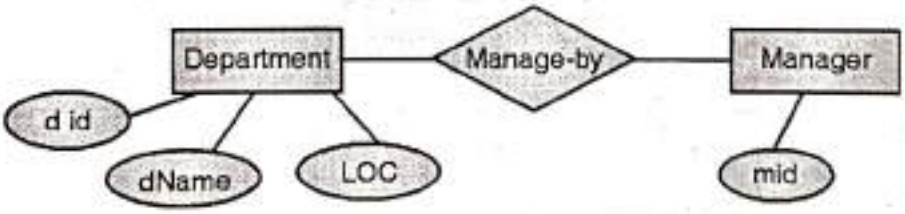


Fig. 4.10.2 : Merged relationship approach

Department table			
did	**Dname**	**Loc**	**mid**
10	IDF	Mahape	11
20	Mayban	Pune	22
30	Lax	Mumbai	33

(c) Cross reference approach

- A relationship type in EER is mapped to new table in relational model.
- Column of such table is all attributes of relation and primary key attributes of all tables linked to this relation.

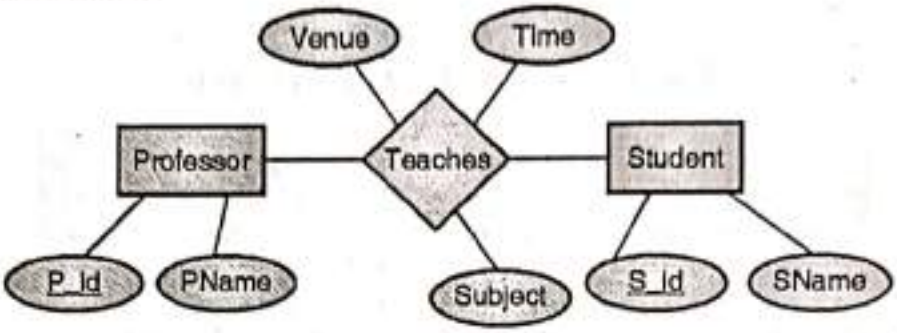


Fig. 4.10.3 : Cross reference approach

Professor table	
Pid	**PName**
1	Om
2	Nitin

Student table	
Sid	**SName**
1	Snehal
2	Tanmay

Teacher relation table				
Pid	**Sid**	**Venue**	**Time**	**Subject**
1	1	SFIT	1 pm	ADBMS
1	2	XIE	1 pm	DBMS

4.11 Mapping Inheritance constraints

Q. Explain how to convert Inheritance in ER to relational Table. (4 Marks)

- **Tables :** Each super class and subclass entity sets represents table in relational model.

- **Columns :** Attributes of entity set is converted to the columns of the tables in relational model.
- **Primary key :** The primary key column of super class is also added to all subclasses and treated as a primary key column for all tables in relational model.

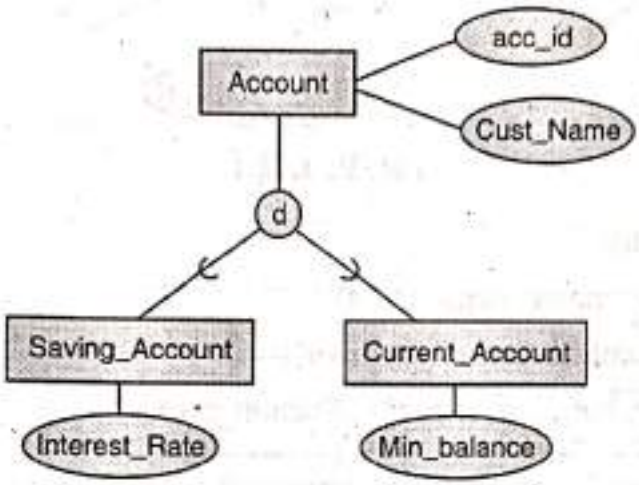


Fig. 4.11.1 : Inheritance relation

Account table	
acc_id	Cust_Name
1	Snehal
2	Tanmay
3	Nikhil

Saving_Account table	
acc_id	Interest_Rate
1	10
2	12
3	15

Current_Account	
acc_id	Min balance
1	1000
2	2000
3	500

4.12 Solved Examples

Example 4.12.1 : Draw an E-R diagram and reduce it to relational database model for a university database for scheduling of classrooms for final exams. This database could be modelled using entities as exam (course_name, section_number, room_number, time);course. (name, department, C_number),room (r_number, capacity, building).Entity section is dependent on course. **(10 Marks)**

Solution :

Step 1 : ER diagram

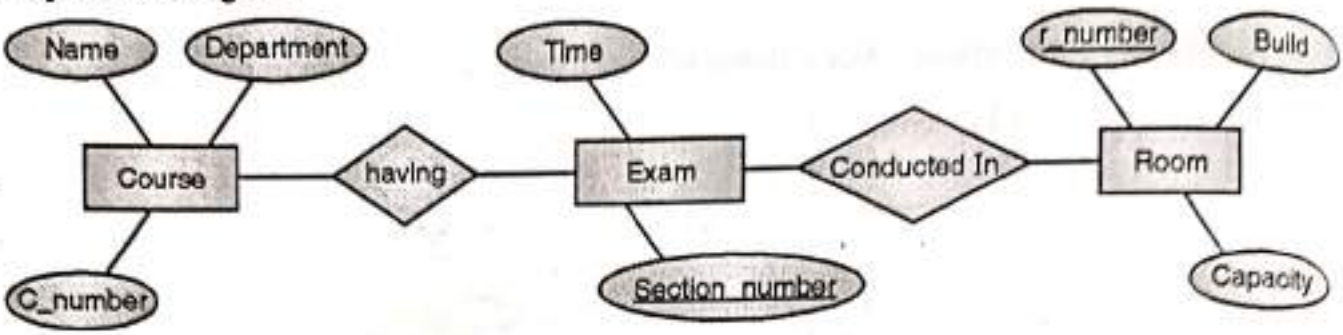


Fig. P. 4.12.1

Step 2 : Mapping to Tables

Course (C_number, name, department)

Exam (Section_number, time, C_number,)

Room (R_number, building, capacity, Section_number)

Example 4.12.2 : Draw an E-R diagram for a university database consisting of 4 entities,

(i) Student (ii) Department (iii) Class (iv) Faculty

and convert it to tables. A student has a unique id, the student can enroll for multiple classes and has at most one major. Faculty must belong to department and faculty can take multiple classes.Every student will get a grade for the class he/she has enrolled. **(10 Marks)**

Solution. :

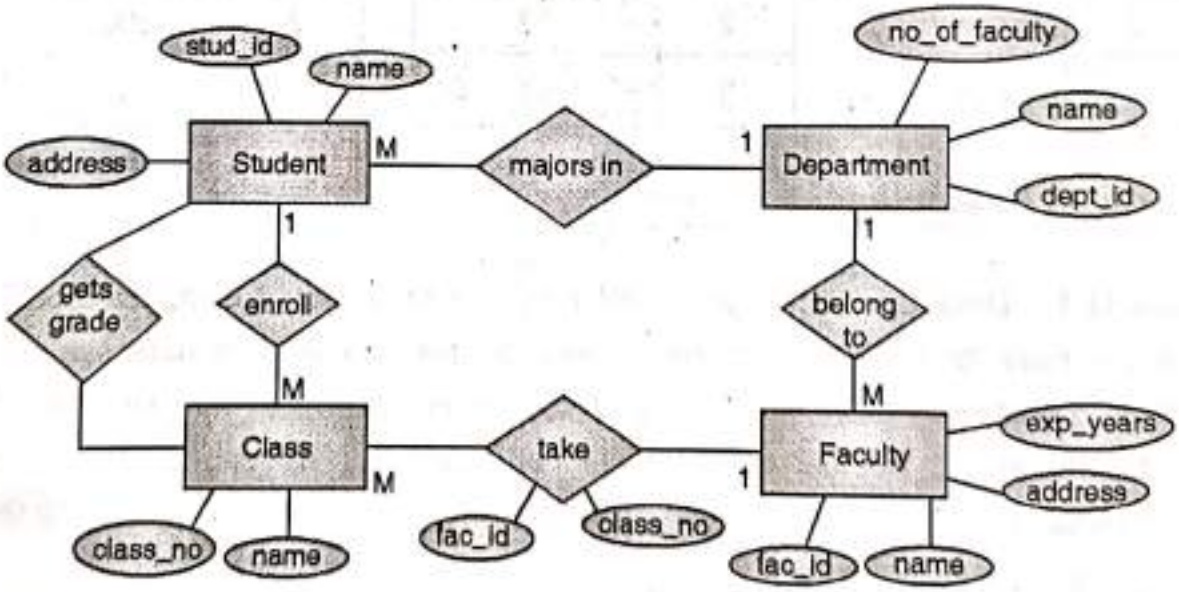


Fig. P. 4.12.2

Student	
Id	Primary Key
Name	
Address	
Dept_id	Foreign key references to dept_id column of Department table
Faculty	
Fac_id	Primary Key
Fac_name	
Exp_years	
Address	
Dept_id	Foreign key references to dept_id column of Department table
Class	
Class_ no	Primary Key
C_name	
Stud_class	
Class_ no	Foreign key references to dept_id column of Department table
Stud_ id	Foreign key references to dept_id column of Department table
take_class	
Fac_id	Foreign key references to fac_id column of Faculty table
Class_ no	Foreign key references to class_ no column of Class table
Department	
Dept_id	Primary Key
D_name	
No_of faculty	
Grade	
Stud_ id	Foreign key references to dept_id column of Department table
Class_ no	Foreign key references to dept_id column of Department table
Grade	

Example 4.12.3 : Construct on E-R diagram for a car-insurance company that has a set of customers each of whom owns one or more cars. Each car has associated with it zero to any number of recorded accidents. **(10 Marks)**

Solution. :

(1) Identify all entities

(a) Insurance company (b) Customer

(c) Car (d) Accidents

(2) Construct ER diagram by merging all above relationships

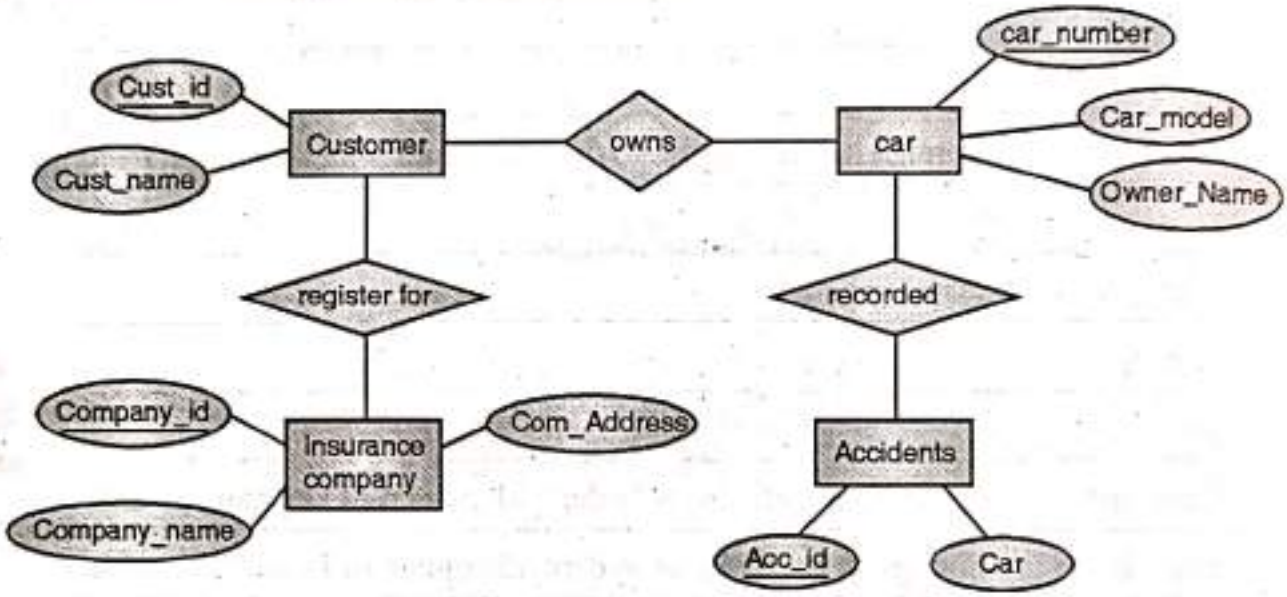


Fig. P. 4.12.3

(1) Mapping Entities

(a) Company (Company_id, Name, Address)

(b) Customer (Customer_id, Name, Address, phone)

(c) Car (Car_Number, Car_Model, Owner)

(d) Accidents (Accident_Id, Location, date, time)

(2) Mapping Relations

(a) Company (Company_id, Name, Address)

(b) Customer (Customer_id, Name, Address, phone, Insurance_Company)

Insurance_Company - refers to customers registered insurance company

(c) Car (Car_Number, Car_Model, Owner_Id)

Owner_Id - refers to customer id owns that car.

(d) Accidents (Accident_Id, Car_Number, Location, date, time)

Car_Number - refers to car involved in accident.

(3) Final Relational Schema

(a) Company (Company_id, Name, Address)

(b) Customer (Customer_id, Name, Address, phone, Insurance_Company)

(c) Car (Car_Number, Car_Model, Owner_Id)

(d) Accidents (Accident_Id, Car_Number, Location, date, time)

Example 4.12.4 : Construct ER diagram and convert into Relational Model for Company Which has several Employees working on different types of projects. Several Employees are working on one department. Every Employee has Manager. Several Employees are supervised by one Employee. **(10 Marks)**

Solution. :

1. Employee (Eid, Ename, mid, sup-id),Pid)
2. Company(Cid, Cname, Location)
3. Project (Pid, Pname, type)

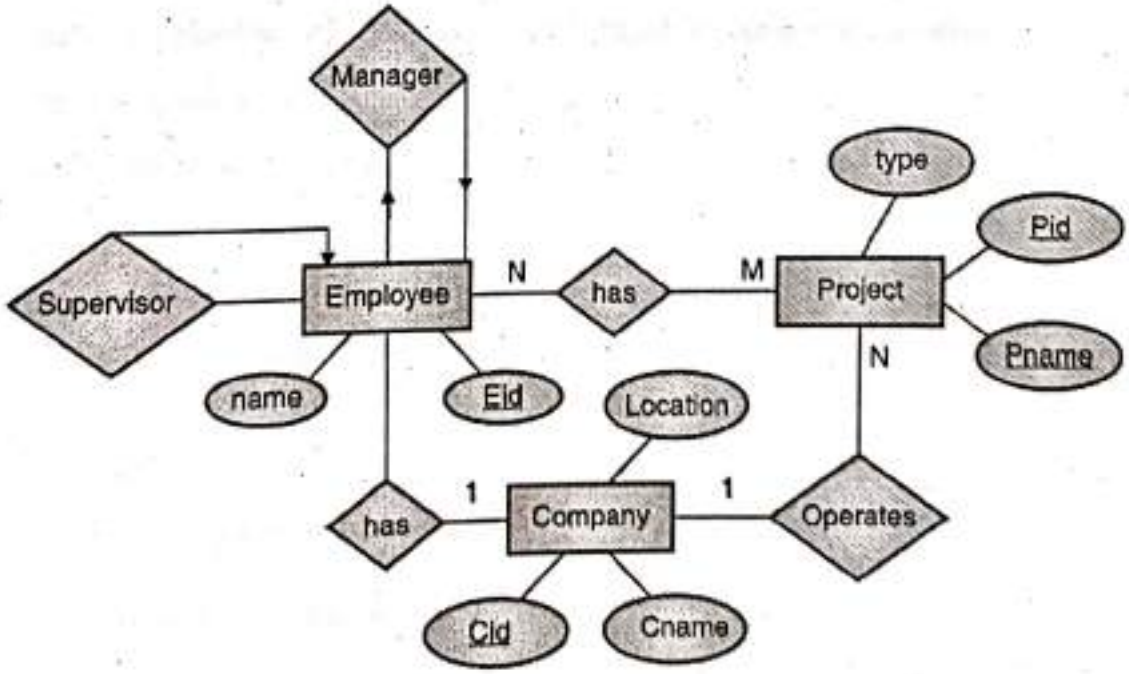


Fig.P. 4.12.4

Review Questions

Q. 1 Explain all types of integrity with help of examples.

Q. 2 Write short note about Delete update rules in Referential Integrity.

Q. 3 Explain the term 'Referential Integrity' and its relation with foreign key.

Q. 4 Write short notes on : Integrity constraints in RDBMS.

Q. 5 Explain terms primary key and foreign key with example.

Q. 6 Explain the steps of an algorithm for ER to relational mapping.

Q. 7 Explain different integrity constraints.

Q. 8 Explain the algorithm to map ER and EER model to relational model in detail.

❑❑❑

CHAPTER 5

Relational Algebra

Module III

Syllabus

Relational Algebra – unary and set operations, Joins, Relational Algebra Queries.

5.1 Relational Algebra

Q. Explain any four relational algebra operations with proper examples. (10 Marks)
Q. Discuss fundamental operations in relational algebra. (10 Marks)

- Relational algebra becomes popular after the publication of E.F. Codd's relational model of data in 1970.
- It is procedural language useful for representing query execution plan and relatively close to SQL.
- Relational algebra is set of operations which accept one relation and produces new relation as a result.
- This query is applied to tables/relations and output is also a table/relation.
- Fundamental operations of Relational Algebra,
  - Unary Relational Operations
    - Project operation (π)
    - Select operation (σ)
    - Rename operation (ρ)
  - SET Theory Operations
    - Union operation ($\cup$)
    - Difference operation (-)
    - Intersection operation ($\cap$)

- Binary Operations
  - Join operations (⋈)
  - Cartesian product operations (X)
  - Division operation (%)

5.2 Selection Operation (σ)

Q. Explain Select relational algebra operation. **MU - Dec. 18, 3 Marks**

(a) Overview

- This operator is used to select some rows from table which satisfy particular selection condition given in selection operation.
- Selection operator selects a set of tuples that satisfy a selection predicate or condition.
- Output of query is exactly same as input schema of table.
- This is unary relational operator having only one input table.

(b) Syntax

$\sigma_{<attribute\_name><comparison\_operator><constant\_value>}$ (Input_Table_Name)

Where,

Attribute_name : Name of column in table

Comparison_operator : =, <, < =, >, > =, < >

(c) Example :

Eid	Ename	Age	Salary
1	Suhas	24	50000
2	Jayendra	24	15000
3	Sachin	25	52000
4	Mahesh	23	41000
5	Satish	34	25000
6	Suma	54	50000
7	Raj	69	45000
8	Anu	74	50000

Query : Select all employees having age below 30 years.

Solution : $\sigma_{age < 30}$ (Employee)

Eid	Ename	Age	Salary
1	Suhas	24	50000
2	Jayendra	24	15000
3	Sachin	25	52000
4	Mahesh	23	41000

(d) Combining multiple conditions

We can have more than one predicate by using logical connectives like AND ($\wedge$), OR ($\vee$).

Query : Select all employees having salary above 5000 and age above 65.

Solution : $\sigma_{salary > 5000 \wedge age > 65}$ (Employee)

Eid	Ename	Age	Salary
7	Raj	69	45000
8	Anu	74	50000

Query : Select all employees with either salary above 50000 or age above 65.

Solution : $\sigma_{salary > 50000 \vee age > 65}$ (Employee)

Eid	Ename	Age	Salary
3	Sachin	25	52000
7	Raj	69	45000
8	Anu	74	50000

5.3 Projection Operation (π)

Q. Describe the following Relational algebra operation : Project (3 Marks)

Q. Explain Project Relational algebra operators with suitable examples.

MU - Dec. 18, 2 Marks

(a) Overview

- This operator is used for selecting some of many columns in table to displayed in result set.
- Projection operator can select a column or set of columns of table to be displayed in output of query.
- We can select only few columns or all columns of a table as per requirements.
- This is unary relational operator having only one input table.

(b) Syntax

$\pi_{<column\_list>}$ (Input_Table_Name)

(c) Example :

Query : Find salary and age of all Employees.

Solution : $\pi_{age, salary}$ (Employee)

Age	Salary
24	50000
24	15000
25	52000
23	41000
34	25000
54	50000
69	45000
74	50000

Query : Find salary of all employees having age less than 25.

Solution : $\pi_{age, salary}$ (Employee)

Age	Salary
24	50000
24	15000
23	41000

5.4 Rename Operation (ρ)

Q. Explain Rename Relational algebra operators with suitable examples. (2 Marks)

(a) Overview

- We can give alternative name to any column (attribute) or any table of query expressions using operator called as RENAME operator.
- This operator is specially introduced to select specific column from joined table (set of two or more tables) containing multiple columns of same column name.
- Rename operator, denoted by the lowercase Greek letter rho (ρ).

(b) Syntax

$\rho_{<New\_Name>}$ (Input_Table_Name)

(c) Example :

Query : Find salary and age of all Employees.

Solution : $\pi_{e.age,\ e.salary}(\rho_e(Employee))$

e.Age	e.Salary
24	50000
24	15000
25	52000
23	41000
34	25000
54	50000
69	45000
74	50000

5.5 SET Operation

Q. Explain various SET operators in relational algebra. (6 Marks)

1. Introduction

- SQL SET operators allow combining results from two or more SELECT statements or combines result set of multiple queries.

- The results of two queries can be combined using the set operations union, intersection and difference.

  Query_1 UNION [ALL] **Query_2**
  Query_1 INTERSECT [ALL] **Query_2**
  Query_1 EXCEPT [ALL] **Query_2**

2. SET Compatibility

- In order to apply SET operations the source tables must possess following Requirements,
- SELECT statement of both queries must retrieve the same number of columns.
- SELECT columns of both queries must be of same Data types.

5.5.1 Union Operator

Q. Explain Set Union Relational algebra operators with suitable examples.
MU - Dec. 18, 3 Marks

(a) Overview

- This operator finds out all combined rows in table 1 and table 2.
- Union effectively appends the result of first query to the result of second query.
- It does not eliminate all duplicate rows and they are printed in result expression.

(b) Syntax

(Query Expression 1) ∪ (Query Expression 2)

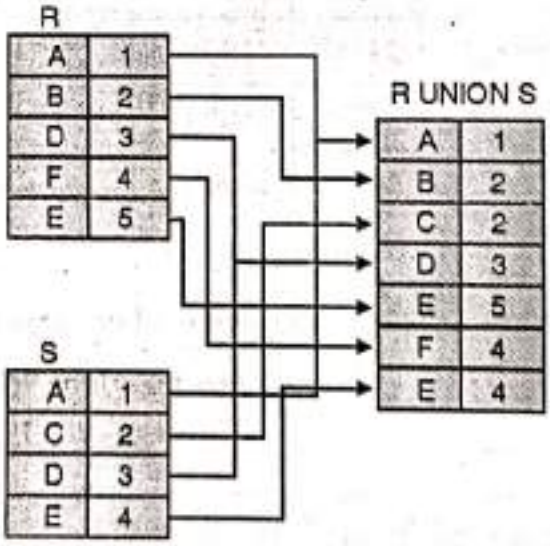


Fig. 5.5.1 : SET Operation

(c) Example :

(i) IT Employee table

Table Name : IT_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23

(ii) Computer department Employee table

Table Name : COMP_Employee		
Eid	**Ename**	**Age**
21	Varsha	24
22	Bhavna	24
23	Geeta	25
24	Amrita	23

Query : Find all Employees in computer and IT departments.

Solution : (IT_Employee) $\cup$ (COMP_Employee)

Eid	Ename	Age
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23
21	Varsha	24
22	Bhavna	24
23	Geeta	25
24	Amruta	23

5.5.2 Intersect Operator

Q. Explain Set Intersection operation with suitable examples. (2 Marks)

(a) Overview

- This operator finds out all rows that are common in table 1 and table 2.
- If Intersect operator is applied on two queries then it will return all rows that are common in the result of Query 1 and Query 2.

(b) Syntax

(Query Expression 1) ∩ (Query Expression 2)

R	
A	1
B	2
D	3
F	4
E	5

S	
A	1
C	2
D	3
E	4

R INTERSECTION S	
A	1
D	3

c) Example :

(i) All employees in IT department.

Table Name : IT_Employee		
Eid	Ename	Age
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23

(ii) All employees in Vidya Engineering College.

Table Name : Vidya_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
23	Geeta	25
24	Amruta	23
35	Sangita	21

Query : Find all Employees in IT department of Vidya Engineering College.

Solution : (IT_Employee) ∪ (Vidya_Employee)

Eid	Ename	Age
11	Suhas	24
12	Jayendra	24

5.5.3 Difference Operator

Q. Explain Set Difference Relational algebra operators with suitable examples. (2 Marks)

(a) Overview

- This operator finds out all rows that are present in table 1 and not in table 2.
- If Intersect operator is applied on two queries then it will return all rows that are present in the result of Query 1 and not in Query 2.

(b) Syntax

(Query Expression 1) - (Query Expression 2)

R	
A	1
B	2
D	3
F	4
E	5

R DIFFERENCE S	
B	2
F	4
E	5

S	
A	1
C	2
D	3
E	4

S DIFFERENCE R	
C	2
E	4

(c) Example :

(i) All faculties in IT department of Vidya Engineering College.

Table Name : Vidya_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23

(ii) All faculties in IT department of all colleges.

Table Name : IT_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23
23	Geeta	25
24	Amruta	23
35	Sangita	21

Query : Find all Employees in IT department but not in Vidya Engineering College.

Solution : (IT_Employee) - (Vidya_Employee)

Eid	Ename	Age
23	Geeta	25
24	Amruta	23
35	Sangita	21

5.6 Cross Product / Cartesian product

Q. Explain Cartesian Product Relational algebra operation. **MU - Dec. 18, 2 Marks**

(a) Overview

- A cross join performs relational product or Cartesian product of two tables specified in query.
- In this case every row in first table will be joined with every row in second table. So finally number of rows in result table will be equals to product of number of rows in table 1 and number of rows in table 2.
- That means all rows in the first table are joined to all rows in the second table.

(b) Syntax

(Query Expression 1) × (Query Expression 2)

(c) Example :

Employee		
Eid	**Ename**	**Did**
1	Mahesh	100
2	Suhas	200
3	Jayendra	100

Department	
Did	**Dname**
100	HR
200	TIS

Query : Find combination all Employees and departments.

Solution : (Employee) X (Department)

Eid	Ename	Did	Did	Dname
1	Mahesh	100	100	HR
1	Mahesh	100	200	TIS
2	Suhas	200	100	HR
2	Suhas	200	200	TIS
3	Jayendra	100	100	HR
3	Jayendra	100	200	TIS

Cartesian product

- In DBMS, CROSS join occur due to WHERE condition is missing in query or some invalid operations in where clause leads to undesired results or CROSS Join.
- A Cartesian product is formed when :

  A join condition is omitted.

  A join condition is invalid.

  To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

5.7 Join Operation ($\bowtie_\theta$)

Q.	Express Join in terms of basic relational algebra operations.	(6 Marks)
Q.	Explain Natural Join Relational algebra operators with suitable examples.	(2 Marks)
Q.	Explain Inner Join Relational algebra operators with suitable examples.	(2 Marks)
Q.	Explain Outer Join Relational algebra operators with suitable examples.	(2 Marks)

(a) Overview

- Join operator helps us to retrieve data from multiple tables or relations.
- Most common type of join is **Natural join ($\bowtie$)** in which column having same name in two table will be taken for joining tables.

(b) Syntax

$$(<\text{table\_name}>) \bowtie_{<\text{join\_condition}>} (\text{table\_name})$$

(c) Types of Joins

There are various types of joins possible in relational algebra.

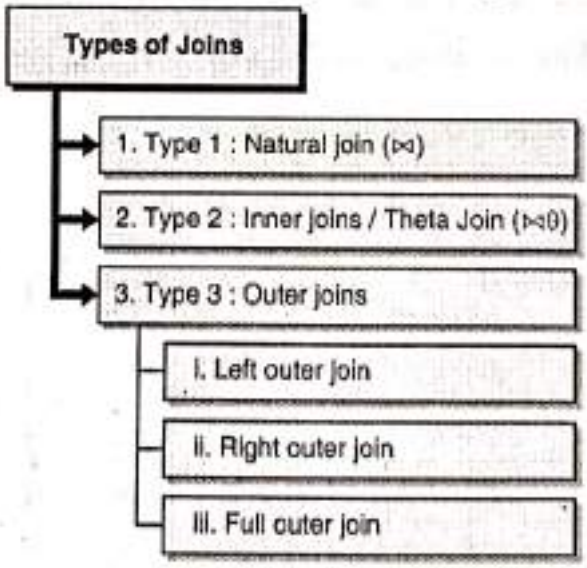


Fig. 5.7.1 : Types of Joins

Type 1 : Natural join (⋈)

- Natural join can join tables based on the common columns in the tables being joined.
- A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.

Prerequisites for Natural Join

Both table must have at list one common column with same column name and same data type.

Steps of Working

- The two table are joined using Cross join
- DBMS will look for a common column with same name and data type
- Tuples having exactly same values in common columns are kept in result.

Example :

Employee		
Eid	**Ename**	**Did**
1	Amit	10
2	Nitin	30
3	Yogesh	50

Department	
Did	**Dname**
10	IT
30	HR
40	TIS

Query : Find all Employees and their respective departments.

Solution : (Employee) ⋈ (Department)

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
Employee Data			Department Data	

- In above example the employee data having same did as department data are kept in result set.
- All non-matching tuples of cross join are ignored.

Type 2 : Inner joins / Theta Join ($\bowtie_\theta$)

- Theta join will combines tuples from multiple relations if they satisfy the specified join condition.
- This join condition is also called as Theta and denoted by the symbol θ.
- The tables are joined according to join conditions.

- The only rows with matching v alues are combined using inner join.
- Inner join will ignore all tuple does not find matching tuple in other table.

Example :

Query : Find all Employees and their respective departments.

Solution : Employee $\bowtie_{employee.did = Department.did}$ Department

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
Employee Data			Department Data	

- In above example the employee data having did exactly same as department data are kept in result set.
- All non-matching tuples of cross join are ignored.

Type 3 : Outer joins

- In an inner join or in case of a simple join, the resultant table contains only the combinations of rows that satisfy the join conditions.
- Rows that do not satisfy the join conditions are discarded. Outer join, joins two table although there is no match between two joining tables.
- Outer joins are useful when you are trying to determine which values in related tables cause referential integrity problem.
- Such problems are created when foreign key values do not match the primary key values in related table.

(i) Left outer join

- Table on left side of operator may contain null values.
- Left outer join takes all tuples in the left relation that did not match with any tuple in the right relation.

Example :

Query : Find all Employees and their respective department data.

Solution : Employee $\bowtie=_{employee.did = Department.did}$ Department

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
3	Yogesh	50	Null	Null
Employee Data			Department Data	

- In above example the employee data having did exactly same as department data are kept in result set.
- All left side non-matching tuples of cross join are also considered.

(ii) Right outer join

- Table on right side of operator may contain null values.
- Right outer join takes all tuples in the right relation that did not match with any tuple in the left relation.

Example :

Query : Find all departments with employee data.

Solution : Employee $=\bowtie_{employee.did = Department.did}$ Department

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
Null	Null	Null	40	TIS
Employee Data			Department Data	

- In above example the employee data having did exactly same as department data are kept in result set.
- All right side non-matching tuples of cross join are also considered.

(iii) Full outer join

Any table on both sides of operator may contain null values.

Example :

Query : Find all Employees and departments.

Solution : Employee $=\bowtie=_{employee.did = Department.did}$ Department

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
3	Yogesh	50	Null	Null
Null	Null	Null	40	TIS
Employee Data			Department Data	

- In above example the employee data having did exactly same as department data are kept in result set.
- All right side non-matching tuples of cross join are also considered.

5.8 Relational Division Operator

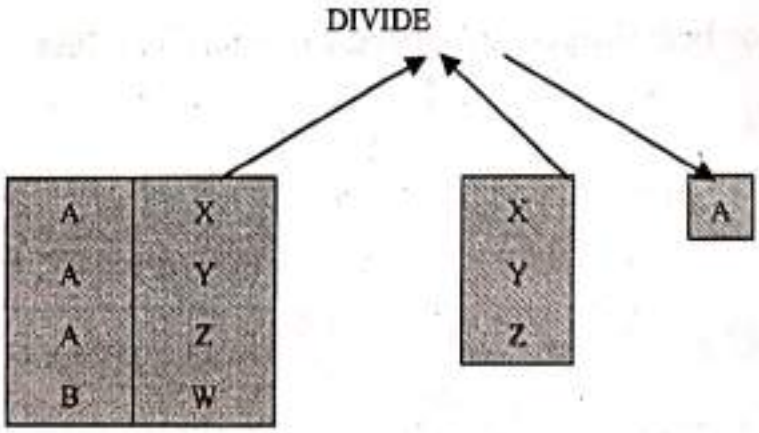
Q. Explain Division Relational algebra operators with suitable examples. (2 Marks)

(a) Overview

- The divide operator operates on two tables that must have common columns between them.
- The relational divide operator (so called to distinguish it from mathematical division) returns the records in one record set that have values that match all the corresponding values in the second record set.
- The output of divide operation is one column in which values of common column in both tables match.

(b) Syntax

(Query Expression 1) (Query Expression 2)



DIVIDE Operator

(c) **Example :**

Student Table			
Stud_ID	**Sname**	**Course_ID**	**Gender**
1	Mahesh	100	M
2	Manish	100	M
3	Amruta	100	F
3	Amruta	200	F
6	Neesha	100	F
3	Amruta	300	F
6	Neesha	300	F
6	Neesha	200	F

Course Table contains all courses that trainer 401 is taking.

Course_ID	Trainer_ID
100	401
200	401
300	401

Find female students take ALL the courses that 401 are taking.

Student ÷ Course

OR

$\pi_{s.Stud\_ID,\ s.Sname,\ s.Gender,\ s.Course\_ID}\ (\rho_s(Student) \div \rho_c(Course))$

s.Stud_ID	s.Sname	s.Gender	s.Course_ID
3	Amruta	F	200
6	Neesha	F	100

5.9 Operator Precedence

The normal way to group operators is according to its precedence of operation.

Precedence	Operators
1.	Unary operators σ, π and ρ have highest precedence.
2.	Next highest are the "multiplicative" like operators ⋈ and X.
3.	Lowest are the "additive" operators ∪(Union), ∩(Intersection) and – (minus) operators.

But there is no universal agreement, so we always put parentheses around the argument of a unary operator and it is a good idea to group all binary operators with parentheses enclosing their arguments.

5.10 Relational Algebra Queries - Solved Examples

Example 5.10.1 : Consider the following relations for database that keeps track of student enrollment in courses and books issued for each course.

STUDENT (Ssn, Name, Subject, DOB)
COURSE (Course_id, Name, Dept)
ENROLL (Ssn, Course_id, Semester, Grade)
Book_issued (Course_id, Semester, ISBN)
TEXT (ISBN, Title, Publisher, Author)

Write any 5 Queries in relational algebra. **(10 Marks)**

Solution :

(1) Write a query to select all courses available in institute.

$\Pi_{course\_id, CName, Dept}$ (COURSE)

(2) Find all student details registered for course id 10.

$\Pi_{Ssn, Name}$ ($\sigma_{course\_id = 10}$(ENROLL ⋈ STUDENT))

(3) Find various book titles and authors for semester higher than 3.

$\Pi_{ISBN, Title, Author}$ ($\sigma_{semester > 3}$(Book_Issed ⋈ TEXT))

(4) Find all students belonging to IT Department.

(a) To find course_id of 'IT' Department

$T_1 \leftarrow \Pi_{course\_id,}$ ($\sigma_{Dept = 'IT'}$ (COURSE))

(b) To find all students enrolled for above course id.

$T_2 \leftarrow \Pi_{ssn}, (ENROLL \bowtie T_1)$

(c) To find student details having above Ssn.

$Ans \leftarrow \Pi_{Ssn, Name, DOB} (STUDENT \bowtie T_2)$

Ex. 5.10.2 : Consider the relations given below :

Dealer (Dealer-no, DealerName, address)

Part (Part-no, Part-name, color)

Assigned-to (Dealer-no, Part-no, cost)

Give an expression in relational algebra the following queries :

(i) Find the name of all dealers who supply 'Red' Parts.

(ii) Find the name of the dealers who supply both Yellow and Green Parts.

(iii) Find the name of the dealers who supply all the Parts.

(iv) List all dealer names. **(10 Marks)**

Solution :

(i) The name of all dealers who supply 'Red' Parts.

$\Pi_{Dealername} (\sigma_{course='red'} (Dealer \bowtie Part))$

(ii) The name of the dealers who supply both Yellow and Green Parts

$\Pi_{Dealername} (\sigma_{course='red' \text{ OR } course='yellow'} (Dealer \bowtie Part))$

(iii) The name of the dealers who supply all the Parts.

$\Pi_{Dealername} (Dealer \bowtie Part))$

(iv) The list of all dealer names

$\Pi_{Dealername} (Dealer)$

Review Questions

Q. 1 Explain various operators in relational algebra.

Q. 2 Write a short note :

(a) Selection operation

(b) Projection Operator

Q. 3 Explain concept of division operation.

Q. 4 Explain concept of JOIN operation in relational algebra.

Q. 5 Explain various SET operators in relational algebra.

Q. 6 Explain concept of product operation in relational algebra.

Q. 7 Explain Outer join relational algebra operators with example :

Q. 8 Explain following relational algebra operators with example :

(i) Natural Join

(ii) Set Difference.

Q. 9 Explain following relational algebra operators with example :

(i) Rename

(ii) Set Intersection

(iii) Division

(iv) Left outer join

(v) Union

Q. 10 Explain any four relational algebra operations with proper examples.

Q. 11 Discuss fundamental operations in relational algebra.

❑❑❑

CHAPTER 6

Structured Query Language

Module IV

Syllabus

Overview of SQL, Data Definition Commands, Data Manipulation commands, Data Control commands, Transaction Control commands. Integrity constraints : key constraints, Domain Constraints, Referential Integrity, check constraints.

6.1 Overview of SQL

- **SQL (Structured Query Language)** is a computer language aimed to store, manipulate, and retrieve data stored in relational databases.
- It was developed by IBM Research in the mid 70's and standardized by ANSI in 1998.
- The first commercial relational database was released by Relational Software (later called as Oracle).
- SQL is a keyword-based language and each statement begins with a unique keyword.
- SQL syntax is not case sensitive.

6.1.1 Role of SQL

Q. Explain role of SQL with example. (3 Marks)

- SQL is an interactive query language which can be used to retrieve data from database.
- SQL is a database programming language which can be used along with programming language to access data from database.
- SQL is a database administration language which can be used to monitor and control data access by various users.
- SQL can be used as an Internet data access language.

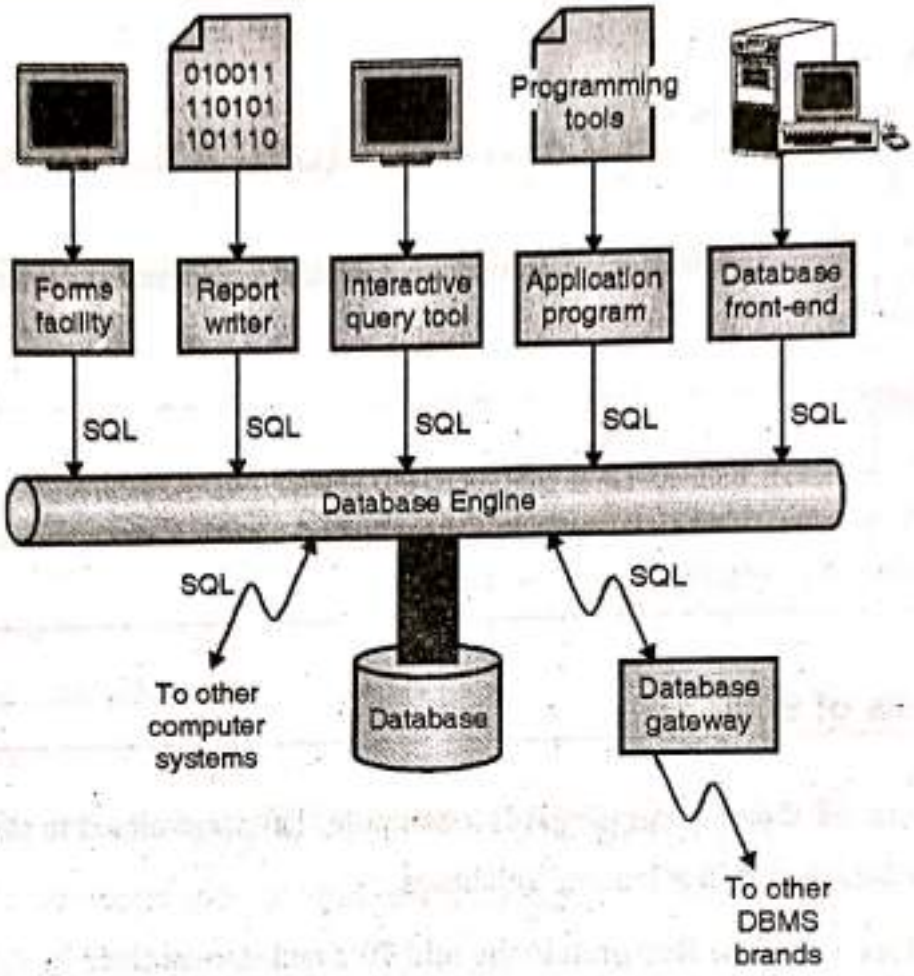


Fig. 6.1.1 : Role of SQL in DBMS

6.2 SQL Data Types

Q. Explain various SQL data types with example. (4 Marks)

- The basic data types available with SQL standard are as enlisted below, all data types may not be supported by SQL server or Oracle.
- Data types include numeric, character string, bit string, Boolean and date time.

1. Numeric data types

This datatype is used to store a number values that can be decimal or floating point values.

(a) Integer number of various size

These types of system are used to store natural numbers which are not having any decimal values.

Example :

111, 23 etc.

As per size of number we can use following types of integers.

(i) INTEGER (p) (ii) INTEGER or INT

(iii) SMALLINT (iv) BIGINT

(b) Floating point numbers of various precision

This system is used for storing decimal numbers which may be of greater size than integers.

Example :

11.2, 12.3 etc.

As per size of floating point number we can use following types of numbers.

(i) FLOAT or REAL

(ii) DOUBLE PRECISION

(c) Formatted numbers

This system used for storing some special numbers which may be of greater size than integers and floating-point numbers.

Example :

1.12342 (Numeric(1,5)),12.234 (Numeric(2,3)) etc.

(i) DECIMAL or DEC (i, j)

(ii) NUMERIC (i, j)

Where i = Precision = Total number of digits in number

j = Scale = Total number of digits after decimal point. (default value is 0)

2. Character string data type

- This data type is used to store a character string which is combination of some alphabets and enclosed in single quotation marks.
- Example: 'Mahesh','abc' etc.

(a) Fixed length : CHAR (n), Where n = number of characters

Example :

If 'abc' is stored in char (10) will be stored as 'abc'.

(abc padded with 7 blank spaces)

(b) Varying length : VARCHAR (n) Where n = maximum number of characters

Example :

If 'abc' is stored in VARCHAR (10) will be stored as 'abc' (no blank spaces)

3. Date time data type

(a) Date

- The DATE data type has ten positions, and its components are YEAR, MONTH and DAY in form YYYY-MM-DD.
- The length is 10.
- Generally not supported by SQL server.(Supported by DB2)

Example :

Date '2009-01-01' (as 'YYYY-MM-DD')

(b) Time

- The TIME data type has at least eight positions, and its components are HOUR, MINUTES and SECOND in form HH:MM:SS [.*s*F] where F is the fractional part of the SECOND value.
- Generally not supported by SQL server.
- If a second's precision is not specified, *s* defaults to 0. The length is 8 (or 9 + *s*, if *s* > 0).

Example :

Time '11:16:59' (as 'HH:MM:SS')

(c) Timestamp / date time

- The TIMESTAMP data type includes both date and time fields, plus a minimum of six positions and for decimal fractions of second and optimal with TIMEZONE Qualifier.
- Represented using the fields YEAR, MONTH, DAY, HOUR, MINUTE and SECOND in the format YYYY-MM-DD HH:MM:SS[.*s*F] where F is the fractional part of the SECOND value.
- If a second precision is not specified, *s* defaults to 6. The length is 26 (or 19, if s = 0 or 20 + *s*, if *s* > 0).

Example :

Timestamp '2009:01:01 11:16:59 648302'

(as 'YYYY-MM-DD HH:MM:SS TIMEZONE')

CurrentTimeStamp : Local date and time without time zone

(d) Interval

This specifies an interval a relative value that can be used to increment or decrement an absolute value of date, time or timestamp.

INTERVAL YEAR TO MONTH Datatype

Example :

'21-5' Year(2) To Month indicates an interval of 21 years and 5 months

'21-5' Year(2) indicates an interval of 21 Years.

'5' Month(2) indicates an interval of 5 month.

INTERVAL DAY TO SECOND Datatype

Example :

'5 03:15:20' Day(2) To Second indicates an interval of 5 days,3 hours,15 minutes and 20 seconds.

Generally not supported by SQL server but supported by Oracle.

6.3 Data Definition Language (DDL)

Q.	Explain DDL commands with example.	(10 Marks)
Q.	Explain Database Languages	MU - Dec. 18, 2 Marks

- To create database schema and database objects like table **Data Definition Language (DDL)** can be used.
- DDL statements are used to build and modify the structure of your tables and other objects in the database.
- The set of DDL commands are as below,
  1. CREATE Statement : To create Database objects
  2. ALTER Statement : To modify structure of database objects
  3. DROP Statement : To remove database objects
  4. RENAME Statement : To Rename Database objects
  5. TRUNCATE Statement : To empty the database table
- When you execute a DDL statement, it takes effect immediately, as it is **Autocommited** into database. Hence no rollback operation (Undo) can be performed with these set of commands.
- Database objects are any data structure created in database.

  Example : Table, View, Sequence etc.

6.4 CREATE Statement / CREATE Table

Q. Explain CREATE command with example. (4 Marks)

- CREATE statement is used to create new database objects like table, index and others.
- CREATE TABLE is the command in database system is used to create a new table with unique name or identifier.
- This statement used to create database object.

Syntax

```
CREATE TABLE <Table_Name>
    ( Column_1 datatype,
    Column_2 datatype,
    ....
    Column_n datatype
    );
```

Example :

```
SQL> CREATE TABLE Employee
    (   Eid INT,
    Name    VARCHAR (20),
    Age     INT,
    Address CHAR (25),
    Salary  DECIMAL (18, 2)
    );
Query OK, 0 rows affected (0.01 sec)
```

To view the structure of newly created table.

```
SQL> DESC Employee;
+---------+------------------+------+------+---------+------+
| Field   | Type             | Null | Key  | Default | Extra |
+---------+------------------+------+------+---------+------+
| EID     | int(10)          | YES  |      | NULL    |      |
| NAME    | varchar(20)      | YES  |      | NULL    |      |
| AGE     | int(11)          | YES  |      | NULL    |      |
| ADDRESS | char(25)         | YES  |      | NULL    |      |
| SALARY  | decimal(18,2)    | YES  |      | NULL    |      |
+---------+------------------+------+------+---------+------+
5 rows in set (0.00 sec)
```

6.5 Create Table with Integrity Constraints

6.5.1 Domain Integrity Constraint

Domain constraints are used to test the values inserted into the table is correct or not.

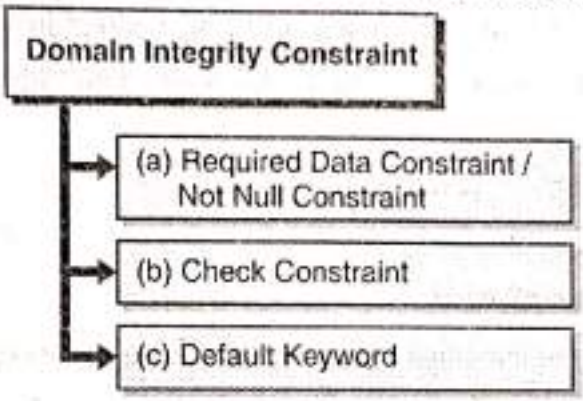


Fig. 6.5.1 : Domain integrity constraint

(a) Required Data Constraint / Not Null Constraint

Some attributes (columns) in a database are not allowed to contain NULL value. NULL values are values which are unknown, unassigned or missing attribute values.

Example :

In the student database, every student must have an associated student name. Student_name should not be NULL.

```
SQL> CREATE TABLE Student
      ( Eid     INT NOT NULL);
Query OK, 0 rows affected (0.01 sec)
```

(b) Check Constraint

Use of **check constraint** is to ensure that attribute value satisfies specific user defined condition.

Example :

Table with customer entity having name, cid and gender which can be M or F. Hence, attribute gender can take only two values either 'M' or 'F'.

```
SQL> CREATE TABLE customer
      (    Name    CHAR (25) NOT NULL,
           Gender  CHAR (1),
           CHECK (Gender IN ('M', 'F'))
      );
Query OK, 0 rows affected (0.01 sec)
```

(c) Default Keyword

Default keyword is used to add some value if no attribute value added for tuple.

Example :

Table with customer entity having name, cid and gender in which cid is primary key. If name is not added for customer that will be taken as 'Unknown'.

```
SQL> Create table customer
     ( Name char (25) DEFAULT 'UNKNOWN');
Query OK, 0 rows affected (0.01 sec)
```

6.5.2 Entity Integrity Constraint

Entity constraints are used to test the values inserted into the database are correct or not with respect to other tuples in same table.

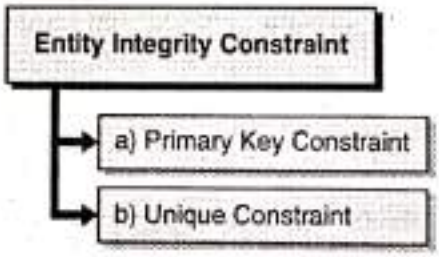


Fig. 6.5.2 : Entity integrity constraint

(a) Primary Key Constraint

- Primary key attribute is same as unique key constraint with Not NULL constraints.
- Primary key attribute values needs to be unique as well as null values are not allowed in primary key attributes.
- The main difference in unique constraint and primary key constraint is that one null value is allowed in unique constraint which can be treated as unique value while nulls are not allowed in primary key constraint.

Example :

Table with customer entity having name, cid and gender in which cid is primary key.

```
SQL> CREATE TABLE customer
     ( Name CHAR (25),
     Cid CHAR (10) PRIMARY KEY
     );
Query OK, 0 rows affected (0.01 sec)
```

(b) Unique Constraint

- In case of unique constraint no two tuples can have equal value for same attributes.

- This constraint says that attributes forms candidates key, which allows one Null value which is unique by itself.
- This UNIQUE constraint can be applicable to user defined domain declaration also.

Example :

```
SQL> CREATE TABLE customer
     ( Name  CHAR (25),
     Cid CHAR (10),
     Email    CHAR (50) UNIQUE
     );
Query OK, 0 rows affected (0.01 sec)
```

6.5.3 Referential Integrity Constraint in SQL

- A value appearing in a one relation (table) for a given set of attributes also appears for another set of attributes in another relation (table).
- This is called referential integrity.
- The referential integrity constraint is defined between two tables to maintain the consistency among tuples in the two relations.

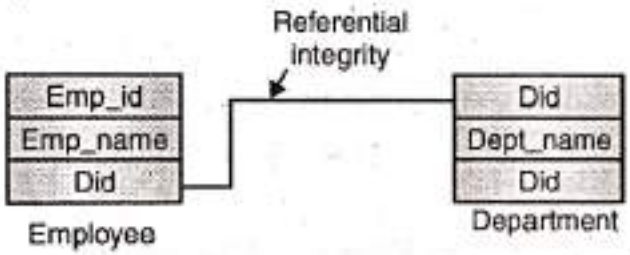


Fig. 6.5.3

Example :

```
SQL> Create table Department
     ( Did     INT,
     Dept_name VARCHAR (100) NOT NULL,
     PRIMARY KEY (did)
     );
Query OK, 0 rows affected (0.01 sec)
SQL> Create table Emp
     ( Emp_id     INT,
Emp_name VARCHAR (100) NOT NULL,
Did   INT REFERENCES department (did)
);
Query OK, 0 rows affected (0.01 sec)
```

6.6 Alter Table

Q. Explain ALTER command with example. **(4 Marks)**

- Once database object is created in database, we may require ALTER command to update structure of database object.
- The ALTER TABLE statement can be used to add, delete, or modify columns in an existing table.
- The ALTER TABLE command can also be used to add and drop various constraints on an existing table.

Syntax

```
ALTER TABLE <Table_Name>
 ADD Column_1 datatype;

ALTER TABLE <Table_Name>
 Modify Column_1 New_datatype;

ALTER TABLE <Table_Name>
 DROP Column_1;
```

Example :

```
SQL> ALTER TABLE Employee
       ADD Address VARCHAR (100);
Query OK, 0 rows affected (0.01 sec)
```

To view the changed structure of table

```
SQL> DESCRIBE TABLE Employee;
+---------+-----------------+-------+------+----------+--------+
| Field   | Type            | Null  | Key  | Default  | Extra  |
+---------+-----------------+-------+------+----------+--------+
| EID     | int(10)         | YES   |      | NULL     |        |
| NAME    | varchar(20)     | YES   |      | NULL     |        |
| AGE     | int(11)         | YES   |      | NULL     |        |
| ADDRESS | varchar(100)    | YES   |      | NULL     |        |
| SALARY  | decimal(18,2)   | YES   |      | NULL     |        |
+---------+-----------------+-------+------+----------+--------+
5 rows in set (0.00 sec)
```

6.7 Rename Table

Q. Explain RENAME command with example. (4 Marks)

- It is possible to change name of table with or without data in it using simple RENAME command.
- We can rename any table object at any point of time.

Syntax

```
RENAME TABLE <Table_Name> To <New_Table_Name>;
```

Example :

```
SQL> RENAME TABLE Employee To EMP;
```

6.8 Truncate Table

Q. Explain TRUNCATE command with example. (4 Marks)

- The TRUNCATE TABLE command is used to delete all data from an existing table.
- It is possible to do same action with DROP TABLE command but it would remove complete table structure from the database.
- A DELETE command will also remove all data from table but with DELETE data deletion can be rolled back and truncate acts as permanent data deletion with no roll back possible.
- If any delete triggers are defined on the table, then the triggers are not fired on truncate table.
- Truncate will de-allocates memory space. So that the free space can be used by other tables unlike DELETE command.

Syntax

```
TRUNCATE TABLE <Table_Name>;
```

Example :

```
SQL> TRUNCATE TABLE EMP;
```

6.9 Drop Command / DROP Table

Q. Explain DROP command with example. (4 Marks)

- Drop command can be used to remove database any objects from user database.

- The SQL DROP TABLE statement is used to remove a table definition and all related data like indexes, triggers, constraints and permission specifications for that table.
- The developer must be careful while running this command because once a table is dropped then all the information available in that table will also be lost forever and no roll back can be done.

Syntax

```
DROP TABLE <Table_Name>;
```

Example :

- If we want to permanently remove the Employee table that we created, we'd use the following command

```
SQL> DROP TABLE Employee;
Query OK, 0 rows affected (0.01 sec)
```

- If we want to permanently remove the Employee table that we created, we'd use the following command

```
SQL> DESC TABLE Employee;
ERROR 1146 (42S02): Table 'TEST.Employee' doesn't exist
```

6.10 Data Manipulation Language (DML)

Q. Explain DML commands with syntax. (Refer section 6.10) (10 Marks)
Q. Explain Database Languages MU - Dec. 18, 2 Marks

- **Data Manipulation Language (DML)** statements are used for manipulating or managing data in database.
- DML commands are not auto-committed like DDL statements.
- It means changes done by DML command can be rolled back. Or in other words the DML statements do not implicitly commit the current transaction.
- DML is set of commands used to,

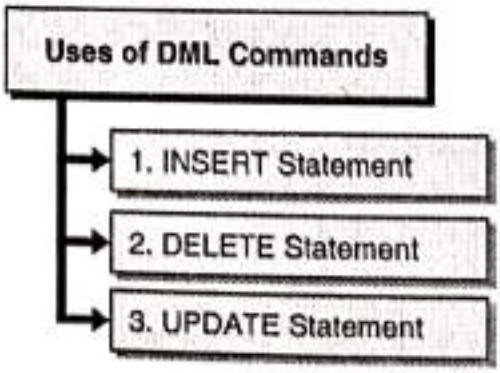


Fig. 6.10.1 : Uses of DML commands

6.10.1 INSERT Statement

- Insert statement used to add records to the existing table.
- To insert data into a table, SQL INSERT INTO command can be used.
- To insert few values in table as per columns names we can use generic syntax as below,

```
INSERT INTO <Table_Name> (Column1, . . . , ColumnN)
VALUES (column1. . . , columnN);
```

- If all values for all the columns of the table are to be added then also no need to specify the column names in the SQL query.
- But, we need to make sure the order of the values is in the same order as the columns in the table.

```
INSERT INTO <Table_Name>
VALUES (column1. . . , columnN);
```

Example :

```
SQL> INSERT INTO Employee VALUES (1001, 'Mahesh');
SQL> INSERT INTO Employee VALUES (NULL, 'Jayendra');

SQL> INSERT INTO Employee (Name, Eid) VALUES ('Sachin' , 1002);
SQL> INSERT INTO Employee (Name, Eid) VALUES ('Suhas' , NULL);
```

The data added in table can be displayed as follows,

```
SQL> SELECT * FROM Employee;
+----+----------+
|EID | NAME     |
+----+----------+
|1001| Ramesh   |
|1002| Khilan   |
|NULL| Kaushik  |
|NULL| Chaitali |
+----+----------+
```

6.10.2 DELETE Statement

- Delete statement is used to delete some or all records from the existing table.
- To delete data into a table, SQL DELETE command can be used.
- To delete all rows in table we can use generic syntax as below,

Syntax

```
DELETE
FROM <Table_Name>;
```

To delete selected rows from table we can specify the WHERE condition.

```
DELETE
FROM <Table_Name>
WHERE <Condition>;
```

Example :

```
DELETE
FROM    Employee
WHERE  Eid IS NULL;
```

To check rows deleted by above delete query is as given.

```
SELECT *
FROM    Employee;
SQL> SELECT * FROM Employee;
+----+---------+
|EID | NAME    |
+----+---------+
|1001| Ramesh  |
|1002| Khilan  |
+---+---------+
```

6.10.3 UPDATE Statement

- The UPDATE statement is used to modify the existing data present in a table.
- To update data in a table, SQL UPDATE command can be used.
- To update all rows in table we can use generic syntax as below,

```
UPDATE <Table_Name>
SET     column1 = new_value;
```

To update selected rows from table we can specify the WHERE condition in Update statement.

```
UPDATE <Table_Name>
SET     column1=new_value
WHERE condition;
```

Example :

```
SQL>    UPDATE    Employee
        SET       Eid=1002
        WHERE          name = 'Suhas';
```

To check rows updated in table using select query,

```
SQL>  SELECT  *
      FROM  Employee;
```

```
SQL> SELECT * FROM Employee;
+----+----------+
|EID | NAME     |
+----+----------+
|1001| Ramesh   |
|1002| Suhas    |
+----+----------+
```

6.11 Data Control Language (DCL)

Q. Write a note on DCL. (4 Marks)
Q. Explain Database Languages MU - Dec. 18, 1 Mark

- **Data Control Language (DCL)** is used to control various user actions (or privileges) in Database.
- To perform any operation in the database user needs **privileges** like creating tables, sequences or views.
- DCL is set of commands used to,
  - **Grant :** Gives some privilege to user for performing task on database.
  - **Revoke :** Take back permissions given from user.
- Privileges can be of many types,
  - **System Privileges :** creating a table is types of system privilege.
  - **Object Privileges :** To execute query on tables object privilege can be used.
  - **Ownership Privileges :** To execute query on tables created by same user.

6.12 Privileges

Q. Explain all types of privileges. (6 Marks)
Q. Enlist the privileges with suitable example (6 Marks)
Q. Write a short note on :
1. Ownership Privileges 2. Object Privileges 3. System Privileges (6 Marks)

Introduction

- The set of actions that a user can perform on a database object are called the privileges.
- Privilege is right to execute particular SQL statement on database.
- The high level user (Like DBA) has power to grant access to database and its object.

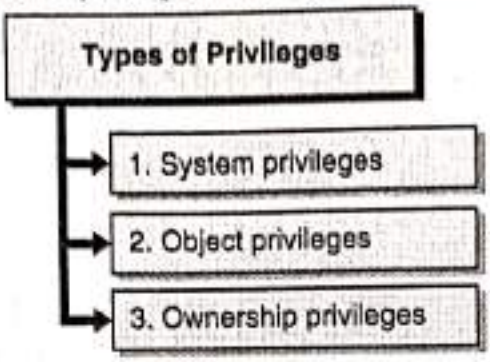


Fig. 6.12.1 : Types of Privileges

1. System privileges

- System privileges are rights and restriction that are implemented on databases to control which users can access how much data in the database.
- User requires system privileges to gain access to database.
- System privileges are generally provided by DBA.
- Few system privileges are as below,

System privileges	Authorized to
CREATE USER	Create number of users in DBMS
DROP USER	Drop any other users in DBMS
CREATE ANY TABLE	Create table object in any schema.
SELECT ANY TABLE	Query table object or view in any schema.
DROP ANY TABLE	Drop table object in any schema.

2. Object privileges

- Object privileges are rights and restrictions to change contents of database objects.
- User requires object privileges to manipulate the content of object within database.
- Once we have created object in a database, after some time there may be few changes needs to be introduced in object.
- Not all database users are allowed to make such changes in database; hence administrator should have control over all objects modification.
- The user which has GRANT ANY PRIVILEGE system privilege granted to him then he

can act like administrator to control database modifications.

- Different objects has different privileges assigned for him.
- Few object privileges are as below,

Object privileges	Authorized to
SELECT	Select rows from table or view
INSERT	Add new rows to table or view
DELETE	Remove some rows from table or view
UPDATE	Modify content of rows from table or view
EXECUTE	To run procedure
REFERENCES	To reference a particular table using foreign key and check constraint

3. Ownership privileges

- Whenever you create a database object (like table or view) with the CREATE statement, you will become its owner and get full privileges for the table. (Like SELECT, INSERT, DELETE, UPDATE, and all other privileges).
- All other users are having no privileges on the newly created database object.
- You as owner of database object can explicitly give grant privileges to any other user by using the GRANT statement.
- Whenever you are creating a view with the CREATE VIEW statement, you become the owner of that view, but you do not necessarily receive all privileges as you require the SELECT privilege on each of base tables on which view is defined.

6.13 Granting Privileges

Q. Write syntax for GRANT privileges. (2 Marks)

1. Introduction

- A system privilege is the right to perform a particular action, or to perform an action on any schema objects of a particular type.
- An authorized user may pass on this authorization to other users. This process is called as granting of privileges
- Generally GRANT statement is used by owner of table or view to give other users access permissions.
- In SQL user accounts must present in system before we can grant privileges to him.

2. Syntax

```
Grant <ALL | privilege list>
ON <relation name or view name>
TO <user | role list | PUBLIC >
   [WITH GRANT OPTION]
```

Privilege list	Meaning
ALTER	Table and views
CREATE	Table and views
DROP	Table and views
DELETE	Tables and views
INSERT	Tables and views
SELECT	Tables and views
UPDATE	Tables and views
ALL	Tables and views

WITH GRANT OPTION

Is used to allow user to grant privileges (which are granted to him) to other users.

3. Example :

- Consider an example for granting update authorization to the *Emp_Salary* relation of the company database. Assume that, initially that the DBA grants update authorization on *Emp_Salary* to other users U1, U2 and U3, who may in turn pass on this authorization to other users. This passing of authorization from one user to other users is called **authorization graph.**
- The following **grant** statement grants user U1, U2 and U3 the **select** privilege on *Emp_Salary* relation :

```
GRANT SELECT, INSERT
ON mydb.*
TO 'mahesh'@'somehost';
```

Following **grant** statement gives all users all authorization on the amount attributes of the *Emp_Salary* relation using public keyword;

```
GRANT ALL
ON *.*
TO 'mahesh'@'somehost';
```

(a) Database privileges

- SQL permits a user to declare foreign keys while creating relations.

Example : Allow user U1 to create relation that references key 'Eid' of Emp_Salary relation.

```
GRANT  ALL
ON  mydb.* TO 'mahesh'@'somehost';
```

(b) Table privileges

This privilege authorizes a user to execute a function or procedure.

```
GRANT  ALL
ON  mydb.mytbl
TO  'mahesh'@'somehost';

GRANT  SELECT, INSERT
ON  mydb.mytbl
TO  'mahesh'@'somehost';
```

(c) Column privileges

This privilege authorizes a user to execute a function or procedure.

```
GRANT  SELECT (col1), INSERT (col1, col2)
ON  mydb.mytbl
TO  'mahesh'@'somehost';
```

6.14 Revoking of Privileges

Q. Write a short note on Revoking of privileges. (4 Marks)

1. Introduction

- We can reject the privileges given to particular user with help of revoke statement.
- To revoke an authorization, we use the **revoke** statement.

2. Syntax

```
REVOKE    <ALL | privilege list>
ON  <relation name or view name>
FROM    <user | role list | PUBLIC>
[RESTRICT/ CASCADE]
```

- **CASCADE :** will revoke all privileges along with all dependent grant privileges
- **RESTRICT :** This will not revoke all related grants only removes that GRANT only.

3. Examples :

- The revocation of privileges from user or role may cause other user or roles also have to leave that privilege.
- This behaviour is called cascading of the revoke.

(a) To remove select privilege from users U1, U2 and U3.

```
REVOKE    SELECT
ON  mydb.mytbl
FROM   'mahesh'@'somehost';
```

(b) To remove update rights on amount column of Emp_Salary from U1, U2 and U3

```
REVOKE    UPDATE (amount)
ON  Emp_Salary
FROM   'mahesh'@'somehost';
```

(c) To remove reference right on amount column from user U1.

```
REVOKE    REFERENCES (amount)
ON  Emp_Salary
FROM   'mahesh'@'somehost';
```

The revoke statements may alternatively specify restrict if we don't want cascade behavior.

```
REVOKE    SELECT
ON  Emp_Salary
FROM   'mahesh'@'somehost'
RESTRICT;
```

6.15 Transaction Control Language (TCL)

1. Introduction

- Any SQL query can be executed with two basic operations on the database objects :
  - Read
  - Write

- After executing SQL query we must specify its final action as commit (save data) or abort (or revert back changes).
- The COMMIT statement ends the operations and makes all changes made to the data permanent, on successful completion.
- ABORT terminates and undoes all the actions done so far.

2. Commit Transaction

- A query that is successful and has encountered no errors is committed by issuing commit. That is, all changes to the database are made permanent and become visible to other users of the database.

The syntax is as follows :

```
COMMIT [WORK]
```

- The keyword WORK is not required, though it might be added for clarity; a simple COMMIT is usually all that is required.
- Microsoft SQL Server 2000 does support the SQL99 standard syntax — in addition to its own. The Microsoft syntax allows for committing named transaction whereas the standard one does not.

```
COMMIT [ TRAN [ SACTION ] [<transaction name>]]
```

- As you can see, only COMMIT is required, everything else is optional, and the keywords can be shortened (i.e., TRAN instead of TRANSACTION). Alternatively COMMIT WORK can be used.

3. Rollback Transaction

- A query that is unsuccessful and has encountered some type of error should be rolled back. That is, all changes to the database are undone and the database remains unchanged by the transaction.
- Transaction-processing systems ensure database integrity by recording intermediate states of the database as it is modified, then using these records to restore the database to a known state if a transaction cannot be committed.

Syntax

```
ROLLBACK
```

6.16 Solved Designing Problem

Example 6.16.1 : For the given database, write SQL queries.

Employee (Eid. Name, Street, City)

Works (Eid. Cid. salary)

Manager (Eid, Manager_Name)
Company(Cid, Company_name, city) (5 Marks)

Solution :

(i) Modify the database so that 'Jack' now lives in 'Newyork'.

```
MySQL>  UPDATE   Employee
    SET      City = 'Newyork'
    WHERE    Name = 'Jack';
```

(ii) Give all employees of 'ANZ corporation' a 10% raise in salary.

```
MySQL>  UPDATE   Works
    SET      Salary = (salary+(0.1*salary))
    WHERE    CID IN ( SELECT      Cid
             FROM   Company
             WHERE Company_name = 'ANZ corporation');
```

Example 6.16.2 : For the following given database ? Write SQL queries
person (driver_id #, name, address)
car (license, model, year)
accident (report_no, date, location)
owns (driver_id #, license)
participated (driver_id, car, report_number, damage_amount) (5 Marks)

Solution :

(i) Update the damage amount for car with licence number "Mum2022" in the accident with report number "AR2197" to Rs. 5000.

```
MySQL>  Update  Participated
    SET     Damage_amount = 500
WHERE    Report_number LIKE 'AR2197' AND Car = 'Mum2022';
```

Example 6.16.3 : For given database, write SQL queries.
Employee (EID, Name, Street, City)
Works (EID, CID, Salary)
Manager (EID, Manager_Name)
Company (CID, Company_name, City) (5 Marks)

Solution :

(i) Modify the database so that 'TRATHAM' now lives in USA

```
MySQL>      UPDATE      Employee
    SET     City = 'USA'
    WHERE   Name = 'TRATHAM';
```

(ii) Give all employees of 'SHARAYU Steel' a 10% raise in salary.

```
MySQL>  UPDATE   works
        SET      salary = (salary + (0.1 * salary))
        WHERE    cid IN ( SELECT      cid
        FROM     company
        WHERE    company_name = 'SHARAYU Steel');
```

Example 6.16.4 : Consider insurance database given below and answer the following queries in SQL.

Person (driver_id, name, address)

Car (license, model, year)

Accident (report_no, adate, location)

Owns (driver_id, license)

Participated (driver_id, license, report_no, damage_amount) **(5 Marks)**

Solution :

(i) Add new accident to database.

```
MySQL>  INSERT INTO Accident (report_no, adate, location)
        VALUES ('111', '01/01/2014','Pune');
```

(ii) Delete 'Santro' belonging to 'John Smith'.

```
MySQL>  DELETE
        FROM     CAR
        WHERE    Model = 'SANTRO'
        AND
        License IN ( SELECT license
                FROM  Owns
               WHERE  Driver_id IN ( SELECT driver_id
                        FROM person
                        WHERE name = 'John Smith')
        );
```

Example 6.16.5 : Consider the following employee database.

Employee (empname, street, city, date_of_joining)

Works (empname, company _name, salary)

Company (company_name, city)

Manages (empname, manager_name).

Write SQL queries for the following statements :

(i) Modify the database so that 'John' now lives in'Mumbai'.

(ii) Give all employees of ABC Corporation' a 10% raise **(5 Marks)**

Solution :

(i) Modify the database so that 'John' now lives in 'Mumbai'.

```
MySQL>  UPDATE  Employee
        SET     City= 'Mumbai'
        WHERE   Empname= 'JOHN';
```

(ii) Give all employees of ABC Corporation' a 10% raise.

```
MySQL>  UPDATE  Works
        SET     Salary=0.1*salary
        WHERE   Company_name= 'ABC Corporation';
```

Example 6.15.6 : Employees (Empid, Fname, Lname, Email, Phoneno, Hiredate, Jobid, Salary, Mid, Did)

Departments (Did, Dname, Managerid, Locationid)

Locations (Locationid, Streetadd, Postalcode, City)

Write the SQL queries for the following.

1. List the employees have a manager who works for a department based in the U.S.
2. Write a query to display the details of all employees in the Finance department.
3. Give 10% hike to all the employees working in Did 20.
4. Write a query to display all the information of the employees whose salary is within the range 1000 and 3000.
5. Display the information of all the employees whose first name starts with 'R' in descending order of their salary.

Solution :

1. List the employees have a manager who works for a department based in the U.S.

```
SELECT *
FROM Employees e
INNER JOIN Departments d ON e.did = d.did
INNER JOIN Locations l ON l.locationid=d.locationid
WHERE City ='US';
```

2. Write a query to display the details of all employees in the Finance department.

```
SELECT *
FROM Employees e
INNER JOIN Departments d ON e.did = d.did
WHERE Dname='Finance';
```

3. Give 10% hike to all the employees working in Did 20.

```
UPDATE employees
SET Salary = 1.1* Salary
```

```
WHERE did =20;
```

4. Write a query to display all the information of the employees whose salary is within the range 1000 and 3000.

```
SELECT *
FROM employees
WHERE Salary BETWEEN 1000 AND 3000;
```

5. Display the information of all the employees whose first name starts with R' in descending order of their salary.

```
SELECT *
FROM employees
WHERE Fname LIKE 'R%'
ORDER BY Salary DESC;
```

Example 6.15.7 : Employee(eid,ename,address,city)
Works(eid,cid,salary)
Company(cid,cname,city)

(1) Modify database so that John now lives in Mumbai

(2) Find Employees who live in same city as the company for which they work.

(3) Give all employees of "AZ Corporation" where there is increase in salary by 15%

(4) Find the names of all employees, company name and city of residence such that Employee name begins with 'I'

(5) Delete all tuples in works relation for employees of small bank corporation.

MU - Dec.18, 10 Marks

Solution :

1. Update employee

```
Set city = 'Mumbai'
Where name = 'John';
```

2. Sectect e-ename

```
From    employee e, company c, works w
When    e.eid = w.eid
And     w.Cid = c.cid
And     e.city = C.city ;
```

3. update employee

```
Set     sal = 1.15 * sal
Where eid = (Sected eid from emp
                where Cid = (select eid
```

```
                    from comp
                    where name)
```

4. select e.ename e.city, c. cname

```
select from e.city,
select where e.city
select And e.city
And e name like 'J'
```

5. delete

```
from employ
When eid = (Sectect eid
            Form works
            Where Cid = (Sectect Cid from comp
```

Review Questions

Q. 1 Explain various data definition statements in SQL.

Q. 2 Explain various data types used in SQL.

Q. 3 What is SQL ? Explain the following structures of SQL queries with appropriate example

(i) Select clause (ii) Where clause (iii) From clause

Q. 4 Write short note on : DDL and DML.

Q. 5 Write syntax for GRANT privileges.

Q. 6 Write a short note on Revoking of privileges.

□□□

CHAPTER 7

SQL Security

Module IV

Syllabus

Aggregate function-group by,having, Views in SQL, Nested and complex queries.

7.1 Aggregate Functions

Q. What is mean by aggregate function ?
Q. Write a note on various aggregate function.

Introduction

- Sometimes for decision making we need summarize data from table like average, sum, minimum etc.
- SQL provides various aggregate functions which can summarize data of given table. The function operates on the table data produces a single output.
- Such queries are generally used for producing reports and summary forms in an application.

7.1.1 Types of aggregate functions

(a) **COUNT ([DISTINCT] C) :** The number of (unique) values in the column C.

(b) **SUM ([DISTINCT] C) :** The sum of all (unique) values in the column C.

(c) **AVG ([DISTINCT] C) :** The average of all (unique) values in the column C.

(d) **MIN (C) :** The minimum value in the column C.

(e) **MAX (C) :** The maximum value in the column C.

Example :

Table 7.1.1 : Exam_Marks

Sid	SName	Marks
1	Mahesh	90
2	Suhas	80
3	Jyendra	89
4	Sachin	99
5	Vishal	88
6	Payal	90

7.1.1.1 COUNT ()

- This function is used to calculate number of rows (or records) in a table selected by query.
- COUNT returns the number of rows in the table when the column value is not NULL.
- Column in the query must be numeric.

Example :

Find total number of students in above Table 7.1.1.

```
SELECT Count(Sid) as Count
FROM   Exam_Marks;
```

COUNT
6

Let us consider above table modified as below,

(Some duplicate rows are present in Table 7.1.2).

Table 7.1.2

Sid	SName	Marks
1	Mahesh	90
1	Mahesh	90
2	Suhas	80
3	Jyendra	89
3	Jyendra	89
4	Sachin	99

Sid	SName	Marks
5	Vishal	88
6	Payal	90

In this case Query 1 results to wrong result as below,

```
SELECT  Count(Sid) as Count
FROM    Exam_Marks;
```

COUNT
8

So we can modify query as given below.

Example :

Find total number of diffrent students in above Table 7.1.2.

```
SELECT  Count(Distinct Sid) as Count
FROM    Exam_Marks;
```

COUNT
6

7.1.1.2 SUM ()

- This function is used to calculate sum of column values in a table selected by query.
- Column in the query must be numeric.
- Value of the sum must be within the range of that data type.

Example :

Find total of marks scored by all students.

```
SELECT SUM(Marks) as Sum
FROM   Exam_Marks;
```

SUM
446

7.1.1.3 AVG ()

- This function is used to calculate Average of column values in a table selected by query.
- This function first calculates sum of column and then divide by total number of rows.

- AVG returns the average of all the values in the specified column.
- Column in the query must be numeric.

Example :

Find avarage marks of students.

```
SELECT AVG(Marks) as AVG
FROM   Exam_Marks;
```

AVG
89.33

7.1.1.4 MIN()

- This function is used to find minimum value out of column values in a table selected by query.
- Column in the query need not be numeric data type.

Example :

Find minimum marks scored by students.

```
SELECT MIN(Marks) as Min
FROM   Exam_Marks;
```

MIN
80

7.1.1.5 MAX()

- This function is used to find maximum value out of column values in a table selected by query.
- Column in the query need not be numeric data type.

Example :

Find maximum marks scored by students.

```
SELECT MAX(Marks) as Max
FROM   Exam_Marks;
```

MAX
80

7.1.1.6 Summary of Aggregate Functions

Q. Compare various Aggregate functions.

Function	Description
AVG ([DISTINCT | ALL] n)	Average value of n, ignoring null values.
COUNT ({ * | [DISTINCT | ALL] expr})	Number of rows, where *expr* evaluates to something other than null.
MAX ([DISTINCT | ALL] expr)	Maximum value of *expr*, ignoring null values.
MIN ([DISTINCT | ALL] expr)	Minimum value of *expr*, ignoring null values.
SUM ([DISTINCT | ALL] n)	Sum value of n, ignoring null values.

7.2 GROUP BY Clause - Grouping Query Results

(1) Introduction

- Related rows can be grouped together by **GROUP BY** clause based on distinct values that exist for specified columns.
- A GROUP BY clause creates a set of data, containing several sets of records grouped together based on some condition.
- The GROUP BY statement is used with the SQL aggregate functions to group the retrieved data by one or more columns or expression.
- The ORDER BY keyword is used to sort the result-set by a specified column.
- The ORDER BY keyword sorts the records in ascending order by default.
- If you want to sort the records in a descending order, you can use the DESC keyword.
- The GROUP BY column does not have to be in the SELECT clause.
- A Grouping Query groups rows based on common values in a set of grouping columns. Rows with the same values for the grouping columns are placed in distinct groups. Each group is treated as a single row in the query result.

(2) Syntax

```
SELECT      column_name
FROM        table_name
GROUP BY    column_name;
```

(3) Example :

The **Student_Dept** table is as follows :

Student_Dept		
Sid	**SName**	**Dept**
1	Mahesh	IT
2	Anu	IT
3	Suhas	CE
4	Jyendra	CE
5	Umang	EXTC
6	Amruta	EXTC
7	Rahul	EXTC
8	Hiral	IT

Retrieve number of students in various departments.

```
SELECT      Dept, count (Sid) "Total Student"
FROM        Student_Dept
GROUP BY    Dept;
```

Dept	Total Student
IT	3
EXTC	2
CE	3

You can use the group function in the ORDER BY clause.

```
SELECT      Dept, count (*) "Total Student"
FROM        Student_Dept
GROUP BY    Dept;
```

Dept	Total Student
CE	3
EXTC	2
IT	3

7.3 HAVING Clause – Filtering grouped Query Results

7.3.1 Apply Conditions with GROUP BY

1. WHERE clause

- A Grouping Query can also group rows based on common values in a set of grouping columns and with specified condition.
- The WHERE clause specifies the rows to be retrieved. Since there is no WHERE clause, all rows are retrieved by default.
- Retrieve customer information whose purchase amount exceeds grouped by Customer id.

```
SELECT      *
FROM        Customer_Loan
WHERE       Amount<4000
GROUP BY    Cust_id ;
```

Cust_id	Loan_no	Amount
103	2010	2555.00
103	2015	2000.00
104	2056	3050.00

2 HAVING clause

- HAVING is conditional clause which checks data for specific search condition.
- A HAVING clause is like a WHERE clause, but applicable only to groups as a whole (that is, to the rows in the result set representing groups), whereas the WHERE clause applies to individual rows.
- A query can contain both a WHERE clause and a HAVING clause.

Example :

```
SELECT      *
FROM        Customer_Loan
GROUP BY    Cust_id
HAVING      Cust_id >103 ;
```

Cust_id	Loan_no	Amount
104	2056	3050.00

- Difference in operation of WHERE and GROUP BY clause
  - Execution hierarchy.

- The WHERE clause is applied first to the individual rows in the result set then rows that meet the conditions in the WHERE clause are grouped using group by clause.
- The HAVING clause is then applied to the rows in the result set that is produced by grouping.

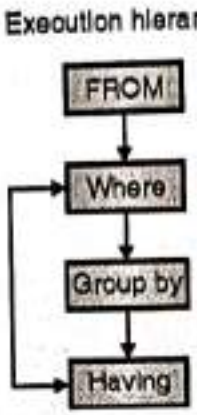


Fig. 7.3.1

- Only the groups that meet the HAVING conditions appear in the query output. You can apply a HAVING clause only to columns that also appear in the GROUP BY clause or in an aggregate function.
- The **Student_Dept** table is as follows :

Sid	SName	Dept
1	Mahesh	IT
2	Anu	IT
3	Suhas	CE
4	Jyendra	CE
5	Umang	EXTC
6	Amruta	EXTC
7	Rahul	EXTC
8	Hiral	IT

Retrieve departments HAVING more than two students in it.

```
SELECT    Dept, count (Sid) "Total Student"
FROM      Student_Dept
GROUP BY  Dept
HAVING    count (Sid)>2;
```

Dept	Total Student
IT	3
EXTC	2
CE	3

7.4 Introduction of Views

Q. What is a view ? How is it created and stored ? **(4 Marks)**
Q. What is the view in SQL, how it is defined ? Discuss the problem that may arise when we attempt to update a view. How views are implemented ? **(10 Marks)**
Q. Describe view. **(3 Marks)**
Q. What are the types of views ? **(2 Marks)**

1. Definition

A view is defined as a database object that allows us to create a virtual table in the database whose contents are defined by a query or taken from one or more tables.

View is defined to hide complexity of query from user.

2. Base table

The table on which view is defined is called as Base table.

3. View - As a window of entire table

Instead of showing entire table to a user we can show a glimpsed of table to the user which is required for him

Example :

Consider a student table contains following columns,

STUDENT (Stud_Id, Stud_Name, Std, Div, Addr, Sports, Fees, Cultural_Activity)

- Now for a sports teacher requires only sports related data of students so we can create view called as Stud_Sports_View for teacher as below which will only depicts sports data of student to sports teacher.

Stud_Sports_View (Stud_Id, Stud_Name, Sports)

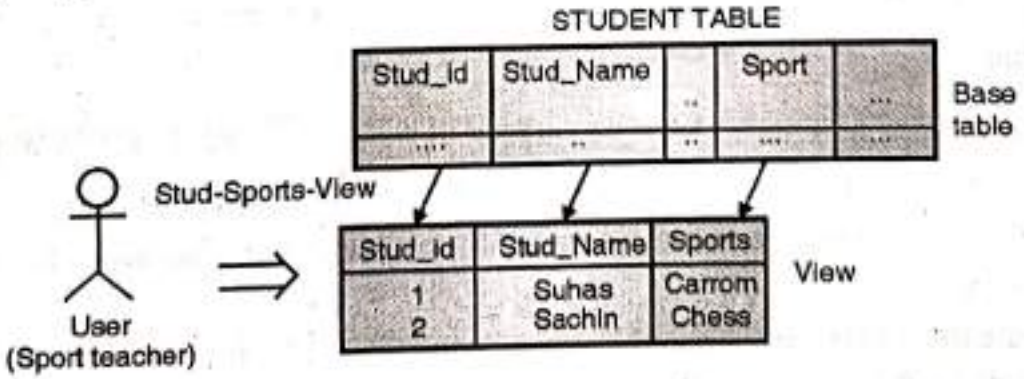


Fig. 7.4.1 : Overview of view

4. Types of views

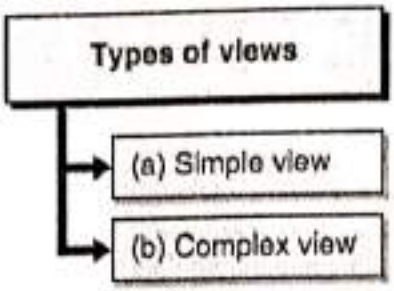


Fig. 7.4.2 : Types of Views

(a) Simple view

- The views which are based on only one table called as Simple view.
- Allow to perform DML **(Data Manipulation Language)** operations with some restrictions.
- Query defining simple view cannot have any join or grouping condition.

(b) Complex view

- The views which are based on more than one table called as complex view.
- Do not allow DML operations to be performed.
- Query defining complex view can have join or grouping condition.

5. Working of views

- When we call view in SQL query it refers to database and finds definition of views which is already stored in database.
- Then the DBMS convert this call of view into equal request on the base tables of the view and carries out the operations written in view definition and returns result set to query from which view is called.

7.4.1 Creating a Views

Q. Explain syntax for creating views. (2 Marks)

1. Introduction

- To create a view a subquery must be embedded within the CREATE VIEW statement.
- A simple query is designed and its output can be recorded as a view.
- The CREATE statement assigns a name to the view and also gives the query which defines the view.
- To create the VIEW one should have privileges to access all of the base tables on which view is defined.

- Also, the user must have create view permissions from DBA to create a view in the database.
- The create view can change the name of the column in view as per requirements.

2. Syntax

```
CREATE [OR REPLACE] VIEW <view name>
AS
SUB QUERY
[WITH CHECK OPTION]
```

- **OR REPLACE :** Change the definition of a view without dropping (ALTER VIEW)
- **VIEW NAME :** Is name given to a view.
- **SUB QUERY :** The query which retrieves the columns of the table that query must have.
- **WITH CHECK OPTION :** This is type of check constraint, which specifies that only those rows which are selected by view can be inserted updated or deleted.

3. Example :

- In the college database, we may want to let a Head of Departments see only the FACULTY rows for own department.

```
SQL> CREATE VIEW IT_Faculty
     AS
     SELECT *
     FROM   FACULTY
     WHERE      Faculty_Dept= 'IT';

/* Selecting Data */

SQL> SELECT *
     FROM  IT_Faculty;
```

- Consider a default HR schema we will create a view of employees having salary below 3000 and retrieve data from view.

```
SQL> CREATE VIEW EmpBelow3K
     2 AS
     3 SELECT *
     4 FROM Emp
     5 WHERE Sal < 3000;

View created.
```

```
SQL> SELECT * FROM EmpBelow3K;
```

EMPNO	ENAME	JOB	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	01-MAY-81	2850		30
7782	CLARK	MANAGER	09-JUN-81	2450		10
7566	JONES	MANAGER	02-APR-81	2975		20
7369	SMITH	CLERK	17-DEC-80	800		20
7499	ALLEN	SALESMAN	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	28-SEP-81	1250	1400	30
7844	TURNER	SALESMAN	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	12-JAN-83	1100		20
7900	JAMES	CLERK	03-DEC-81	950		30
7934	MILLER	CLERK	23-JAN-82	1300		10

11 rows selected.

Create a view of employee's jobs having salary below 3000 and retrieve data from view

```
SQL> CREATE VIEW EmpBelow3K
  2 AS
  3 SELECT DISTINCT(Job)
  4 FROM Emp
  5 WHERE Sal < 3000;
View created.
SQL> SELECT * FROM JobBelow3K;
JOB
---------
CLERK
SALESMAN
MANAGER
3 rows selected.
```

7.4.2 Dropping Views

Q. How to drop view ? Give Syntax. **(2 Marks)**

1. Introduction

– To drop a view we use DROP VIEW statement.

- The DROP VIEW statement requires a name to the view.
- To DROP the VIEW one should have privileges to from DBA to DROP a view in database.
- The DROP view dose not affects base table or any column of base table.

2. **Syntax**

```
DROP VIEW <View_name> [RESTRICT|CASCADE]
```

- **RESTRICT :** Delete view only if there is no other view dependent on original view.
- **CASCADE :** Delete view along with all dependent views on original view.

3. **Example :**

Remove a view created in above step.

```
SQL>  DROP VIEW JobBelow3K;
View Dropped;
```

7.4.3 Modifying a Views

Q. Give syntax for altering views. (2 Marks)

- There are some situations in which some modifications are required in view definition.
- The CREATE OR REPLACE statement in view syntax is used to modify view.
- This statement is used by SQL to overwrite the old view definition with new definition without raising any error like existing view with same name.

Syntax

```
CREATE OR REPLACE VIEW <View_name>
AS
SUB QUERY
[WITH CHECK OPTION]
[WITH READ ONLY];
```

Example :

Consider a view defined above 'IT_Faculty' and change it to select all IT faculties having salary above Rs. 25000.

```
SQL> CREATE OR REPLACE VIEW IT_Faculty
     AS
     SELECT *
     FROM  FACULTY
     WHERE    Faculty_Dept= 'IT' AND Salary > 25000;
```

7.4.4 Advantages of Views

Q. What are advantages of views ? **(5 Marks)**

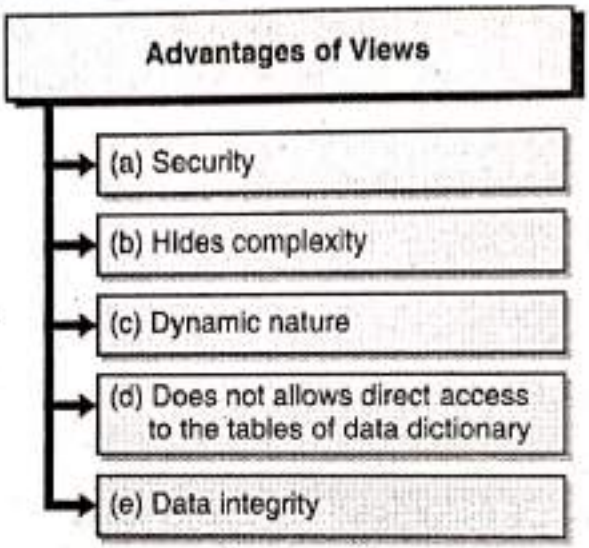


Fig. 7.4.3 : Advantages of Views

(a) Security

- View can restrict user from accessing all data.
- In case of view only data that is given in view is accessible to user. So all data of base table is not accessible to user which will give you security of information.
- For example sports teacher can see data related to sports only and view preventing him from manipulating data pertaining to fees of students.

(b) Hides complexity

- The view may be result of very complex query. Hence instead of writing such complicated query again and again we can store such result to a view and access it whenever we want to access.
- So by writing query we can hide the complexity of original query.

(c) Dynamic nature

- View definition remains unaffected although there is any change in structure of a table.
- This dynamic nature does not hold true in case if base table is dropped or the column selected by view is altered.

Example :

If view is made on two tables, selecting two columns from first table and two columns from second table if we add one more column to first table does not cause any change on view.

(d) Does not allows direct access to the tables of data dictionary

- This act like functionality of safeguard to data stored in the data dictionary.
- By this way user cannot change data dictionary to damage database.
- Views can helps to make data in data dictionary easily comprehensible and helpful.

(e) Data integrity

If data is accessed through a view, the DBMS can automatically check the data to check for specified integrity constraints.

7.4.5 Disadvantages of Views

Q. What are disadvantages of views ? (3 Marks)

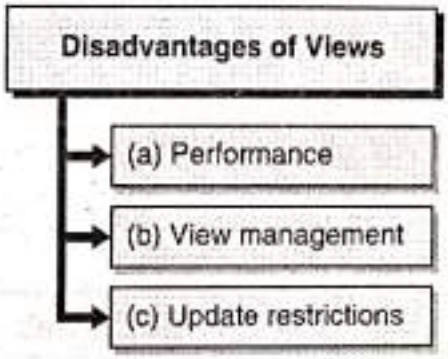


Fig. 7.4.4 : Disadvantages of views

(a) Performance

- DBMS translates queries of view to queries on base table.
- Sometimes a simple query may take longer time to run. If view is defined by complex multi table query.
- As the complexity of query is hidden by view hence, users are not aware of how much complicated task the query is actually performing.

(b) View management

- The view should be created as per standard then it will simplify the job of DBA.
- This happens generally when views are references to the other views.
- We need to keep all information of all views in such case so as to it will become very difficult to manage views.

(c) Update restrictions

- When a user tries to update a view, the DBMS must translate this query into an update on rows of the underlying base tables.

- Update is possible for simple views.
- Complex views cannot be updated as they are read-only type of views.

7.5 Nested and Complex Queries

Q. Explain concept of sub query.
Q. Describe various operators for multiple sub query.
Q. What is nested sub query ? Explain ANY ALL operators with example.

- A subquery is a "query within a query".
- Subquery is query appear within **WHERE** or **HAVING** clause of other query.
- Outer query is called as **main query** and inner query which is written in (where or having clause) main query is called **subquery**.
- **Subquery in the WHERE clause :** The result of the subquery (inner query) is used to select some rows from main query.

SELECT .... **FROM**	
WHERE .... (<,>,<=,>=,<>)	
	(Sub Query)
HAVING .... (IN/ANY/ALL)	
	(Sub Query)

Fig. 7.5.1 : Subquery

- **Subquery in the HAVING clause :** The result of the subquery (inner query) is used to select some groups from main query.
- Sub queries can be nested within other sub queries.

Syntax

```
SELECT    select_list
FROM      table
WHERE     expr_operator ( SELECT select_list FROM table)
```

Expr_operator can be of two types like,

(a) Single row operator (<,>,<=,>=,<>)

(b) Multiple row operator (IN,ANY,ALL)

7.5.1 Independent Subquery

- A subquery is one which returns only one row to main query.
- A subquery which uses single row operators in above syntax is called as single row subquery.

Example :

Find all employees whose salary less than Smriti's salary

```
SELECT   ENAME, Salary
FROM     EMPLOYEE
WHERE    Salary <
                    (SELECT   Salary
                     FROM     EMPLOYEE
                     WHERE    ENAME    = 'Smriti' );
```

50,000 Smriti's salary

Ename	Salary
Mahesh	60000
Jay	51000

List the books whose average sale of books published by TechMax is higher than overall average sale. (Use of subquery in having)

```
SELECT      NAME, AVG (Sale)          // Outer Query
FROM        SALES, BOOKS
WHERE       Publication = 'TechMax'
AND         SALES.Bid = Books.Bid
GROUP BY book_name
HAVING      AVG (Sale) > (

Average sale    SELECT      AVG (Sale)      // Sub query
Of books published  FROM    SALES   sale    overall average
By TechMax Publication      )
```

Name	AVG(Sale)
DBMS	500
TCS	400

As only one row (Value) is returned by inner query (Subquery) so these types of query is called as Single row subquery.

7.5.2 Multiple Row Subquery

- If subquery returns more than one row then that type of query known as multiple row sub query.
- A subquery which uses multiple row operators in above syntax is called as multiple row subqueries.
- The quantified tests (i.e. **ANY** and **ALL**) uses any one of the simple comparison operators to compare a test value to all of the values returned by a subquery, checking to see whether the comparison holds for some or all of the values.

Sr. No.	Operator	Meaning	Name
1.	**IN**	Equal to any member in the list	**Set Membership Test**
2.	**ANY**	Compare value to each value returned by the subquery	**Quantified Tests** **Set Comparison**
3.	**ALL**	Compare value to every value returned by the subquery	
4.	**EXISTS**	Checks weather subquery returns a value or not.	**Existence Tests**

Review Questions

Q. 1 What is a view ? How is it created and stored ?

Q. 2 What is view in SQL ? Discuss the problem that may arise when we attempt to update a view. How views are implemented ?

Q. 3 Write short notes on Views in SQL.

Q. 4 Explain syntax for creating views.

Q. 5 What is a view ? How is it created and stored ?

□□□

CHAPTER 8

Trigger

Module IV

Syllabus

Triggers.

8.1 Trigger

Q. What are triggers? Explain with example. **(10 Marks)**
Q. Describe opTrigger. **(3 Marks)**

1. Introduction

- A trigger is a procedure that is automatically invoked by the DBMS in response to specific alteration to the database or a table in database.
- Triggers are stored in database as a simple database object.
- A database that has a set of associated triggers is called an active database.
- A database trigger enables DBA (Database Administrators) to create additional relationships between separate databases.

2. Components of Trigger (E-C-A model)

- **Event (E)** - SQL statement that causes the trigger to fire (or activate). This event may be insert, update or delete operation database table.
- **Condition (C)** - A condition that must be satisfied for execution of trigger.
- **Action (A)** - This is code or statement that execute when triggering condition is satisfied and trigger is activated on database table.

3. Trigger syntax

```
CREATE [OR REPLACE] TRIGGER <Trigger_Name>
[<ENABLE | DISABLE>]
<BEFORE | AFTER>
<INSERT | UPDATE | DELETE>
ON <Table_Name>
```

```
    [FOR EACH ROW]
DECLARE
    <Variable_Definitions>;
BEGIN
    <Trigger_Code>;
END;
```

OR REPLACE	If trigger is already present then drop and recreate the trigger
<Trigger_Name>	Name of trigger to be created.
BEFORE	Indicates that trigger is to be fired before the triggering event occurs.
AFTER	Indicates that trigger is to be fired After the triggering event occurs.
INSERT	Indicates that trigger is to be fired whenever insert statement adds a row to table.
UPDATE	Indicates that trigger is to be fired whenever Update statement modifies a row in a table.
DELETE	Indicates that trigger is to be fired whenever delete statement removes a row from table.
FOR EACH ROW	Trigger will be fired only once for each row.
WHEN	Contains condition that must be satisfied to execute trigger.
<trigger_code>	Code to be executed whenever triggering event occurs

Fig. 8.1.1 : Trigger parameters

4. Trigger types

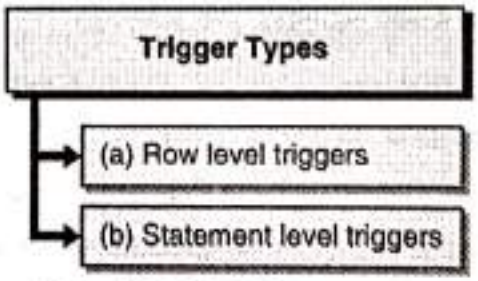


Fig. 8.1.2 : Trigger types

(a) Row level triggers

- **A row level trigger** is fired each time the table is affected by the triggering statement.
- For example, if an UPDATE statement changes multiple rows in a table, a row trigger is fired once for each row affected by the UPDATE statement.
- If a triggering statement do not affect any row then a row trigger will not run only.
- If FOR EACH ROW clause is written that means trigger is row level trigger.

(b) Statement level triggers

- A statement level trigger is fired only once on behalf of the triggering statement, irrespective of the number of rows in the table that are affected by the triggering statement.
- This trigger executes once even if no rows are affected.
- For example, if a DELETE statement deletes several rows from a table, a statement-level DELETE trigger is fired only one time.
- This is Default type, when FOR EACH ROW clause is not written in trigger that means trigger is statement level trigger.

5. Trigger example

- Creating a trigger on employee table whenever new employee added a comment is written to EmpLog Table.
- Let us write a trigger and study its effect.

Example :

```
SQL> CREATE OR REPLACE TRIGGER AutoRecruit
  2   AFTER INSERT ON EMP
  3   FOR EACH ROW
  4   BEGIN
  5   Insert into EmpLog values ('Employee Inserted');
  6   END;
  7   /

Trigger created.

SQL> INSERT INTO EMP
2 VALUES
3 (1,'Mahesh','Manager','1-JAN-1986',3000,null,10);

1 row created.

SQL> SELECT * FROM EmpLog;

STATUS
------------------------
Employee Inserted
```

– Consider another example, whenever there comes a new student add him to CS (Computer Science).

Example :

```
SQL>    CREATE TRIGGER CSAutoRecruit
AFTER INSERT ON Student
FOR EACH ROW
BEGIN
INSERT INTO Take VALUES (111, 'CS');
END;
```

6. Trigger operations

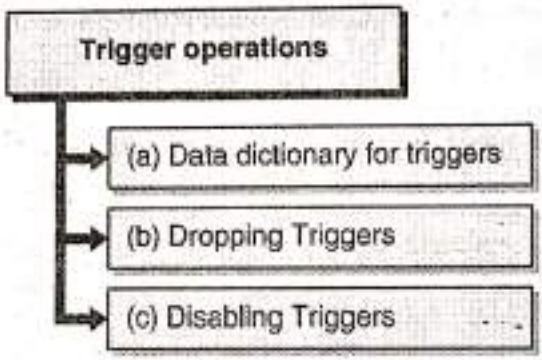


Fig. 8.1.3 : Triggers operations

(a) Data dictionary for triggers

Once triggers are created their definitions can be viewed by selecting it from system tables as shown below :

Syntax

```
MySQL>Select *
From   User_Triggers
Where  Trigger_Name = '<Trigger_Name>';
```

This statement will give you all properties of trigger including trigger code as well.

(b) Dropping Triggers

To remove trigger from database we use command DROP

Syntax

```
MySQL> Drop trigger <Trigger_Name>;
```

(c) Disabling Triggers

To deactivate trigger temporarily this can be activated again by enabling it.

Syntax

```
MySQL> Alter trigger <Trigger_Name> {disable | enable};
```

7. Trigger advantages

(i) Triggers are useful for enforcing referential integrity, which preserves the defined relationships between tables when you add, update, or delete the rows in those tables. Make sure that a column is filled with default information.

(ii) After finding that the new information is inconsistent with the database, raise an error that will cause the entire transaction to roll back.

8. Trigger disadvantages

A trigger **hampers the performance of system** as database operations will go on slower due to triggering action.

(i) Restrictions on triggers

- You cannot modify the same table on which triggering event is written.
- You cannot modify a table which is connected to the triggering table by primary key foreign key relation.

(ii) Mutating table errors

- This happens when the trigger is querying or modifying table whose modification activated the trigger, or a table that might need to be updated because of a foreign key constraint with a CASCADE policy.
- This problem is called as **mutating table problem.**
- **Error :** ORA-04091: table name is mutating, trigger/function may not see it.

9. Solution for mutating table error

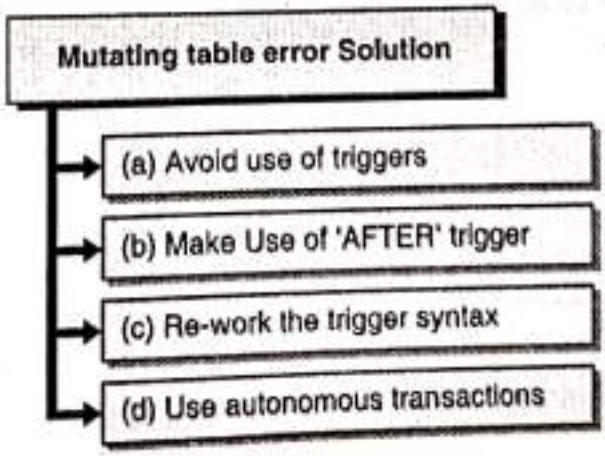


Fig. 8.1.4 : Mutating table error solution

The mutating table error is usually the result of a poor application design and mutating triggers should be avoided whenever possible.

(a) Avoid use of triggers

The best way to avoid the mutating table error is not to use such triggers. While the Oracle provides methods or procedures that are associated with tables, generally PL/SQL developers avoid triggers unless absolutely necessary.

(b) Make Use of 'AFTER' trigger

If use a trigger is must for you, then it is best to avoid the mutating table error by using 'after' trigger, to avoid the currency issues associated with a mutating table. For example, using a trigger "after update on salary", the original update has completed and the table will not be mutating.

(c) Re-work the trigger syntax

We can avoid mutating tables with a combination of row-level and statement-level triggers.

(d) Use autonomous transactions

You can avoid the mutating table error by marking your trigger as an autonomous transaction, making it independent from the table that calls the procedure.

Review Questions

Q. 1 What are Triggers ? Give an example.

Q. 2 Explain working of triggers with help of example. In which conditions Triggers cannot be used in database.

Q. 3 What are triggers ? Give an example. Illustrate the cases when triggers must not be used.

□□□

CHAPTER 9

Relational Database Design

Module IV

Syllabus

Pitfalls in Relational-Database designs, Concept of normalization, Functional Dependencies, First Normal Form, 2nd, 3rd, BCNF, multi valued dependencies, 4NF.

9.1 Pitfalls in Relational Database Design

(i) Introduction

- Pitfalls in database design suggest the effects when database design is wrong or incorrect.
- When we design any relational database schema, we need to take care of many important aspects of databases in order to avoid problems to database.

(ii) Pitfalls in database design : bad database design may due to

- Redundant data
- Inability to represent some data
- Dependency of various attributes of relation
- Loss of information

(a) Redundant data

- Redundancy is root of many problems in RDBMS.
- Redundant data causes following problems
  - Insert, delete or update anomalies
  - Wastage of storage
  - Generation of invalid data
- Consider a example of Employee_Project table in which employee information and information about project for which he works is kept as follows :

Eid	Ename	Address	Project_Id	Project	Salary
1	Sachin	Malad	12	IPF	25000
2	Jayendra	Vashi	44	CAP	30000
3	Suhas	Andheri	55	FID	12000
1	Sachin	Malad	24	LEX	25000
2	Jayendra	Vashi	24	LEX	30000
3	Suhas	Andheri	12	IPF	12000

- In above example there is repeatable data present in table which causes wastage of space in database, also this data complicates the update operations.
- In above design (modified design) as data for both table employee and project are kept together in one table hence, if we want to update a project name then entire record (tuple) which is updated will have all data for both tables.
- In case of adding new row to table we need to give information about both tables.
- Hence this approach will lead to redundant data in table. So this design shows bad database design.
- To avoid redundancy, we can make use of functional dependency by which we can perform decomposition of data into multiple table but that causes generation of spurious or invalid data.

(b) Inability to represent some data

- Another problem with design in above example is that we cannot represent information directly in any project, unless there is at least one employee working for that project.
- As above schema requires data about employee such as eid, ename etc.
- One solution that we can find is Null values to be inserted for the following attributes :
  - When attributes are not applicable to this tuple
  - When attributes value is unknown
  - When Value is known but absent (not yet recorded)

- But these Null values may introduce many problems
- It is difficult to handle Null values in schema.
- It becomes difficult to specify joins and other operations.

(c) Dependency of various attributes of relation

- Dependency or semantics specifies how to interpret attribute values in a tuple of the relation are related with each other.
- If the database design is done carefully, followed by a systematic mapping into relations, most of the semantics/dependencies will have been accounted for and the resulting design should have a simple structure.

Example : EMP (Emp_Id, Name, Salary)

- Salary is dependent on Emp_Id, as each employee will have specific salary.

(d) Loss of information

- Relational database design gives us summary information in form of tables or relations.
- Hence many a time some information is not possible to include in database design which causes a loss of some information or information loss may be the result of some operation on relation.
- This information that was lost can be useful or useless.
- If information is useful then it will be a severe problem for understanding.

Example : "Every student have different grasping power".

- In above statement we cannot have attribute to actually measure grasping power for each student.

9.2 Design Guidelines for Relational Schema

Q. Describe various design guidelines for relational schema. (5 Marks)

(1) Introduction

- In previous chapters, we have already studied various aspects of the relational model and the languages associated with it. Each relation schema consists of a number of attributes, the relational database schema consists of a number of relation schemas.

Relational Schema (Table Structure)	Relational Database Schema (Database Structure)
$R_1 (A_1, A_2, .., A_n)$ $R_2 (B_1, B_2, .., B_n)$ ... $R_m (C_1, C_2, .., C_n)$	$R_1, R_2, ..., R_n$

Where,

$R_1, R_2, ... R_n$ = Relation or Tables

A, B, C = Attributes or Columns

- The attributes are grouped to form a relation schema by mapping a conceptual data model i.e. ER or EER data model to relational schema.
- The relational mapping will identify entity types and relationship types and their respective attributes.

(2) Goodness of Relational Design

- In process of database design, we need to develop some measure of goodness for the quality of the design. In this module, we will discuss some measure of goodness to evaluate design quality of relational schemas.
- The above measure will help the developer to analyze why one grouping of attributes is better than another grouping of attributes.
- There are two levels for measuring goodness of relation schemas.

Logical (or Conceptual) Level

- The goodness of relational schema depends on how users understand the meaning of various attributes of relation.
- This will helps users to understand the meaning of the data stored in the relations, and hence it will make easy to formulate relational queries correctly.

Implementation (or Physical Storage) Level

- It helps user to understand how the tuples in a base relation are stored and updated.
- This level is applied to base relational schema stored as files.

(3) Database Design Approach

The database design can be done using the bottom-up approach to design relational schema using individual attributes of relation or top-down approach to identify individual attributes from relational schema.

Bottom-up design methodology / Design by Synthesis

- The basic relationships among individual attributes are identified then it will be combined to construct relation schemas.
- This is not very popular approach as it needs to study many binary relationships among attributes at the beginning which is very difficult.

Top-down design methodology / Design by Analysis

- The starts with relational schema and it will be further decomposed to a group of attributes to achieve desirable properties.
- This is very practical approach and used in real world database projects.

(4) Guidelines for pitfalls in Relational Database Design

- To determine the quality of relation schema design some informal guidelines can be used.
- **Guideline 1 :** Clear semantics of the attributes in relational schema
  - Semantics of attribute should be very clear in relational schema so that relational schema will have some real-world meaning associated with it.
  - The relational schema has a clear meaning associated with it.

Example :

- Employee (Emp_Id, Ename, Address, Salary)
- The Employee table contains information about all employees in company with their address and Salary.
- **Guideline 2 :** Reducing the Redundant Data in Tuples
  - A relational schema may have some redundancy in database design, if it stores data redundantly.
  - If same data is stored at more than one location will leads to redundancy and wastage of memory space.
  - **Data Anamolies :** An inconsistent data may cause some problems while adding, updating or deleting data in table which is called as data anomalies.
  - Redundant data is more vulnerable to various data anomalies as if data is updated at only one location and not at other locations, then that data becomes inconsistent, and this problem referred to as an update anomaly.
  - A normalized database stores non-primary key data in only one location.
  - A relational database table should avoid all data anomalies.

Example :

Employee (Emp_Id, Ename, Address)

Emp_Salary (Emp_Id, Ename, payScale, grossSalary, netSalary)

Emp_Designation (Emp_Id, Ename, Desg, fromDate, toDate)

(a) Update Anomaly

- The relational schema may have same data stored in multiple relations, if we update such data from only one relation may result in logical inconsistencies.

Example :

- All 3 tables contain the Ename attribute, thus any change in name of one employee will lead to updating his name in all 3 tables. Otherwise, if all the records are not updated then some tables may leave in an inconsistent state.

(b) Insertion Anomaly

- There is a possibility in which certain facts cannot be recorded in database.
- An Insert Anomaly arises when certain attributes cannot be added into the database without the presence of other attributes.

Example :

- It is not possible to add a row in Emp_Salary table or Emp_Designation table for an employee who is not exists in employee table.

(c) Deletion Anomaly

- If data deleted from one table all relevant data in another related tables must also be deleted otherwise it will create data inconsistency problem.
- Deletion of some data from one relation necessitates the deletion of some other data in other table.

Example :

- It is not possible to delete a row in Employee table if Emp_Salary table or Emp_Designation table contains data for respective employee.
- **Guideline 3 :** Reducing Null values in Tuples
  - A value of NULL is different from an empty, White spaces (blank spaces) or zero value.
  - Null values in tuple will cause wastage of memory space and it will also create problem of understanding.
  - Relations should be designed in such a way that their tuples should not contain any NULL values.
  - We can at least try make number of NULL values as low as possible.
  - Attributes with NULL values can be placed in separate relations with the primary key.

- There are certain resons for Null values,
- Not applicable data
- Invalid data
- Unknown data or data not available

Guideline 4 : Disallowing Spurious Tuples

- The bad designs of a relational database may result in erroneous results for some JOIN operation. As it is not possible to get original relation data from new relation after JOIN operation.
- The relational schema must be designed to satisfy the property of lossless join.
- If original relation contains fewer number of tuple then tuples generated by doing a natural-join of original relations.

9.3 Normalization Process

Q. What is Normalization ? **(2 Marks)**
Q. Define Normalization. **MU - Dec. 18, 2 Marks**

(1) Introduction

- Normalization is a step by step decomposition of complex records into simple records.
- Normalization is a process of organizing data in database in more efficient form. It results in tables that satisfy some constraints and are represented in a simple manner.
- This process is also called as canonical synthesis.
- Normalization is a step by step decomposition and database designers may not normalize relation to the highest possible normal form.
- The relations may be left in a lower normal form like 2NF, which may cause some penalties like data anomalies.

(2) Definition

***Normalization** is a process of designing a consistent database by minimizing redundancy and ensuring data integrity through decomposition which is lossless.*

(3) Goals of Database Normalization

1. Ensures data integrity.
2. Prevents redundancy in data.
3. To avoid data anomaly

(a) Update anomaly

(b) An insertion anomaly

(c) Deletion anomaly

9.4 Functional Dependencies

Q. Explain concept of functional dependency. (4 Marks)

(1) Introduction

- The concept of functional dependency is given by E. F. Codd.
- Functional Dependency (FD) provides a constraint between various attributes of a relation.
  - Functional dependencies are restrictions imposed between two set of attributes in relation from a database.
  - In a Relation R with attributes X and Y represented as R(X, Y), where Y is functionally dependent on other column X or we can say X functionally determines Y.
  - This dependency can be denoted with help of arrow (→)

    $X \rightarrow Y$
  - The data value in column Y must change when data value in another column X is modified.
  - All the attributes before arrow is called as **determinant** and attributes after arrow is called as determine.

(2) Example :

Consider an employee table with columns as shown in Table 9.4.1

Table 9.4.1 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

Case 1 : (X→Y)

- Consider an Employee table for specific employee_Id there is one and only one Ename

whereas for another employee_Id there can be other Ename.

Employee_Id → Ename

- As per above constraint, it is possible to have multiple employees with same Ename and different Employee_Id. But it is not allowed to have two employees with same Employee_Id and different Ename.

Case 2 : (X→ YZ)

- In above Employee table using below given functional dependency, for specific employee_Id there is one and only one set of Ename and Salary whereas for another employee_Id there can be other values of Ename and salary.

Employee_Id → Ename, Salary

- As per above constraint, it is possible to have multiple employees with same Ename and Salary.

Case 3 : (XY→ ZW)

- In above Employee table using below given functional dependency, for one employee_Id and Project_Id pair there is only one amount of time spent (Hours) and allowance given by company whereas for another pair there can be other values of Hours and Allowance.

Employee_Id, Project_Id → Hours, Allowance

- As per above constraint, it is possible to have multiple employee_Id and Project_Id pairs with same values of Hours and Allowance.

9.5 Solved Examples on Functional Dependencies

Example 9.5.1 : List all functional dependencies satisfied by the relation. **(5 Marks)**

a	b	c
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

Solution :

To find out all functional dependencies satisfied by the relation, we must remember one determinant (attributes before arrow) will give one and only one value of determine (attribute after arrow).

A	B	C	Tuple
a_1	b_1	c_1	Tuple 1
a_1	b_1	c_2	Tuple 2
a_2	b_1	c_1	Tuple 3
a_2	b_1	c_3	Tuple 4

Step 1 : Test for type 1 FD (X→Y)

Functional Dependency	Violated First by Tuple	Result
$A \rightarrow B$	--	Valid FD
$A \rightarrow C$	Tuple 2 $a_1 \rightarrow c_1$ $a_1 \rightarrow c_2$	Not Valid FD
$B \rightarrow A$	Tuple 3	Not Valid FD
$B \rightarrow C$	Tuple 2	Not Valid FD
$C \rightarrow A$	Tuple 3	Not Valid FD
$C \rightarrow B$	--	Valid FD

Step 2 : Test for type 1 FD (X→YZ)

Functional Dependency	Violated First by Tuple	Result
$A \rightarrow BC$	Tuple 2 $a_1 \rightarrow b_1c_1$ $a_1 \rightarrow b_1c_2$	Not Valid FD
$B \rightarrow AC$	Tuple 2	Not Valid FD
$C \rightarrow AB$	Tuple 3	Not Valid FD

Step 3 : Test for type 1 FD (XY→Y)

Functional Dependency	Violated First by Tuple	Result
$AB \rightarrow C$	Tuple 2 $a_1b_1 \rightarrow c_1$ $a_1b_1 \rightarrow c_2$	Not Valid FD
$AC \rightarrow B$	--	Valid FD
$BC \rightarrow A$	Tuple 3	Not Valid FD

Therefore, all FDs satisfied by relation are,

$A \rightarrow B$; $C \rightarrow B$; $AC \rightarrow B$

Example 9.5.2 : Consider the following relation. **(10 Marks)**

A	B	C	Tuple #
10	b_1	c_1	#1
10	b_2	c_2	#2
11	b_4	c_1	#3
12	b_3	c_4	#4
13	b_1	c_1	#5
14	b_3	c_4	#6

Given the previous state which of the following dependencies may hold in the above relation ? If the dependency cannot hold explain why by specifying the tuples that cause the violation :

(i) $A \rightarrow B$ (ii) $B \rightarrow C$ (iii) $C \rightarrow B$

(iv) $B \rightarrow A$ (v) $C \rightarrow A$

Solution :

To find out all functional dependencies satisfied by the relation, we must remember one determinant (attributes before arrow) will give one and only one value of determine (attribute after arrow).

(i) $A \rightarrow B$

As in above relation to test $A \rightarrow B$, $\{10\} \rightarrow \{b1\}$ but in Tuple # 2 $\{10\} \rightarrow \{b2\}$

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #2.

(ii) $B \rightarrow C$

As in above relation to test $B \rightarrow C$, $\{b1\} \rightarrow \{c1\}$, $\{b2\} \rightarrow \{c2\}$, $\{b3\} \rightarrow \{c4\}$ for all tuples.

Therefore, Dependency holds for all tuples in above table.

(iii) $C \rightarrow B$

As in above relation to test $A \rightarrow B$, $\{c1\} \rightarrow \{b1\}$ but in Tuple #3 $\{c1\} \rightarrow \{b4\}$

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #3.

(iv) $B \rightarrow A$

As in above relation to test $A \rightarrow B$, $\{b1\} \rightarrow \{10\}$ but in Tuple #5 $\{b1\} \rightarrow \{13\}$

Also {b3} → {13} but in Tuple #6 {b3} → {14}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #5 and Tuple #6.

(v) C→ A

As in above relation to test C→ A, In Tuple #1 {C1} → {10,11,13}

But in Tuple #3 {c1} → {11} and in Tuple #5 {c1} → {13}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #3 and Tuple #5

9.6 Types of Functional Dependencies

Q. What are types of functional dependencies? (5 Marks)

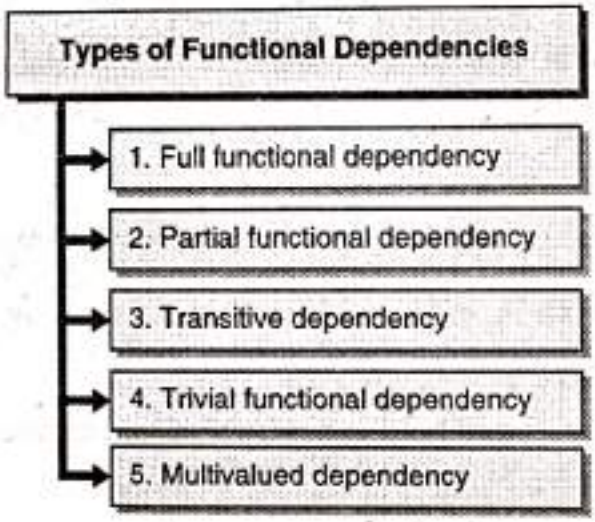


Fig. 9.6.1 : Types of Function Dependencies

(1) Full functional dependency

- A functional dependency is a fully functional dependency if removal of any attributes from determinant will invalidate the dependency.
- A functional dependency A → B is a fully functional dependency if removal of any attributes from A means that the dependency does not hold any more.

Example :

Consider an employee table with columns as shown in Table 9.6.1.

Table 9.6.1 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

- In the above example, Hours and allowance are fully functionally dependent on both Employee_Id and Project_Id.

Employee_Id, Project_Id → Hours, Allowance

- The number of hours spent on the project by a particular employee cannot be determined with the project number (Project_no) alone. It needs the employee number (Emp_no) as well.

(2) Partial functional dependency

- A partial dependency means that a non key column is depend on some columns in composite primary key of a table.
- An FD A → B is a partial dependency if there is some attribute X ∈ A (X subset of A), that can be removed from A and the dependency will still hold.
- If determine (attributes after arrow) attributes depends on **part of** (partial) determinant attributes. Such dependency is called as partial functional dependency.

Example :

- Consider an employee table with columns as shown in Table 9.6.2.

Table 9.6.2 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

- In the above example, attribute salary is considered to be functionally dependent on both Employee_Id and Project_Id.

Employee_Id, Project_Id → Salary

- But, Attribute salary functionally dependent on Employee_Id is also holds true.

Employee_Id → Salary

- So, Salary is partial functionally dependent on attribute pair Employee_Id and Project_Id.

(3) Transitive dependency

- If one attribute of relation is functionally dependent on other dependent attribute then such a dependency is called as **transitive dependency.**
- An FD X →Y in a relation R is a transitive dependency, if there is a set of attributes Z that is not a subset of any key of R, and both X → Z and Z → Y holds true.

Example :

- Consider an employee table schema,
- Employee_Id → Department_Id
- Department_Id → Dname

Table 9.6.3 : Employee Table

Employee_Id	Ename	Salary	Department_Id	Dname
10	Mahesh	50000	C1	IT
12	Suresh	25000	E2	HR
15	Ganesh	26000	C1	IT
18	Mahesh	50000	E2	HR

- Dependency of Department_Id on key attribute Dname is transitive functional dependency as Dname is depends on Department_Id which is depend on Employee_Id. So, Dname is transitive functionally dependent on Employee_Id.

(4) Trivial functional dependency

- Functional dependency (FD) X → Y and Y is a subset of X, then it is called as a trivial FD.
- Functional dependency X → Y and Y is not a subset of X, then it is called as a non-trivial functional dependency.

Example :

- For employee table the below given FD is trivial as Ename is a subset of {Employee_Id, Ename}

Employee_Id, Ename → Ename

- And below given FD is non-trivial as Hours is not a subset of {Employee_Id, Project_Id}

Employee_Id, Project_Id → Hours

(5) Multi valued dependency

- Multivalued dependency defined by $X \rightarrow\rightarrow Y$ is said to hold for a relation R (X, Y, Z) if for a given set of values of X, there is a set of associated values of attribute Y, and X values depend only on X values and have no dependence on the set of attributes Z.
- **Multivalued dependency** in turn, is defined as relationship which accepts the cross-product pattern.

Example :

- For employees car table as given below,

Table 9.6.4 : Employee_Car Table

Employee_Id	Ename	Car
10	Mahesh	Ertiga
12	Suresh	Zen
15	Ganesh	Sentro
10	Mahesh	Wagon R

- The FDs are given as below,

Employee_Id $\rightarrow\rightarrow$ Car

- $\alpha \rightarrow\rightarrow \beta$ says relationship between α and β independent of relationship between α and R-β. That means **Employee_Id $\rightarrow\rightarrow$ Car** relationship is independent of Employee_Id and EName relation.

i.e. **Employee_Id $\rightarrow\rightarrow$ Car** is independent of Employee_Id $\rightarrow\rightarrow$ Ename

9.7 FD Properties (Armstrong's Axioms / Closures of FD)

Q Give Armstrong axioms. **(2 Marks)**

Q. List the Armstrong's axioms for functional dependencies. What do you understand by soundness and completeness of these axioms? **(2 Marks)**

- Given that Relation R(X, Y, Z, W); represents a table R with set of indivisible attributes X, Y, Z and W.
- It is possible to derive many properties of functional dependencies.
- Axioms are nothing but rules of inference which provides a simple technique for reasoning about functional dependencies.

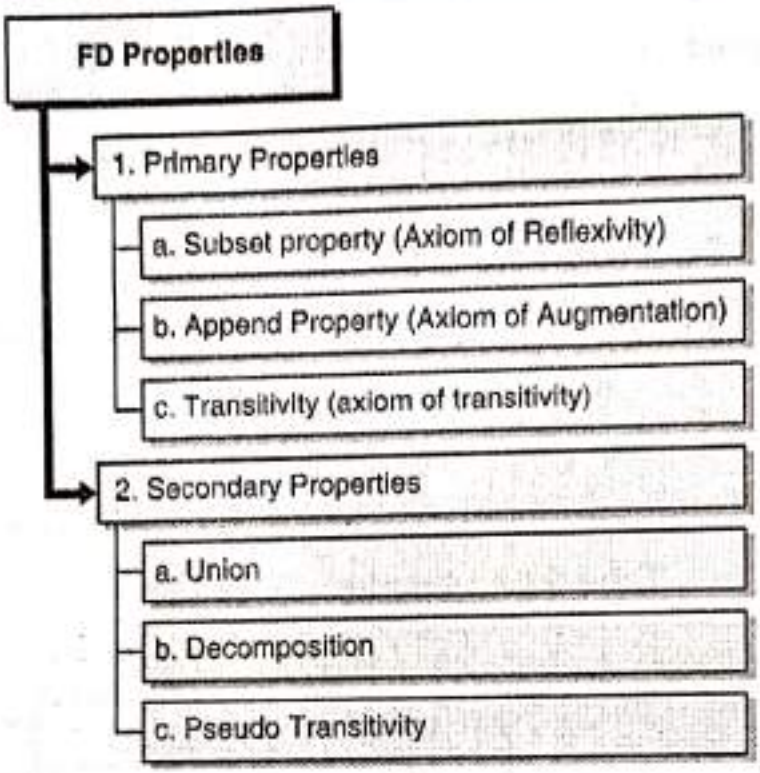
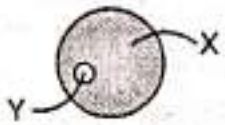


Fig. 9.7.1 : FD Properties

(1) Primary Properties

a. Subset property (Axiom of Reflexivity)



For given relation R(X, Y, Z, W),

If Y is a subset of X as shown in diagram,

Then $X \rightarrow Y$

(Which can be referred as X is functionally dependent on Y)

b. Append Property (Axiom of Augmentation)

For given relation R(X, Y, Z, W),

If $X \rightarrow Y$

Then $XZ \rightarrow YZ$

It is possible to append attribute Z to both sides of FD provided that it is part of same table.

c. Transitivity (axiom of transitivity)

For given relation R(X, Y, Z, W),

If $X \rightarrow Y$ and $Y \rightarrow Z$

Then $X \rightarrow Z$

It is possible to use transitivity if attribute X, Y and Z are part of the same table.

(2) Secondary Properties

a. Union

For given relation R(X, Y, Z, W),

If $X \rightarrow Y$ and $X \rightarrow Z$

Then $\mathbf{X \rightarrow YZ}$

b. Decomposition

For given relation R(X, Y, Z, W),

If $X \rightarrow YZ$

Then $X \rightarrow Y$ and $X \rightarrow Z$

c. Pseudo Transitivity

For given relation R(X, Y, Z, W),

If $X \rightarrow Y$ and $YZ \rightarrow W$

Then $\mathbf{XZ \rightarrow W}$

Example 9.7.1 : Consider relation R = (A, B, C, D, E, F) having set of FD's

$A \rightarrow B$ $A \rightarrow C$

$BC \rightarrow D$ $B \rightarrow E$

$BC \rightarrow F$ $AC \rightarrow F$

Calculate some members of Axioms as be below:

(i) $A \rightarrow E$ (ii) $BC \rightarrow DF$

(iii) $AC \rightarrow D$ (iv) $AC \rightarrow DF$ **(5 Marks)**

Solution :

(i) $A \rightarrow E$

As $A \rightarrow B$ and $B \rightarrow E$

So using Transitive rule,

$\therefore$ $\mathbf{A \rightarrow E}$

(ii) $BC \rightarrow DF$

As $BC \rightarrow D$ and $BC \rightarrow F$

So using union rule,

$\therefore$ **BC → DF**

(iii) **AC → D**

As A → B and BC → D

So using pseudo transitivity,

$\therefore$ **AC → D**

(iv) **AC → DF**

As AC → D and AC → F

So using union rule,

$\therefore$ **AC → DF**

Example 9.7.2 : Consider relation R = (A, B, C, D, E, F) having set of FD's

A → B A → C

C → D B → E

AC → F

Calculate some closures as $\{A\}^+$, $\{B\}^+$, $\{AC\}^+$ and also find key of above relation. **(5 Marks)**

Solution :

(i) A → B, A → C $\{A\}^+ = \{A, B, C\}$

So using union rule,

A → BC

(ii) B → E $\{B\}^+ = \{B, E\}$

(iii) $\{A\}^+ = \{A, B, C\}$ and $\{AC\}^+ = \{A, B, C, D, E, F\}$

C → D, B → E and AC → F

(iv) $\{AC\}^+ = \{A, B, C, D, E, F\}$ {AC} can determine all attributes in relation R

So, {AC} is a key of Relation R

9.8 Decomposition

Q. What is decomposition? Explain various rules of normalization. (5 Marks)

1. Introduction

(a) If a relation is not in the normal form and we wish the relation to be normalised so that some of the anomalies can be eliminated, it is necessary to decompose the relation in two or more relations.

(b) This is a process of dividing one table into multiple tables can be done using projection operator.

(b) Decomposed table can be reconstructed using join operation.

2. Desirable Properties or Goals

Q. What are goals of decomposition? **(3 Marks)**

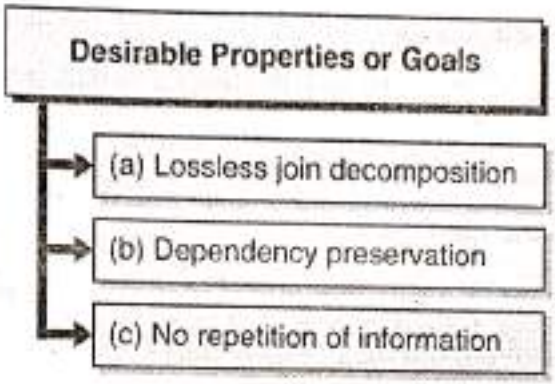


Fig. 9.8.1 : Desirable Properties of Goals

(a) Lossless join decomposition

- The original relation and relation reconstructed from joining decomposed relations must contain same number of tuples if number is increased or decreased then it is Losssy Join decomposition.
- Lossless join decomposition ensures that we can never get the situation where spurious tuple are generated in relation, for every value on the join attributes there will be a unique tuple in one of the relations.

Example :

- Employee (Employee_Id, Ename, Salary, Department_Id, Dname)
- Can be decomposed using lossless decomposition as,
- Employee_desc (Employee_Id, Ename, Salary, Department_Id)
- Department_desc (Department_Id, Dname)
- Alternatively the lossy decomposition would be as joining these tables is not possible so not possible to get back original data.
- Employee_desc (Employee_Id, Ename, Salary)
- Department_desc (Department_Id, Dname)

- Rules for lossless decompositions are,

  (i) The relations to be decomposed must have at least one common attribute in pair of relations.

  i.e. $R_1 \cap R_2 \neq \Phi$

  (ii) The attributes in common must be a key for one of the relation for decomposition to be lossless.

(b) Dependency preservation

- All functional dependencies result in just one relation.
- Dependency preservation is another important requirement since a dependency is a very important constraint on the database.
- As a result of any database updates, the database should not result in illegal relation being created. Hence, our design should allow us to check updates without natural joins.
- If $X \rightarrow Y$ holds then we know that the two (sets) attributes are closely related or functionally dependent and it would be useful if both attributes in the same relation so that the dependency can be checked easily.
- This can be done by maintaining functional dependency.

Example :

- Consider relation R(X, Y, Z, W) that has the following dependencies F,

  $X \rightarrow Y$

  $Y \rightarrow ZW$

- If we decompose the above relation into R1 (X, Y) and R2 (X, Z, W) the dependency $Y \rightarrow ZW$ is not preserved.
- But, If we decompose the above relation into R1 (X, Y) and R2 (Y, Z, W) the all dependencies are preserved.

(c) No repetition of information

- Decomposition that we have done should not suffer from any repetition of information problem.
- It is desirable not to have any redundancy in database.

9.9 Keys and Attributes in keys

Q. Write a note on : Candidate Key, Secondary Key and Super Key. (6 Marks)

(1) Introduction

- Normalization process will makes use of some types of keys to remove the redundancies present in data of relational tables.
- The column value that uniquely identifies a single record in a table called as Key of table.
- Any key consisting of a single attribute is called a **simple key,** while that consisting of a combination of attributes is called a **composite key**.
- The non-normalized relations may cause redundancy problems which leads to data anomalies.

(2) Super Key

- A **superkey** of a relation is a set of attributes $S \subseteq R$ with the property that no two tuples in relation will have same combination of attribute values in S.
- In other words, a relation will have unique set of attributes S.
- If we add additional attributes to above set of attributes S, the resulting combination would still uniquely identify a single record in a table. Such augmented keys are also called as **superkey**.

Example :

Consider a Relation R (A, B, C, D, E, F) with Functional dependencies,

$$A \rightarrow BC$$

$$AC \rightarrow DE$$

$$E \rightarrow F$$

To find out key of relation R,

$$\{A\}^+ = \{A, B, C, D, E, F\}$$

$$\{AC\}^+ = \{A, B, C, D, E, F\}$$

$$\{ACD\}^+ = \{A, B, C, D, E, F\}$$

So {A}, {AC}, {ACD} are all superkey.

Example :

- STUDENT (Id, Name, Class, Branch, Age, Address, Mobile)
- The ID is a key attribute of STUDENT table, so ID and Name can be a superkey.

(3) Candidates Key

- Superkey concept given above may contain unnecessary attributes to key so the concept of a superkey is not sufficient.
- A candidate key (CK) is a superkey with the minimal attribute from super key.
- A minimal (irreducible) superkey is called as candidate's key.
- A superkey that does not contain a subset of attributes that is itself a superkey.
- Minimum attributes of superkey by omitting unnecessary attributes of table which is sufficient for identifying entity (row/record) uniquely are called as **candidate keys**.
- Candidate key is also a potential primary key.

Example :

Consider a Relation R (A, B, C, D, E, F) with Functional dependencies,

$$A \rightarrow BC$$
$$AC \rightarrow DE$$
$$E \rightarrow F$$

To find out key of relation R,

$$\{A\}^+ = \{A, B, C, D, E, F\}$$
$$\{AC\}^+ = \{A, B, C, D, E, F\}$$
$$\{ACD\}^+ = \{A, B, C, D, E, F\}$$

So {A}, {AC}, {ACD} are all superkey and {A} is a candidate's key.

Example :

STUDENT (Id, Name, Class, Branch, Age, Address, Mobile)

The ID is a candidate key attribute of STUDENT table.

(4) Secondary Key

- A candidate key selected to uniquely identify all other attribute values in any given row.
- If a number of candidate keys in a relation schema then one is arbitrarily selected as **primary key** and other keys are called as **secondary keys.**
- Secondary key of a table is a column or combination of some columns used for data retrieval process.

Example :

Consider a Relation R (A, B, C, D, E, F) with Functional dependencies,

AB → C, BC → DE, E → AF

To find out key of relation R,

$\{AB\}^+ = \{A, B, C, D, E, F\}$

$\{BC\}^+ = \{A, B, C, D, E, F\}$

So {AB} and {BC} are all candidates key.

If {AB} is selected as primary key then {BC} is called as secondary key.

(5) Prime Attributes

- An attribute in a relation schema R is called as prime attribute, if it is a member of any of the candidate key present in a relation.
- If an attribute is not a member of any candidate key then it is called as nonprime attribute.

Example :

- Consider an Employee table with following FDs,

  Employee_Id → Ename, Salary

  Employee_Id, Project_Id → Hours, Allowance

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

In above table candidate key is {Employee_Id, Project_Id}

- **Prime Attributes :** Employee_Id, Project_Id
- **Non-Prime Attributes :** Ename, Salary, Hours, Allowance

9.10 First Normal Form (1NF)

Q. Explain 1NF. **(2 Marks)**
Q. Discuss 1 NF, 2 NF and 3 NF in Detail **MU - Dec. 18, 2 Marks**

(1) Introduction

- This is simplest form of normalization, simplifies each attribute in relation.
- This normal form given by E.F. Codd (1970) and the later version by C.J. Date (2003).

(2) Definition

1NF states that all attributes in relation must have atomic (indivisible) values and an attribute in a tuple must have a single value from the domain of that attribute.

- A relation is in 1NF, if every row contains exactly one value for each attribute.
- In short rules for data in 1NF is,
- A column in a table should contain only indivisible data.

(3) Example :

- Consider an employee table with columns as shown in diagram,
  - The relational schema not in 1 NF is represented as,

Employee Table

Employee_Id	Ename	Salary	Ecity

  - The state of Employee relational schema is as given below and it contains the Ecity which is non atomic (divisible) domain.

Table 9.10.1 (Non-Normalised)Employee Table

Employee_Id	Ename	Salary	Ecity
10	Mahesh	50000	Mumbai, Pune
12	Suresh	25000	Mumbai
15	Ganesh	26000	Pune
18	Kasturi	50000	Mumbai, Delhi

  - To convert relational schema in 1NF, the Ecity attribute is divided in atomic domains it may introduce some data redundancy.

Table 9.10.2 1NF Employee Table

Employee_Id	Ename	Salary	Ecity
10	Mahesh	50000	Mumbai
10	Mahesh	50000	Pune
12	Suresh	25000	Mumbai
15	Ganesh	26000	Pune
18	Kasturi	50000	Mumbai
18	Kasturi	50000	Delhi

(4) Minimizing Domain Redundancy

- The first normal for will solve the group redundancy occurs in domain value as it allows only a single value from the domain of that attribute.
- So, 1NF will solve all problems related to domain redundancy.
- Nested relations must be removed to convert relation in 1 NF.

9.11 Second Normal Form (2NF)

Q Explain 2NF. (2 Marks)
Q. Discuss 2 NF in Detail MU - Dec. 18, 3 Marks

(1) Introduction

- This normal form makes use of full functional dependency and tries to remove problem of redundant data introduced by 1NF decomposition.
- This normal form is given by E.F. Codd in 1971.

(2) Definition

- A relation is in 2NF, if it is in 1NF and all non-key attributes in relation are fully functionally dependent on the primary key of the relation.

OR

- A relation is in 2NF, if it is in 1NF and every non-key attribute is fully functionally dependent on the complete primary key of relation (and not depends on part of (partial) primary key).
- In short 2NF means,
- It should be in 1NF.
- There should not be any partial dependency on primary key attributes.

(3) Example :

- Consider an employee table with columns as shown in diagram,
- The relational schema **not in 2 NF** is represented as,

Consider an Employee table with following FDs,

Employee_Id → Ename, Salary

Employee_Id, Project_Id → Hours, Allowance

As

{Employee_Id, Project_Id} → Ename, Salary, Hours, Allowance

Therefore,

Candidate key {Employee_Id, Project_Id} is selected as primary key.

- As attributes Hours, Allowance of employee table are full functionally dependent on primary key whereas attributes Ename and Salary are partially depends on primary key. (As Ename, Salary are depends **on part of primary key)**
- The state of Employee relational schema is,

Table 9.11.1 : Non-2NF Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000
15	Ganesh	26000	E001	24	20000
18	Mahesh	50000	B056	11	10000

- To normalize above schema to 2NF we can decompose tables as,

Employee (Employee_Id, Ename, Salary)

Employee_Id → Ename, Salary

Table 9.11.2 : 2NF Employee Table

Employee_Id	Ename	Salary
10	Mahesh	50000
12	Suresh	25000
15	Ganesh	26000
18	Mahesh	50000

Project (Employee_Id, Project_Id, Hours, Allowance)

Employee_Id, Project_Id → Hours, Allowance

Table 9.11.3 : 2NF Project Table

Employee_Id	Project_Id	Hours	Allowance
10	E001	44	40000
12	B056	31	30000
15	C671	23	20000
18	E002	12	15000
15	E001	24	20000
18	B056	11	10000

- Consider, Relation R(A, B, C, D, E, F) and the FDs as below,

  A → BC, B → DC, D → EF

  (i) The candidate Key is {AD} → {A, D, B, C, E, F} selected as primary key.

  All attributes are partially dependent on primary key.

  Hence, Relation R is not in 2NF.

  (ii) The 2NF Relation Schema is,

  R1 (A, B, C, D) with FDs A → BC, B → DC

  R2 (D, E, F) with FDs D → EF

(3) Minimizing Tuple Redundancy

- The second normal form will avoid same tuples to be repeated in a table as it forces all non-key attributes must be full functionally depends on primary key of a relation.
- 2NF will create a new table for each partial key with all its dependent attributes.

9.12 Third Normal Form (3NF)

Q. Explain 3NF. (2 Marks)

Q. Discuss 3 NF in Detail MU - Dec. 18, 3 Marks

(1) Introduction

- This normal form is given by E.F. Codd in 1971.
- This normal form introduced to minimize the transitive redundancy.
- To remove the data anomalies left in relational schema even after applying second normal form like transitive dependencies.

(2) Definition

- A relation is in 3NF, if it is in 2NF and all non-prime attributes of the relation are non-transitively dependent on the every key.
- A relation R is in 3NF if all non prime attributes are,
  1. Full functionally dependent on primary key.
  2. Non-transitive dependent on every key.
- A relational schema R is in 3NF, if non-trivial functional dependency X→A holds true where X is a superkey and A is a prime attribute.

(3) Example :

- Consider an employee table with columns as shown in diagram,
- The relational schema **not in 2 NF** is represented as,
- Consider an Employee table with following FDs,

  Employee_Id → Ename, Salary, Department_Id

  Department_Id → Dname
- The state of Employee relational is,

Table 9.12.1 : Employee Table

Employee_Id	Ename	Salary	Department_Id	Dname
10	Mahesh	50000	C1	IT
12	Suresh	25000	E2	HR
15	Ganesh	26000	C1	IT
18	Mahesh	50000	E2	HR

{Employee_Id} → Ename, Salary, Department_Id, Dname

Therefore,

Candidate key {Employee_Id} is selected as primary key.

- As all attributes in employee table are full functionally dependent on primary key. Therefore, **Relation R is in 2NF.**
- Non-prime attributes Ename, Salary, Department_Id are non-transitively dependent on primary key. But, Dname attribute is transitively dependent on key. Therefore, **Relation R is not in 3NF.**

  Employee_Id → Department_Id

  Department_Id → Dname

- To normalize above schema to 3NF we can decompose tables as,

Employee (Employee_Id, Ename, Salary, Department_Id)

Employee_Id → Ename, Salary, Department_Id

Table 9.12.2 : 3NF Employee Table

Employee_Id	Ename	Salary	Department_id
10	Mahesh	50000	C1
12	Suresh	25000	E2
15	Ganesh	26000	C1
18	Mahesh	50000	E2

Department (Department_Id, Dname)

Employee_Id → Dname

Table 9.12.3 : 3NF Department Table

Department_Id	Dname
C1	IT
E2	HR

- Consider, Relation R(A, B, C, D, E, F) and the FDs as below,

A → BC, B → D, D → EF

(i) The candidate Key is {A} → {A, D, B, C, E, F} selected as primary key.

- All attributes are full functionally dependent on primary key.
- Hence, Relation R is **in 2NF**.
- But, non-prime attributes B, D, E and F are transitively depends on key.
- So, Relation R is **not in 3NF**.

(ii) The 3NF Relation Schema is,

- R1 (A, B, C) with FDs A → BC
- R2 (B, D) with FDs B → D
- R3 (D, E, F) with FDs D → EF

Note : In most cases, third normal form is the sufficient level of decomposition. But some case requires the design to be further normalized up to the level of 4th as well as 5th. These are based on the concept of Multi Valued Dependency.

(4) Minimizing Group Redundancy

- The third normal form will avoid repeating groups in same table as it forces all non-prime attributes must be non-transitively depends on key of a relation.
- 3NF will create a new table for each transitive attribute and its dependent attributes.

Table 9.12.4 : Summary of normal form based on primary keys and corresponding normalization

Normal form	Test	Remedy (normalization)
First (1NF)	Relation should have to multivalued attributes or nested relations.	Form new relations for each multivalued attributed to nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attributed should be functionally.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original promarykey and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should bot have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determinte(s) other nonkey attribute(s).

9.13 Boyce-Codd Normal Form (BCNF)

Q. Describe BCNF in detail. (5 Marks)

(1) Introduction

- This normal form is governed by Raymond F. Boyce and E.F. Codd in 1974.
- BCNF is more rigorous form of 3NF and every relation in BCNF is always in 3NF.
- The intention of Boyce-Codd Normal Form (BCNF) is that 3NF does not satisfactorily handle the case of overlapping candidate keys.
- If transitivity is present in prime attributes of relation may not be removed by 3NF.

(2) Definition

- A relation R is said to be in BCNF, if and only if every determinant is a candidate key.

- A relational schema is in BCNF, if a non-trivial functional dependency X→A is true then X is a superkey of relation R.
- In 3NF definition A should be prime attribute, which is not the case in BCNF definition.

(3) **Example :**

- Consider an employee table in which employee can work in more than one department,
- The relational schema **not in 2 NF** is represented as,
- Consider an Employee table with following FDs,

  Employee_Id → Ename, Salary, Department_Id

  Department_Id → Dname
- The state of Employee relational is,

Table 9.13.1 : Employee Table

Employee_Id	Ename	Department_Id	Dname	Dtype
10	Mahesh	C1	IT	Technical
12	Ganesh	E2	HR	Skill
12	Ganesh	C1	IT	Technical
10	Mahesh	E2	HR	Skill
13	Satish	E1	TS	Technical

Employee_Id → Ename

Department_Id → Dname, Dtype

Therefore,

- Candidate key {Employee_Id, Department_Id} is selected as primary key.
- As no attribute in employee table is full functionally dependent on primary key. Therefore, **Relation R is not in 2NF.** All non-prime attribute are non-transitively dependent on primary key. Therefore, **Relation R is in 3NF.**
- To normalize above schema to **BCNF** we can decompose tables as,

  Employee (Employee_Id, Ename)

  Employee_Id → Ename
- The determinant Employee_Id is candidate key.

Table 9.13.2 : 3NF Employee Table

Employee_Id	Ename
10	Mahesh
12	Ganesh
13	Satish

Department (Department_Id, Dname, Dtype)

Department_Id → Dname, Dtype

- The determinant Department_Id is candidate key.

Table 9.13.3 : 3NF Department Table

Department_Id	Dname	Dtype
C1	IT	Technical
E2	HR	Skill
E1	TS	Technical

- **Emp_Dept** (Employee_Id, Department_Id)

Table 9.13.4 : 3NF Emp Dept Table

Employee_Id	Department_Id
10	C1
12	E2
12	C1
10	E2
13	E1

(4) Minimizing Key Transitivity (Key redundancy)

- The BCNF will produces a table for each functional dependency to make all determinants as key of relation.
- BCNF will create a new table for each transitive attribute and its dependent attributes.

Example 9.13.1 : Consider the following relation,

CAR-SALE (Car#, Date-sold, Salesman#, commission%, Discount-amt)

Assume that {Car#, Salesman#} is the primary key.

Additional dependencies are,

Date-sold → Discount_amt

Salesman# → commission%

Based on the given primary key, is this relation in 1NF, 2NF or 3NF ?
Why or why not ?
How would you successively normalize it completely ? **(10 Marks)**

Solution :

CAR-SALE (Car#, Salesman#, Date-sold, commission%, Discount-amt)

Assuming {Car#, Salesman#} is the primary key

Therefore,

Car#, Salesman# → Date-sold, commission%, Discount-amt

Additionally,

Date-sold → Discount_amt

Salesman# → commission%

- The above relation **is in 1NF as** all attributes of relation are atomic domains.
- The above relation **is in 2NF as** primary key is assumed and all non key attributes are full functionally depends on primary key.
- The above relation **is in 3NF as** all non prime attributes are non-transitively depending on primary key.
- Normalized Relation

  CAR-SALE (Car#, Salesman#, Date-sold, commission%, Discount-amt)

  Car#, Salesman# → Date-sold, commission%, Discount-amt

  Date-sold → Discount_amt

  Salesman# → commission%

9.14 Converting Relational Schema to higher normal forms

Example 9.14.1 : Relation R(A, B, C, D, E, F, G, H, I, J). Having following set of FD, show convert table to highest normal form.

AB → C, C → EF, AD → GH, G → I, H → J **(10 Marks)**

Solution :

(a) 1NF

Assuming all attributes are atomic domains so relation R is in 1NF.

(b) 2 NF

$\{A, B, D\}^+ \rightarrow \{\underline{A, B, D}, C, E, F, G, H, I, J\}$

$\{A, B, D\}$ is candidate key for relation R.

No attribute in relation is full functionally dependent on above key.

Therefore, Relation R is not in 2NF.

Decomposition to 2NF

R_1 ($\underline{A, B}$, C, E, F); $AB \rightarrow C$, $C \rightarrow EF$

R_2 ($\underline{A, D}$, C, E, F); $AD \rightarrow GH$, $G \rightarrow I$, $H \rightarrow J$

(c) 3NF

In relation R1, non-prime attributes E, F are transitively depends on key. So, relation is not in 3NF.

The relation R1 in 3NF can be written as,

R_{1a} ($\underline{A, B}$, C); $AB \rightarrow C$

R_{1a} ($\underline{C}$, E, F); $C \rightarrow EF$

In relation R2, non-prime attributes I, J are transitively depends on key. So, relation is not in 3NF.

The relation R1 in 3NF can be written as,

R_{2a} ($\underline{A, D}$, G, H); $AD \rightarrow GH$

R_{2b} ($\underline{G}$, I); $G \rightarrow I$

R_{2c} ($\underline{H}$, J); $H \rightarrow J$

(c) BCNF

Relation	FDs	Determinant	Key	BCNF?
R_{1a} ($\underline{A, B}$, C)	$AB \rightarrow C$	AB	AB	Yes
R_{1a} ($\underline{C}$, E, F)	$C \rightarrow EF$	C	C	Yes
R_{2a} ($\underline{A, D}$, G, H)	$AD \rightarrow GH$	AD	AD	Yes
R_{2b} ($\underline{G}$, I)	$G \rightarrow I$	G	G	Yes
R_{2c} ($\underline{H}$, J)	$H \rightarrow J$	H	H	Yes

So, relational schema in BCNF is as given below,

R_{1a} ($\underline{A, B}$, C); $AB \rightarrow C$

R_{1a} ($\underline{C}$, E, F); $C \rightarrow EF$

R_{2a} (A, D, G, H); AD → GH

R_{2b} (G, I); G → I

R_{2c} (H, J); H → J

Example 9.14.2 : We are given Relation R with Attributes A, B, C, D, E, F and the FDs as below, Find and explain which Armstrong's Axioms can be applied here to find closure,

A → BC, B → E, CD → EF **(10 Marks)**

Solution :

1. $\{A\}^+$

(a) Axiom of Pseudo transitivity

A → BC and B → E ...(1)

∴ A → EC ...(2)

(c) Decomposition from Equations (1) and (2)

A → B, A → C, A → E

∴ $\{A\}^+ = \{A, B, C, E\}$

$\{A\}^+ = \{A, B, C, E\}$

2. $\{A, B\} = \{A, B, C, E\}$
3. $\{C, D\} = \{C, D, E, F\}$
4. $\{A, D\} = \{A, B, C, D, E, F\}$

9.15 Solved Examples on Normalization

Example 9.15.1 : Consider a dependency diagram of relation R and normalize it up to third normal form. **(10 Marks)**

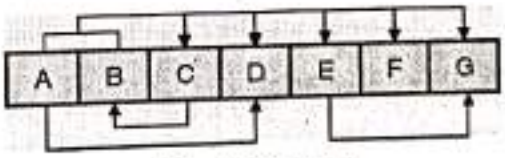


Fig. P. 9.15.1

Solution :

Normalized Relation R (A, B, C, D, E, F, G) with set of FDs

AB → CDEFG

C → B

A → D

E → G

Example 9.15.2 : Consider a dependency diagram of relation R and normalize it upto third normal form.

(10 Marks)

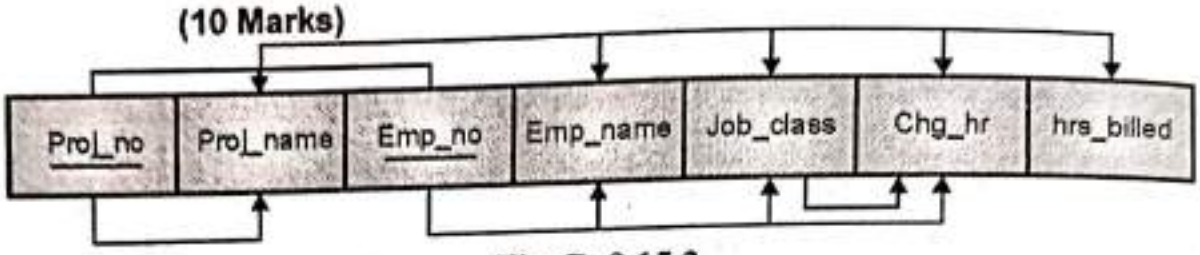


Fig. P. 9.15.2

Solution :

- Normalized Relation,
- Employees (Proj_no, Emp_no, Proj_name, Emp_name, Job_Class, Chg_Hr, Hrs_Billed)
- With set of FDs

  Proj_no, Emp_no → Proj_name, Emp_name, Job_Class, Chg_Hr, Hrs_Billed

  Emp_no → Emp_name, Job_Class, Chg_Hr

  Proj_no → Proj_name

  Job_Class → Chg_Hr

Example 9.15.3 : Consider the following dependency diagram of relation R and normalize til 3NF form.

(10 Marks)

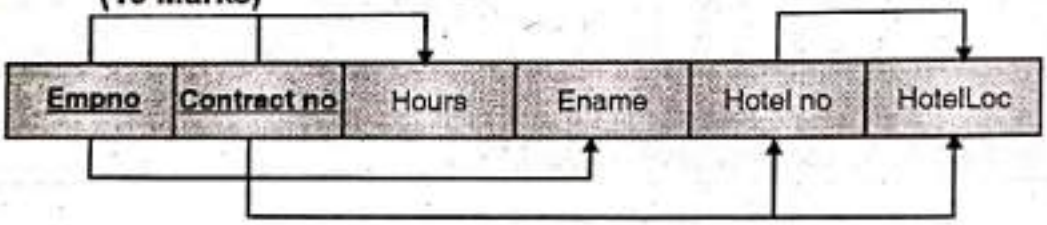


Fig. P. 9.15.3

Solution :

Converting dependency diagram to dependencies,

Empno,Contactno -> Hours

Empno ->Ename

Contactno ->Hotelno,HotelLoc

Hotelno->HotelLoc

3NF schema is as given below,

Emp(Empno,Contactno,Hours); Empno,Contactno -> Hours

Emp_Details(Empno,Ename); Empno ->Ename

Hotel_Contact(Contactno,Hotelno,HotelLoc); Contactno ->Hotelno,HotelLoc

Hotel(Hotelno,HotelLoc); Hotelno ->HotelLoc

9.16 Fourth Normal Form (BCNF)

1. Introduction

- This Normal form is given by Ronald Fagin (1977).
- Fourth Normal Form tries to remove multi valued dependency among attributes.

2. Definition

A relation is said to be in **fourth normal form** if each table contains no more than one multivalued dependency per key attribute.

3. Example :

Seminar	Faculty	Topic
DBP-1	Brown	Database Principles
DAT-2	Brown	Database Advanced Techniques
DBP-1	Brown	Data Modeling Techniques
DBP-1	Robert	Database Principles
DBP-1	Robert	Data Modeling Techniques
DAT-2	Maria	Database Advanced Techniques

- In the above example, same topic is being taught in a seminar by more than 1 faculty and each Faculty takes up different topics in the same seminar.
- Hence, Topic names are being repeated several times.
- This is an example of multivalued dependency. (No multivalued dependency)

Seminar	Topic
DBP-1	Database Principles
DAT-2	Database Advanced Techniques
DBP-1	Data Modeling Techniques

- To eliminate multivalued dependency, split the table such that there is no multivalued dependency. (One multivalued dependency).

Seminar	Faculty
DBP-1	Brown
DAT-2	Brown
DBP-1	Robert
DAT-2	Maria

Review Questions

Q. 1 Write a short note : Functional dependency

Q. 2 What is normalizations ? What is its importance in DBMS design ? Explain 1NF, 2NF, 3NF and BCNF with suitable example.

Q. 3 Describe data redundancy in relational schema.

Q. 4 Distinguish between functional dependency and multivalued dependency.

Q. 5 List the Armstrong's axioms for functional dependencies ?

Q. 6 Why BCNF is more desirable than 3 NF ?

Q. 7 State and compare 3 NF and BCNF.

□□□

CHAPTER 10

Transaction

Module VI

Syllabus

Transaction concept, Transaction states, ACID properties , Concurrent Executions, Serializability – Conflict and View.

10.1 Concept of Transaction

Q.	What is transaction? Explain concept of transaction using example.	**(4 Marks)**
Q.	List types of operations included in transaction.	**(4 Marks)**
Q.	Give syntax for transaction and explain its structure.	**(4 Marks)**

(1) Introduction

- Single SQL command is sent to database server as a query and server will reply with answer.
- Multiple SQL commands (DML,DRL etc.) are sent to database server which executed one after other (as shown in Fig. 10.1.1),

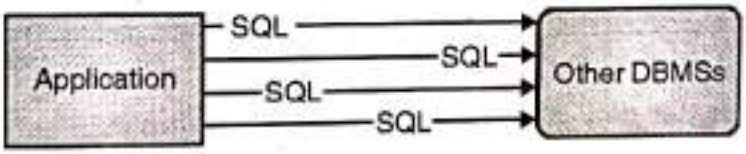


Fig. 10.1.1 : Executing single operation in DBMS

- In place of sending one by one SQL command to server we can combine multiple operations that are logically similar and send to serve as a single logical unit called **transaction.**

Example : Transferring Rs.100 from one account to other

1. Withdraw Rs.100 from account_1
2. Deposit Rs.100 to account_2

- Simple query fired on DBMS is called **SQL operation.**
- Collection of multiple operations that forms a single logical unit is called as **transaction.**

- A transaction is a sequence of one or more SQL statements that combined together to form a single logical unit of work.

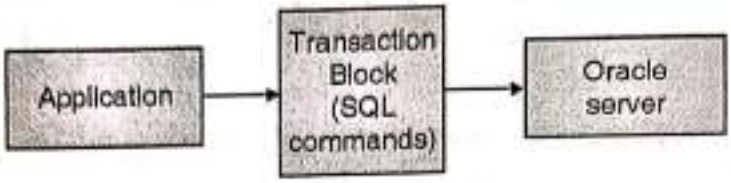


Fig. 10.1.2 : Executing transaction in DBMS

- Types of operations that can be done inside transaction,

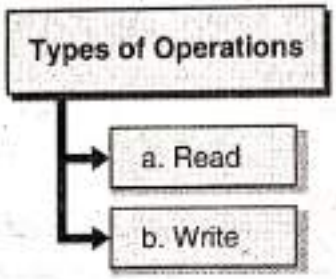


Fig. 10.1.3 : Types of operations

a. Read operation

- Read operation transfers data item from (e.g. table) the database memory to a local buffe of the transaction that executed the read operation.

Example : Data selection / retrieval language

```
SELECT    *
FROM      Students;
```

b. Write operation

- Write operation transfer data from local memory to database.
- Write operation transfers one data item from the local buffer of the transaction tha executed the write back to the database.

Example : Data Manipulation Language (DML)

```
UPDATE    Student
SET       Name = 'Bhavna'
Where  Sid  = 1186;
```

- Information processing in DBMS divide operation individual, indivisible operational logical units, called transactions.
- A transaction is a sequence of small database operations.
- Transactions will execute to complete all set of operations successfully.

Example : During the transfer of money between two bank accounts it is problematic if one of transaction fails.

```
BEGIN TRANSACTION transfer
    UPDATE accounts
    SET balance=balance – 100
    WHERE account=A

    UPDATE accounts
    SET balance=balance + 100
    WHERE account=B
If no errors then
        Commit Transaction
Else
        Rollback Transaction
End If
END TRANSACTION transfer
```

Fig. 10.1.4 : Sample implementation of transaction in SQL

- Transaction structure and boundaries :

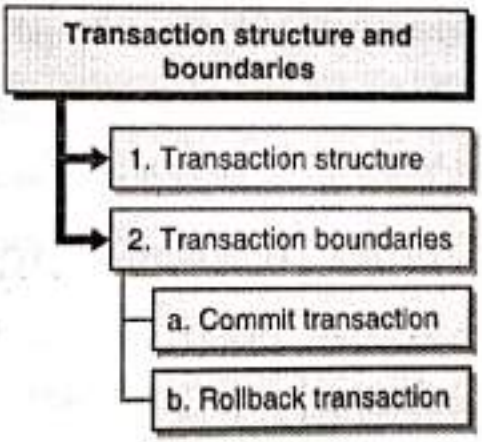


Fig. 10.1.5 : Transaction structure and boundaries

1. **Transaction structure**

- The transaction consists of all SQL operations executed between the begin transaction and end transaction.

- A transaction starts with a BEGIN transaction command.
- BEGIN command instructs transaction monitor to start monitoring the transaction status.
- All operations done after a BEGIN command is treated as a single large operation.

2. Transaction boundaries

- Transaction must ends either by executing a COMMIT or ROLLBACK command.
- Until a transaction commits or rolls back the data in database remains unchanged.

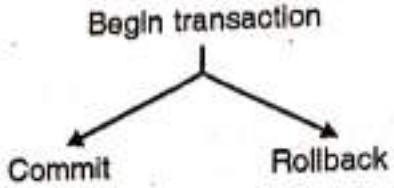


Fig.10.1.6

a. Commit transaction

All successful transactions are required to be committed by issuing commit command. To make all changes permanent and available to other users of the database.

b. Rollback transaction

- If transaction is unsuccessful due to some error then it must be rolled back.
- Roll back command will remove all changes to the database which are undone and the database remains unchanged by effect of transaction.
- The DBMS should take care of transaction it should be either complete or fail.
- When a transaction fails all its operations and the database is returned to the original state it was in before the transaction started.

10.2 Fundamental Properties of Transaction / ACID Properties

Q. What are ACID properties ? (4 Marks)
Q Explain ACID Properties of transaction MU - Dec. 18, 5 Marks

To understand transaction properties, we consider a transaction of transferring 100 rupees from account A to account B as below.

Let T_i be a transaction that transfers 100 from account A to account B. This transaction can be defined as,

(a) Read balance of account A.

(b) Withdraw 100 rupees from account A and write back result of balance update.

(c) Read balance of account B.

(d) Deposit 100 rupees to account B and write back result of balance update.

T_1	Read (A); A := A − 100; Write(A); Read(B); B := B + 100; Write(B)

Fig. 10.2.1 : Sample transaction

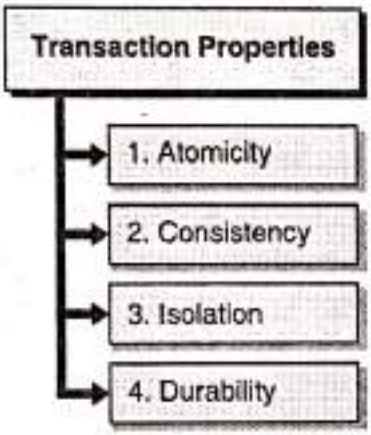


Fig. 10.2.2 : Transaction Properties

1. Atomicity

- Transaction must be treated as a single unit of operation.
- That means when a sequence of operations is performed in a single transaction, they are treated as a single large operation.

Examples :

(a) Withdrawing money from your account.

(b) Making an airline reservation.

- The term atomic means thing that cannot be divided in parts as in atomic physics.
- Execution of a transaction should be either complete or nothing should be executed at all.
- No partial transaction executions are allowed. (No half done transactions)

Example 1 : Money transfer in above example

Suppose some type of failure occurs after Write (A) but before Write (B) then system may lose 100 rupees in calculation which may cause error as sum of original balance(A+B) in accounts A and B is not preserved (such situation is called as inconsistency explained later), In such case database should automatically restore original value of data items.

Example 2 : Making an airline reservation

(a) Check availability of seats in desired flight.

(b) Airline confirms your reservation

(c) Reduces number of available seats

(d) Charges your credit card (deduct amount from your balance)

(e) Increases number of meals loaded on flight (Sometimes)

- In above case either all above changes are made to database or nothing should be done as half-done transaction may leave data as incomplete state.
- If one part of the transaction fails, the entire transaction fails and the database state is left unchanged.

2. Consistency

- Consistent state is a database state in which all valid data will be written to the database.
- If a transaction violates some consistency rules, the whole transaction will be rolled back and the database will be restored to its previous consistent state with those rules.
- On the other hand, if a transaction is executed successfully then it will take the database from one consistent state to another consistent state.
- DBMS should handle an inconsistency and also ensure that the database is clean at the end of each transaction.
- Consistency means transaction will never leave your database in a half finished (inconsistent) state.
- If one part of the transaction fails, all of the pending changes made by that transaction are rolled back.

Example : Money transfer in above example

Initially in total balance in accounts A is 1000 and B is 5000 so sum of balance in both accounts is 6000 and while carrying out above transaction some type of failure occurs after Write (A) but before Write (B) then system may lose 100 rupees in calculation.

As now sum of balance in both accounts is 5900 (which should be 6000) which is not a consistent result which introduces inconsistency in database.

This means that during a transaction the database may not be consistent.

3. Isolation

- Isolation property ensures that each transaction must remain unaware of other concurrently executing transactions.
- Isolation property keeps multiple transactions separated from each other until they are completed.
- Operations occurring in a transaction (example, insert or update statements) are invisible to other transactions until the transaction commits (on transaction success) or rolls back (on transaction fails).
- For example, when a transaction changes a bank account balance other transactions cannot see the new balance until the transaction commits.
- Different isolation levels can be set to modify this default behaviour.
- Transaction isolation is generally configurable in a variety of modes. For example, in one mode, a transaction locks until the other transaction finishes.
- Even though many transactions may execute concurrently in the system. System must guarantees that, for every transactions (T_i) all other transactions has finished before transactions (T_i) started, or other transactions are started execution after transactions (T_i) finished.
- That means, each transaction is unaware of other transactions executing in the system simultaneously.

Example : Money transfer in above example.

The database is temporarily inconsistent while above transaction is executing, with the deducted total written to A and the increased total is not written to account B. If some other concurrently running transaction reads balance of account A and B at this intermediate point and computes A+B, it will observe an inconsistent value (Rs. 5900). If that other transaction wants to perform updates on accounts A and B based on the inconsistent values (Rs. 5900) that it read, the database may be left database in an inconsistent state even after both transactions have completed.

A way to avoid the problem of concurrently executing transactions is to execute one transaction at a time.

4. Durability

- The results of a transaction that has been successfully committed to the database will remain unchanged even after database fails.

- Changes made during a transaction are permanent once the transaction commits.
- Even if the database server is fails in the between transaction, it will return to a consistent state when it is restarted.
- The database handles durability by transaction log.
- Once the execution of the above transaction completes successfully, and the user will be notified that the transfer of amount has taken place, if there is no system failure in this transfer of funds.
- The durability property guarantees that, all the changes made by transaction on the database will be available permanently, although there is any type of failure after the transaction completes execution.

10.3 Transaction States

Q. Explain transaction state diagram. (4 Marks)

(1) Introduction

- If transaction complete successfully it may be saved on to database server called as **committed** transaction.
- A committed transaction transfers database from one consistent state to other consistent state, which must persist even if there is a system failure.
- Transaction may not always complete its execution successfully. Such a transaction must be **aborted**.
- An aborted transaction must not have any change that the aborted transaction made to the database. Such changes must be undone or **rolled back**.
- Once a transaction is committed, we cannot undo its effects by aborting it.

(2) Transaction states

- A transaction must be in one of the following states :

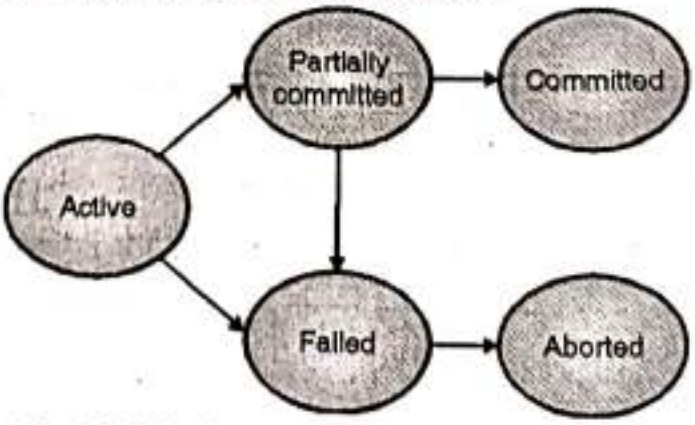


Fig. 10.3.1 : State diagram of a transaction

(a) Active

- This is initial state of transaction execution.
- As soon as transaction execution starts it is in active state.
- Transaction remains in this state till transaction finishes.

(b) Partially committed

- As soon as last operation in transaction is executed transaction goes to partially committed state.
- At this condition the transaction has completed its execution and ready to commit on database server. But, it is still possible that it may be aborted, as the actual output may be there in main memory, and thus a hardware failure may prohibit its successful completion.

(c) Failed

A transaction enters the failed state after the system determines that the transaction can no longer proceed with its normal execution.

Example : In case of Hardware or logical errors occurs while execution.

(d) Aborted

- Failed transaction must be rolled back. Then, it enters the aborted state.
- Transaction has been rolled back restoring into prior state.
- In this stage system have two options :
  - **Restart the transaction :** A restarted transaction is considered to be a new transaction which may recover from possible failure.
  - **Kill the transaction :** Because the bad input or because the desired data were not present in the database an error can occurs. In this case we can kill transaction to recover from failure.

(e) Committed

- When last data item is written out, the transaction enters into the committed state.
- This state occurs after successful completion of transaction.
- A transaction is said to have terminated if has either committed or aborted.

10.4 Transactions Schedules

Q. What are transaction schedules ? (2 Marks)

(1) Introduction

- Schedule is a sequence of instructions that specify the sequential order in which instructions of transactions are executed.
- A schedule for a set of transactions must consist of all instructions present in that transactions, it must save the order in which the instructions appear in each individual transaction.
- A transaction that successfully completes its execution will have to commit all instructions executed by it at the end of execution.
- A transaction that fails to successfully complete its execution will have to abort all instructions executed by transaction at the end of execution.

(2) Representation

- We denote this transaction T as,

  $R_T(X)$: Denotes read operation performed by the transaction T on object X.

  $W_T(X)$: Denotes write operation performed by the transaction T on object X.

- Each transaction must specify its final action as commit or abort.

10.5 Serial Executions / Transactions / Schedules

Q. Explain serial transaction with example. (4 Marks)

(1) Introduction

- This is simple model in which transactions executed in a serial order that means after finishing first transaction second transaction starts its execution.
- This approach specifies first my transaction, then your transaction other transactions should not see preliminary results.

(2) Example :

Consider below two transactions T_1 and T_2.

Transaction T_1 : deposits Rs.100 to both accounts A and B.

Transaction T_2 : doubles the balance of accounts A and B.

T_1:	Read(A)
	A ← A + 100
	Write(A)
	Read(B)
	B ← B + 100
	Write(B)

T_2:	Read(A)
	A ← A*2
	Write(A)
	Read(B)
	B ← B*2
	Write(B)

Serial schedule for above transaction can be represented as below,

Schedule A : A consistent serial schedule

A consistent serial schedule is obtained by executing T_1 right after T_2.

T_1	T_2	A	B	Operations
		25	25	Initial Balance
	Read(A); A ← A*2			
	Write(A)	50	25	
	Read(B); B ← B*2			
	Write(B)	50	50	
Read(A); A ← A + 100				
Write(A)		150	50	
Read(B); B ← B + 100				
Write(B)		150	150	
		150	150	Final Balance

In above Serial schedule for we can first execute transaction T_2 then T_1 which may results in some final values.

Representation : $T_1 \rightarrow T_2$

Schedule B : A consistent serial schedule

A serial schedule that is also consistent is obtained by executing T_2 right after T_1.

T_1	T_2	A	B	Operations
		25	25	Initial Balance
Read(A); A ← A + 100				
Write(A)		125	25	
Read(B); B ← B + 100				

T_1	T_2	A	B	Operations
Write(B)	Read(A); A ← A*2	25	125	
	Write(A) Read(B); B ← B*2	250	125	
	Write(B)	250	250	
		250	250	Final Balance

Representation : $T_2 \rightarrow T_1$

In above schedule T_1 is executed entirely then T_2 has started. Assume account A with Rs.25 and B with Rs.25. Then transaction T_1 will update A as 125 and B as 125. Now T_2 will read updated values of A and B. T_2 will update value of A as 250 and B as 250. The consistency constraint is A + B should remain unchanged. So at end of T_2, A + B.

i.e. 250 + 250 = 500 remains unchanged so execution of this schedule keeps database in consistent state.

10.6 Concurrent Executions / Transactions / Schedules

Q. Explain concurrent execution and give example. **(4 Marks)**
Q. What are advantages of concurrency ? **(4 Marks)**

(1) Introduction

- Transactions executed concurrently, that means operating system executes one transaction for some time then context switches to second transaction and so on.
- Transaction processing can allows multiple transactions to be executed simultaneously on database server.
- Allowing multiple transactions to change data in database concurrently causes several complications with consistency of the data in database.
- It was very simple to maintain consistency in case of serial execution as compare to concurrent execution of transactions.

(2) Advantages of concurrency

(i) Improved throughput

- Throughput of transaction is defined as the number of transactions executed in a given amount of time.

- If we are executing multiple transactions simultaneously that may increase throughput considerably.

(ii) Resource utilization

- Resource utilization defined as the processor and disk performing useful work or not (in ideal state).
- The processor and disk **utilization** increase as number of concurrent transactions increases.

(iii) Reduced waiting time

- There may be some small transaction and some long transactions may be executing on a system.
- If transactions are running serially, a short transaction may have to wait for a earlier long transaction to complete, which can lead to random delays in running a transaction.

(3) Example :

Consider below two transactions T_1 and T_2,

Transaction T_1 : deposits Rs.100 to both accounts A and B.

T_1 :

Read(A)
A ← A + 100
Write(A)
Read(B)
B ← B + 100
Write(B)

Transaction T_2 : doubles the balance of accounts A and B.

T_2 :

Read(A)
A ← A*2
Write(A)
Read(B)
B ← B*2
Write(B)

Above transaction can be executed concurrently as below,

Schedule C : A schedule that is not serial but is still consistent

Obtained by interleaving the actions of T_1 with those of T_2

Table 10.6.1

T_1	T_2	A	B	Operations
		25	25	Initial Balance
Read(A); A ← A + 100;				
Write(A);	Read(A); A ← A*2; Write(A);	125 250		
Read(B); B ← B + 100; Write(B);			125	
	Read(B); B ← B*2; Write(B);			
		250	250	Final Balance

In above given schedule Part of T_1 is executed which updates A to 125. Then processor switches to T_2 and part of T_2 which updates A to 250 is executed. Then context switch to T_1 and remaining part of T_1 which updates B to 250 is executed. At the end remaining part of T_2 which reads B as 125 and updates it to 250 by multiplying value of B by two. This concurrent schedule also maintains consistency of database as ultimately A + B is 250 + 250 = 500 (unchanged).

The result of above transaction is exactly same as serial schedule shown in above Table 10.6.1. Therefore above schedule can be converted to equivalent serial schedule and hence it is consistent schedule.

10.7 Serializability / Serializable Schedule

Q. Explain the concept of Serializability with its types **MU - Dec. 18, 10 Marks**

(1) Introduction

- The database system must control above concurrent execution of transactions serializability will ensure that the database state remains in consistent state.
- We first need to understand which schedules will ensure consistency, and which schedules will not.
- A schedule is the actual execution sequence of concurrent transactions.
- A schedule of two transactions T_1 and T_2 is 'serializable' if and only if executing this schedule has the same effect as any serial schedule (either $T_1 : T_2$ or $T_2 : T_1$).

- We consider only two operations: read and write for purpose of computational simplicity.
- Between a read (Q) instruction and a write (Q) instruction on a data item Q, a transaction may perform sequence of operations on the copy of Q that is residing in the local buffer of the transaction.
- Thus, the only important operations of a transaction from a scheduling point of view are its read and write instructions.
- Various forms of schedule equivalence are :

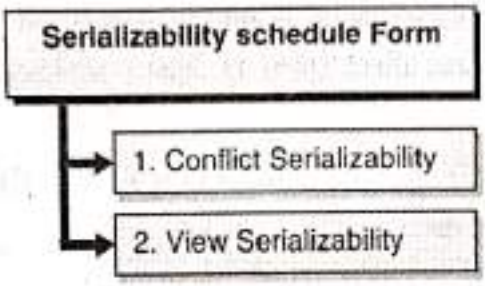


Fig. 10.7.1 : Serializability schedule form

10.7.1 Conflict Serializability

Q. Explain conflict serializability, describe using example. **(4 Marks)**
Q. Describe conflict serializability. **(4 Marks)**

(1) Introduction

The database system must control concurrent execution of transactions which ensure that the database state remains in consistent state.

(2) Conflict

- A pair of consecutive database actions (reads, writes) is in conflict if changing their order would change the result of at least one of the transactions.

Transaction T_i \ Transaction T_j		Read(D)	Write(D)
	Read(D)	No Conflict	Conflict
	Write(D)	Conflict	Conflict

- Consider schedule S has two consecutive instructions I_i and I_j from transactions T_i and T_j respectively.
- If I_i and I_j access to different data items then they will not conflict and can be swapped, without any problem.
- If I_i and I_j **access to same data item D** then consider following consequences :

- I_i = READ (D), I_j = READ (D) then **no conflict** as they only read value.

  This operation is called as non conflicting swap.
- I_i = READ (D), I_j = WRITE (D) then they **conflict** and cannot be swapped.
- I_i = WRITE (D), I_j = READ (D) then they **conflict** and cannot be swapped.
- I_i = WRITE (D), I_j = WRITE (D) then they **conflict** and cannot be swapped.

– So we can say that instructions conflict if both consecutive instructions operate on same data item and from different transactions and one of them is WRITE operation.

– If I_i and I_j **access to different data item D** then consider following all consequences **no conflict** as they only read or writing different values.

– I_i = READ (D)/WRITE (D), I_j = READ (P) /WRITE (P) then **no conflict** as they only reading or writing different data.

– The following set of actions is conflicting :

T_1:R(X), T_2:W(X), T_3:W(X)

– While the following sets of actions are not :

T_1:R(X), T_2:R(X), T_3:R(X)

T_1:R(X), T_2:W(Y), T_3:R(X)

(3) Conflict equivalence

Two schedules are conflict equivalent if they can be turned into one another by a sequence of non conflicting swaps of adjacent actions.

(4) Example :

– A schedule is conflict serializable if it is conflict equivalent to a serial schedule.

T_1	T_2
Read(P)	
Write(P)	
	Read(P)
Read(Q)	
	Write(P)
Write(Q)	
	Read(Q)
	Write(Q)

Fig. 10.7.2 : Schedule S

- Instruction WRITE(P) of T_1 and READ(P) of T_2 cannot be swapped as they conflict. (as shown in Fig. 10.7.3)

	T_1	T_2
	Read(P)	
Can not Be Swap	Write(P)	
		Read(P)
	Read(Q)	
		Write(P)
	Write(Q)	
		Read(Q)
		Write(Q)

Fig. 10.7.3

- Instruction READ(Q) of T_1 and READ(P) of T_2 can be swapped as they operate on different data items so do not conflict. (as shown in Fig. 10.7.4)

	T_1	T_2
	Read(P)	
	Write(P)	
Can be swapped		Read(P)
	Read(Q)	
		Write(P)
	Write(Q)	
		Read(Q)
		Write(Q)
	Before Swap	

T_1	T_2
Read(P)	
Write(P)	
Read(Q)	
	Read(P)
	Write(P)
Write(Q)	
	Read(Q)
	Write(Q)
After Swap	

Fig. 10.7.4

- Instruction WRITE(Q) of T_1 and WRITE(P) of T_2 can be swapped as they operate on different data items so do not conflict.

Can be swapped

T_1	T_2
Read(P)	
Write(P)	
Read(Q)	
	Read(P)
	Write(P)
Write(Q)	
	Read(Q)
	Write(Q)

T_1	T_2
Read(P)	
Write(P)	
Read(Q)	
	Read(P)
Write(Q)	
	Write(P)
	Read(Q)
	Write(Q)

Fig. 10.7.5

- Instruction WRITE (Q) of T_1 and READ (P) of T_2 can be swapped as they operate on different data items so do not conflict.

can be Swap As no conflict

T_1	T_2
Read(P)	
Write(P)	
Read(Q)	
	Read(P)
Write(Q)	
	Write(P)
	Read(Q)
	Write(Q)

T_1	T_2
Read(P)	
Write(P)	
Read(Q)	
Write(Q)	
	Read(P)
	Write(P)
	Read(Q)
	Write(Q)

Fig. 10.7.6 : Schedule R

- Now the schedule S after performing swapping can transformed into schedule R as shown above which also results in same values of P and Q.
- The above schedule is same as serial schedule $<T_1, T_2>$.
- If a schedule S can be transformed into a schedule R by a series of swap operations on non conflicting instructions, then we can say **schedule S and R are Conflict equivalent.**
- If a concurrent schedule is conflict equivalent to a serial schedule of same transactions then it is Conflict Serializable. So schedule **S is conflict serializable** to serial schedule $< T_1, T_2>$.

10.7.2 View Serializability

Q. Explain view serializability, describe using example. (4 Marks)

(1) Introduction

View equivalence is less strict than conflict equivalence, but it is like conflict equivalence based on only the read and write operations of transactions.

(2) Conditions for view equivalence

Let, D = Data item

S_1, S_2 = Transaction schedules

T_i, T_j = Database transaction

- Schedules S_1 and S_2 are view equivalent if they satisfy following conditions for each data item D,
  - (a) S_1 and S_2 must have same transactions included and also they are performing same operations on same data. If T_i reads initial value of D in S_1, then T_i also reads initial value of D in S_2.
  - (b) If T_i reads value of D written by T_j in S_1, then T_i also reads value of D written by T_j in S_2.
  - (c) If T_i writes final value of D in S_1, then T_i also writes final value of D in S_2.
- First 2 conditions ensure that transaction reads same value in both schedules.
- Condition 3 ensures that final consistent state.
- If a concurrent schedule is view equivalent to a serial schedule of same transactions then it is **View serializable.**
- Consider following schedule S_1 with concurrent transactions $<T_1, T_2, T_3>$.
- In both the schedules S_1 and a serial schedule $S_2<T_1, T_2, T_3>$ T_1 reads initial value of D. Transaction T_3 writes final value of D. So schedule S_1 satisfies all three conditions and is view serializable to $<T_1, T_2, T_3>$.

(3) Example :

Below two schedules S and R are view equivalent,

Schedule S		
T_1	T_2	T_3
Read(P)		
	Write (P)	
Write (P)		
		Write (P)

Schedule R		
T_1	T_2	T_3
Read(P)		
Write (P)		
	Write (P)	
		Write (P)

Fig. 10.7.7

(4) Note

- Every conflict serialisable schedule is view serialisable but not vice versa.
- In above example T_2 and T_3 Writes data without reading value of data item by themselves so they are called as "Blind Writes".

10.8 Precedence Graph (Test for Serializability)

Q. Explain method of precedence graph used for checking serializability. **(4 Marks)**
Q. What is term precedence graph ? **(4 Marks)**

(1) Introduction

- A particular transaction schedule can be serialized; we can draw a precedence graph.
- Precedence (or serializability) graph for schedule S is a graphical representation of transactions executed.
- A precedence graph is also known as conflict graph or serializability graph.
- Precedence graph is a graph of nodes and vertices, where the nodes are the transaction names and the vertices are attribute collisions.

(2) Algorithm for precedence graph

(a) Add a node for each transaction.

(b) Add a directed edge from T_i to T_j, if T_j reads the value of an item written by T_i.

(c) Add a directed edge from T_j to T_i, if T_j writes a value in to an item after it has been read by T_i.

(3) Example 1 :

Let us consider Schedule C

T_1	T_2
Read(A); A ← A + 100;	
Write(A);	
	Read(A); A ← A*2;
	Write(A);
Read(B); B ← B + 100;	
Write(B);	
	Read(B); B ← B*2;
	Write(B);

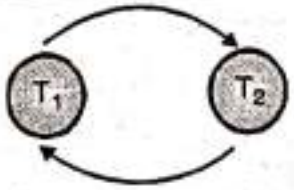


Fig.10.8.1

Example 2 :

T_1	T_2	T_3
READ(A)		
READ(B)		
	READ(A)	
	READ(B)	
		WRITE (A)
WRITE (C)		
WRITE (B)		
		WRITE (C)

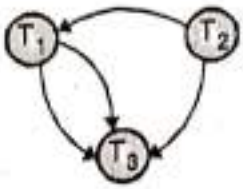


Fig. 10.8.2

As no cycle in graph

∴ Schedule is conflict serializable

∴ Corresponding serial schedule is given as,

$T_2 \rightarrow T_1 \rightarrow T_3$

(4) Test for conflict serializability

1. A schedule is conflict serialzable if it produces an acyclic precedence graph.
2. **Example :**

$r_1(a), w_1(a), r_1(b), r_2(b), w_2(b), w_1(b)$

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
	Read(B)
	Write(B)
Write(B)	

Fig.10.8.3

3. **Conclusion**

Not conflict serialzable as there is cycle in above precedence graph.

10.9 Solved Examples

Example 10.9.1 : A schedule has transactions T_1 and T_2 as given below;

$r_1(x), r_2(z), r_1(z), r_3(x), r_3(y), w_1(x), w_3(y), r_2(y), w_2(z), w_2(y)$

1. Draw precedence graph.

2. Is schedule conflict serialisable or not ? Find respective serial schedule.
3. Is above Schedule view serialisable or not ? (10 Marks)

Solution :

(1) **Step 1 :**

$r_1(x), r_2(z), r_1(z), r_3(x), r_3(y), w_1(x), w_3(y), r_2(y), w_2(z), w_2(y)$

Schedule S : Given Schedule

T_1	T_2	T_3
r(x)		
	r(z)	
r(z)		
		r(x)
		r(y)
w(x)		
		w(y)
	r(y)	
	w(z)	
	w(y)	

(2) **Step 2 :** Precedence graph

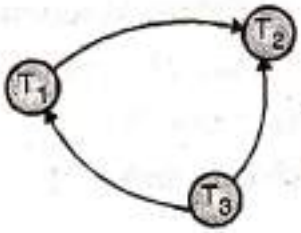


Fig. P. 10.9.1

∴ Above graph show no cycle

∴ Schedule s_1, is conflict serializable.

(3) **Step 3 :** Corresponding serial schedule

$T_3 \rightarrow T_1 \rightarrow T_2$

Schedule S_1 : equivalent serial schedule

T_1	T_2	T_3
		r(x)
		r(y)
		w(y)
r(x)		

T_1	T_2	T_3
r(z)		
w(x)		
	r(z)	
	r(y)	
	w(z)	
	w(y)	

∴ **Serial Schedule S_1 :** $r_3(x), r_3(y), w_3(y), r_1(x), r_1(z), w_1(x), r_2(z), r_2(y), w_2(z), w_2(y),$

(4) Step 4 : Above schedule is view serializable or not.

Conditions for view serializability

(1) S and S_1 must have same number of transaction and same number of read write operation.

(2) Initial read operation
- X is read initially by T_3 in S as well as S_1
- Y is read initially by T_3 in S and S_1 both.
- Z is read initially by T_1 in both S and S_1

(3) If S reads value of X or Y written by T; then S_1 also reads value of x or y written by same transaction.
- S_1 reads value Y written by T_3.
- S also reading value of Y which is written by T_3.

(4) Final write operation
- X-final write (x) done by T_1 in S as well as S_1
- Y-final write (y) done by T_2 in S and S_2 both.

∴ S is view serializable.

Or S is view equivalent of S_1.

Review Questions

Q. 1 Explain with example transaction processing.

Q. 2 Discuss the ACID properties of transaction processing.

Q. 3 Explain the transaction processing with the help of state diagram.

Q. 4 What is transaction ? What are functions of commit and rollback ?

Q. 5 Write short note on : Commit and rollback.

□□□

CHAPTER 11

Concurrency Control

Module VI

Syllabus

Concurrency Control : Lock-based, Timestamp-based protocols.

11.1 Concept of Concurrency Control

Q. Explain the concept of concurrency control (2 Marks)

- In a single user database only one user is accessing the data at any time.
- This means that the DBMS does not have to be concerned with how changes made to the database will affect other users.
- In a multi-user database many users may be accessing the data at the same time. The operations of one user may interfere with other users of the database.
- The DBMS uses concurrency control to manage multi-user databases.
- Concurrency control is concerned with preventing loss of data integrity due to interference between users in a multi-user environment.

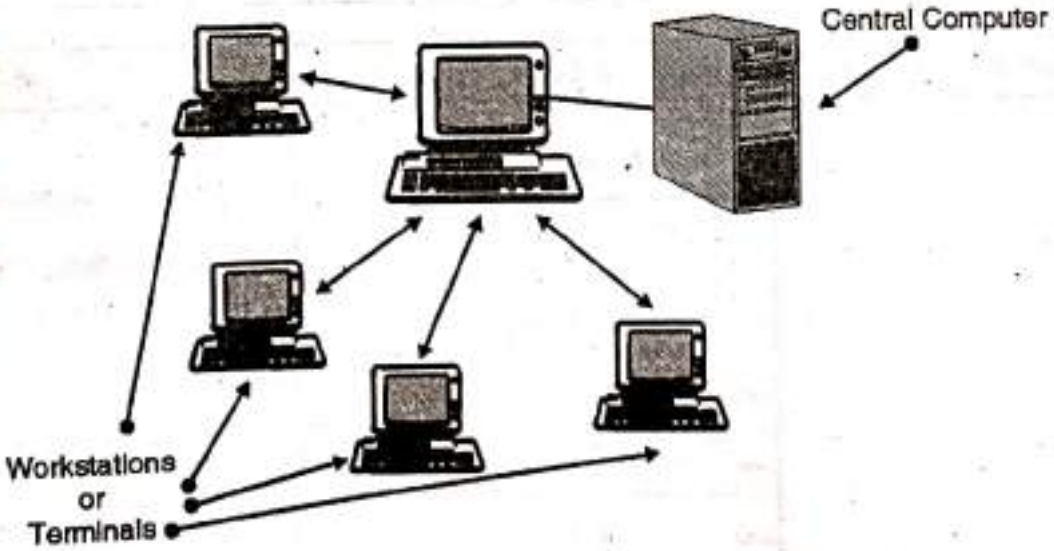


Fig. 11.1.1 : Concurrent database access

- Concurrency control should provide a mechanism for avoiding and managing conflict between various database users operating same data item at same time.

11.2 Conflicting Transactions

Q.	Explain types of conflicting transaction	(4 Marks)
Q.	Explain conflict table with all cases.	(4 Marks)
Q	Explain Conflicting Transactions with example.	(4 Marks)

- A pair of consecutive database actions (reads, writes) is in conflict if changing they are accessing and changing same data simultaneously.
- The main conditions for conflict is transactions reading and/or writing the same data items.

Table 11.2.1

Transaction T_j	Transaction T_i			
	Read		Write	
	Accesses Same Data	Accesses Different Data	Accesses Same Data	Accesses Different Data
Read	No Conflict	No Conflict	**Conflict**	No Conflict
Write	**Conflict**	No Conflict	**Conflict**	No Conflict

- In Table 11.2.1 if transaction T_i and T_j accesses same data item for read(T_j) and write(T_j) then there is said to be **conflict** but if they are accessing different data items for read(T_j) and write(T_j) then there is said to be **no conflict**.

11.3 Problems Caused by Concurrent Executions

Q.	Explain various problems of concurrent executions.	(4 Marks)

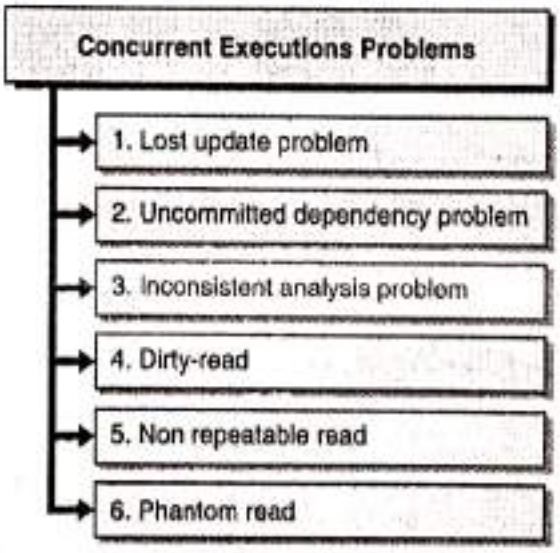


Fig. 11.3.1 : Concurrency execution problems

(1) Lost update problem

- Update made by one transaction is overwritten by other transaction or other user.
- This may loss updates of one transaction.

Example 11.3.1 :

Time	X	Y	Balance
t1	-	BEGIN TRANSACTION	500
t2	BEGIN transaction	READ (A)	500
t3	READ (A)	A = A + 100	500
t4	A = A – 10	WRITE (A)	600
t5	WRITE (A)	COMMIT	490
t6	COMMIT	-	490

T_1 & T_2 - Transaction Y has read the value of Aas 500.

T_3 - Transaction Xhas read the value of Aas 500.

T_4 - TransactionY writes the new value of A as 200

- Transaction X has subtracted 10 from its value of Ato produce 490.

T_5 - Transaction X updates the value of Aon disc.

T_6 - The result of this operation T_4performed by transaction Y is lost.

Transaction X has overwritten the result of transaction Y.

Solution :

The problem is avoided by not allowing only one transaction to update data at a time.

(2) Uncommitted dependency problem

- This problem occurs when user sees data coming from intermediate step of another ongoing transaction which is yet uncommitted.

Time	X	Y	A
t_1	–	begin transaction	100
t_2		read (A)	100
t_3		A=A + 100	100
t_4	begin transaction	Write (A)	200
t_5	read (A)		200

Time	X	Y	A
t_6	A=A – 10	Rollback	100
t_7	write (A)		190
t_8	commit		190

Example 11.3.2 : Transaction Y reads and updated the value of A (100 + 100 = 200) and writes the result at time t_4.

Transaction X should be updating A = 100 because transaction Y has been rolled back and its changes undone.

Transaction X has used the result of transaction Y but this result was incorrect as transaction Y failed.

Solution :

This problem is avoided by not allowing transaction X to read the value of A until transaction Y either commits or rolls back.

(3) Inconsistent analysis problem

Transaction reads partial results of incomplete transaction update made by other transaction.

T_1	T_2
–	BEGIN Transaction
BEGIN Transaction	SUM=0
R (X)	R (X)
X = X – 20	SUM = SUM + X
W (X)	R (Y)
R (Z)	SUM = SUM + Y
Z = Z + 10	
W (Z)	
COMMIT	R (Z)
	SUM = SUM + Z
	COMMIT

Example 11.3.3 : Transaction T_2 is adding the values of X, Y and Z.

However, at the same time, transaction T_1 is transferring Rs.100 between X and Z.

As transaction T_2 has used the old balances of X and Zits final result is incorrect.

Solution :

This problem can be solved by preventing transaction T_1 from transferring the amount, Before transaction T_2 has committed.

(4) Dirty-read

- In database transactions, one transaction reads and changes the value while the other reads the value before committing or rolling back by the first transaction.
- Dirty data : Data, updated by a transaction, but not yet committed hence all users reading old data which is called as dirty data.
- Dirty read : A transaction reading dirty data is called as 'dirty read'.
- Because there is always a chance that the first transaction might rollback the change which causes the second transaction reads an invalid value.
- In short, dirty read is changes made during a Transaction are 'visible' to other parties.
- Whether dirty reads are actually avoided or not depends on the database backend used and/or its configuration.

(5) Non repeatable read

- A non-repeatable read occurs when a persistent object is read twice within a same transaction
- It is possible that between the reads, data is modified by another transaction, therefore the second read returns different values as compared to the first;
- If Transaction T_1 reads a row and Transaction T_2 changes the same row, when T_1 rereads and sees the changes made by T_2.Then this is non - repeatable read.

(6) Phantom read

- Phantom reads means insert or delete action is performed on a table row which referred by another transaction.
- Transaction T_2 inserts a row, T_1 rereads the query and if T_1 see the additional row, it is a ghost row to T_1 then this is called as **Phantom read.**

11.4 Concurrency Control Schemes

Q. Explain various concurrency control schemes. (4 Marks)

- Fundamental property of transaction is isolation.

- Isolation property ensures that each transaction must remain unaware of other concurrently executing transactions.
- In case of concurrent transactions, when several transactions are executing simultaneously on a database may not preserve isolation property for long time.
- To implement concurrent system there must be interaction among various concurrent transactions.
- This can be done by using one of the concurrency control schemes.
- All the following schemes are based on serializability of schedules.

11.5 Lock Based Protocols

Q.	Explain concept of locking.	(4 Marks)
Q.	Explain algorithm of lock based protocol.	(4 Marks)
Q	Explain lock based protocols.	(4 Marks)
Q	What are the types of locks ?	(4 Marks)
Q	What are locking levels ?	(4 Marks)

(1) Introduction (concept of locking) - implementation of isolation

- In concurrent environment, many users can access same data in a DBMS simultaneously; each user feels that he is having exclusive access to the database.
- To achieve such system, we must have interaction amongst those concurrent transactions which is also called as Mutual Exclusion.
- Mutual exclusion is required for concurrent executions of various transactions.
- Whenever one transaction is accessing data, second transaction should not change data otherwise there may be dirty read problem. This can be done with help of locking concept.
- Transaction can access data if it is locked by that transaction.
- Locking is necessary in a concurrent environment to assure that one process should not retrieve or update a record which another process is updating.
- Failure to this would result in inconsistent and corrupted data.
- For example, in case of traffic signals only one lane (line) is allowed to pass at a time and other lanes are locked. Similarly, in data transaction only one transaction can perform operations at a time other transitions are locked.

(2) Types of locks

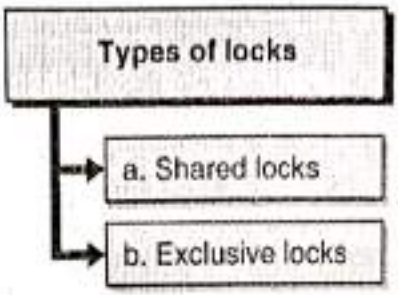


Fig. 11.5.1 : Types of locks

There are two types of locking to control concurrent access :

(a) Shared locks

- This type of locking is used by the DBMS when a transaction wants to only **read** data without performing modification to it from the database.
- Another concurrent transaction can also acquire a shared lock on the same data, allowing the other transaction to read the data.
- Shared locks are represented by S.
- If a transaction T_1 has obtained a shared lock on data item X, then transaction T_1 can only read data item X, but cannot write on data item X.
- SQL Implementation :

```
LOCK TABLE customer IN
SHARED MODE;
```

(b) Exclusive locks

- This type of locking is used by the DBMS when a transaction wants to **read or write** (i.e. performing update) data in the database.
- When a transaction has an exclusive lock on some data, other transactions cannot acquire any type of lock (shared or exclusive) on the data.
- Exclusive locks are represented by X.
- If a transaction T_1 has obtained a exclusive lock on data item X, then transaction T_1 can read data item X and also can write on data item X.
- SQL Implementation :

```
LOCK TABLE customer IN
EXCLUSIVE MODE;
```

(c) Lock compatibility matrix

		Lock by Transaction T_j	
		S	X
Lock by Transaction T_i	S	No Conflict	Conflict
	X	Conflict	Conflict

- Transaction can acquire shared lock although there is other transaction which currently has a shared or an exclusive lock on the data. As many transactions can READ data without any conflict.
- Transaction can acquire an exclusive lock only if no other transaction currently has a shared or an exclusive lock on the data.
- To avoid these kinds of problems, every transaction in the system should follow a set of rules which are also called as **locking protocol**.
- **Locking protocols** tells when every transaction should lock or unlock data item.

(3) Locking Levels

There are two locking levels to achieve concurrency :

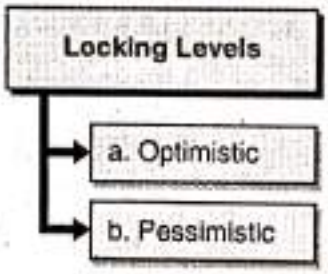


Fig. 11.5.2 : Locking levels

a. Optimistic

- Transactions with optimistic concurrency, work on the assumption that resource conflicts when more than one transaction works on the same set of data are unlikely (though possible).
- Optimistic transactions check for potential conflicts when committing changes to a database and conflicts are resolved by resubmitting data.

b. Pessimistic

- Pessimistic transactions expect conflicts from beginning and lock all resources.
- User can select either of the locking techniques.

11.5.1 Working of Locking Protocol / Locking Scheduler / Locking Manager

Q. How does locking protocol works ? **(4 Marks)**
Q. Write a short note on scheduler. **(4 Marks)**

- Whenever any user transaction want to access any data item (D) then he will issue a locking request to concurrency control manager.
- A lock manager is a process that receives **locking request messages** and sends response accordingly.
- Now based on some test criteria lock request can be granted or rejected.
- The response given may be
  - Lock grant - On granting to request.
  - If there is deadlock then rollback transaction.
  - If lock is already held by other transaction then asking to wait.
  - On unlock just an acknowledgement is sent.
- Transaction can access data if it is locked by that transaction.
- Lock manager uses linked list and hash tables to store information about all locking requests.
- The table used to store locking information is called as **lock-table**. Each record store which transaction has locked data, locking mode, request grant response.
- Records are chained using linked list for a data item.
- The linked list is set of transaction requesting for that item and holding lock on data item and others are waiting.

Example :

Consider two transactions T_1 and T_2,

T_1 :
LOCK – X(Q)
READ (Q)
Q = Q – 100
WRITE (Q)
UNLOCK (Q)
LOCK-X(P)
READ (P)
P = P + 100
WRITE(P)
UNLOCK(P)

T_2 :	LOCK-S(P)
	READ(P)
	UNLOCK(P)
	LOCK – S(Q)
	READ (Q)
	UNLOCK (Q)
	DISPLAY(P + Q)

Following is the schedule for T_1 and T_2, which is not scheduled properly because of granting locks at improper timings.

Table 11.5.1 : Schedule A

T_1	T_2	Concurrency Control Manager
LOCK – X(Q)		
		Grant - X(Q, T_1)
READ (Q)		
Q = Q – 100		
WRITE (Q)		
UNLOCK (Q)		
	LOCK-S(P)	
		Grant - S(P,T_2)
	READ(P)	
	UNLOCK(P)	
	LOCK – S(Q)	
		Grant - S(Q,T_2)
	READ (Q)	
	UNLOCK (Q)	
	DISPLAY(P + Q)	
LOCK-X(P)		
		Grant - X(P,T_1)
READ(P)		
P = P + 100		
WRITE(P)		
UNLOCK(P)		

In above schedule, T_1 unlocks Q very early, because of which the inconsistent state is exposed to transaction T_2 which results in wrong value of P + Q.

Assume P = 400, Q = 600 then serial schedule < T_1, T_2 > will show 1000 but above given schedule will result in 900 as T_1 has not updated P and before that only P + Q displayed by T_2.

Every time transaction requests for lock and concurrency control manager grants it (if it is not conflicting) then only next instructions are executed. Always this sequence is followed.

Such kind of schedules may lead to some undesirable conditions, called as **deadlock.**

Both the transactions are holding a lock on two different data items, and waiting to acquire lock on each other's data item, this will never end up **waiting** of them and schedule cannot be executed further. This situation can be solved by roll back of one of the transactions. But this may lead to rollback of another transaction (such situation is called as cascaded rollback occurs in dependent transactions).

Locking protocol :

- To avoid these kinds of problems, every transaction in the system should follow a set of rules called as 'locking protocol'.
- It tells when every transaction should lock and unlock data item.
- It restricts set of possible schedules and all those are serializable schedule.

11.5.2 Granting Locks

Q. What is granting locks ? (4 Marks)

- Lock is granted only when no other conflicting type of lock on it, is held by other transactions.
- Lock compatibility matrix shows which locks will be granted and which will be rejected.
- It may happen that a series of transactions holding shared lock on data item X one by one so transaction T has to wait for exclusive lock on data item X forever, this situation is called as starvation of T.
- To avoid starvation, Concurrency Control Manager should consider two things,
  (a) No other transaction is holding conflicting lock on it.

(b) No other transaction is waiting on that data item for locking it.

11.5.3 Rejecting Locks

Q. What is rejecting locks ? (4 Marks)

- Lock is rejected when other conflicting type of lock is held by other transactions.
- If any transaction has exclusive lock for writing on data item D then no other transaction can acquire any type of lock on data item D.

Lock compatibility matrix will show the locks that will be granted or rejected.

11.6 Two-Phase Locking (2PL)

Q. Explain advance locking concept using 2P locking (4 Marks)
Q. Explain terms two phase locking. (4 Marks)
Q. How does 2P protocols work ? Write its advantages and disadvantages. (4 Marks)

(1) Introduction

- Two-Phase Locking (2PL) synchronizes reads and writes by explicitly detecting and preventing conflicts between concurrent operations.
- Before reading data item X, a transaction must "own" a read lock on X. Before writing into X, transaction must "own" a write lock on X.
- The ownership of locks is governed by two rules :
  (a) Different transactions cannot simultaneously own conflicting locks (i.e. WR).
  (b) Once a transaction surrenders ownership of a lock, it may never obtain additional locks.
- The definition of conflicting lock depends on the type of synchronization being performed : (Refer lock compatibility matrix)
- For **'RW'** synchronization two locks conflict if
  - Both are locks on the same data item.
  - One is a read lock and the other is a write lock.
- For **'WW'** synchronization two locks conflict if.
  - Both are locks on the same data item.
  - Both are write locks.

- The second lock ownership rule causes every transaction to obtain locks in a two phase manner.

Growing phase

In this phase, transaction may obtain n number of new locks but may not release any lock.

Shrinking phase

Transaction may release locks but cannot obtain any new Lock.

(2) Working of 2P protocols

- Initially transaction is in 'growing phase', it acquires locks as needed. Once it starts releasing locks it enters into the 'Shrinking Phase' and now it may not acquire any lock after shrinking phase starts.
- The point at which transaction obtains final (last) lock is called as **lock point of transaction.**

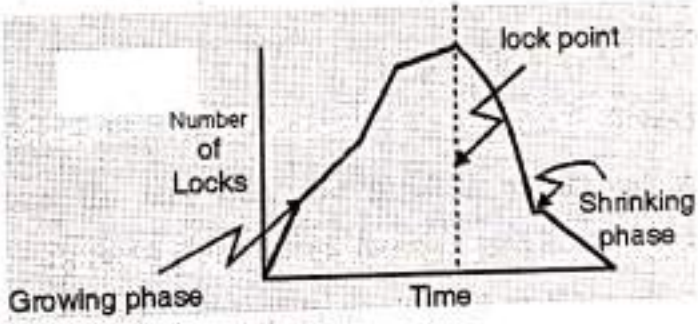


Fig. 11.6.1 : Locking point

- **Lock point** is end of growing phase for that transaction and start of shrinking phase.
- The schedule to be serializable, transactions should be arranged according to their lock points.
- When the transaction terminates (or aborts), all remaining locks are automatically released. Some systems also require that transactions hold all locks until termination.

(3) Advantages

- Two - phase locking protocol ensures conflict serializability.
- Two-phase locking protocol is simple to implement and understand.

(4) Drawbacks

- Deadlock may occur in two phase locked schedule.
- Cascaded rollback may occur under two phase locking. Cascaded rollbacks do mean by if

one transaction is rolling back all transactions depends on this transaction will be rolled back.

Example :

(A) No 2P (B) with 2P locking.

11.6.1 Modified Versions of Two-phase Locking Protocol

Q. Explain different variant of 2PL. (4 Marks)

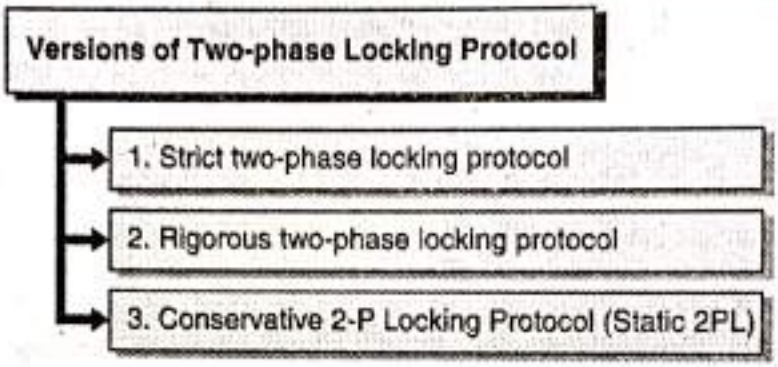


Fig. 11.6.2 : Versions of Two-phase Locking Protocol

(1) Strict two-phase locking protocol

- Avoids cascaded rollbacks.
- It requires not only two-phase locking but also that all exclusive-locks held by transaction should be held until that transaction commits or abort.
- This property ensures that if data is being modified by one transaction (holding lock-X) then other transaction can't read it until first transaction commits.
- Strict schedule recoverability.
- Not deadlock free.

(2) Rigorous two-phase locking protocol

- It also avoids cascading rollbacks.
- It requires that all share and exclusive locks to be held until the transaction commits.
- So transactions can be serialized in the sequence they commit.
- Most of the database systems implement strict or rigorous two phase locking protocol.

(3) Conservative 2-P Locking Protocol (Static 2PL)

- It is also called as static 2P locking protocol.

- This scheme requires locking all items needed to access before the transactions starts.
- It begins execution by declaration about read set and write Set of all data items needed in advance.
  - **Read set :** Set of all data transactions lock.
  - **Write set :** Set of all data than n-calls transactions lock.
- If any one item of above list is currently not available for locking then lock will not be granted it waits till all items are ready for locking.
- It is almost free from deadlocks as all required items are listed in advanced.

11.6.2 Lock Conversions - Upgrading and Downgrading Lock

Q. Explain Lock Conversions. (4 Marks)

Consider following example,

T_1	T_2
READ (d_1)	.
READ (d_2)	.
.	READ (d_1)
.	READ (d_2)
.	Show ($d_1 + d_2$)
.	
READ (d_n)	
WRITE (d_q)	

- If we use two-phase locking protocol then we have to have exclusive lock on d_1 and d_2 otherwise T_2 may show inconsistent value. But this will lower concurrency and schedule will turn into serial schedule.
- In such cases 'Lock conversion' is needed which will convert lock-S to lock-X and vice-versa.
- **Upgrade** - Converting shared lock to exclusive lock.
- **Downgrade** - Converting exclusive lock to shared lock.
- Upgrading is done in growing phase and downgrading is done in shrinking phase.
- It increases level of concurrency in schedule.
- Two phase locking with lock conversion generates only conflict serializable schedules and transactions are serialized by their lock points.
- A partial schedule under two phase locking is as shown below.

T_1	T_2	T_3
LOCK – X(P)		
READ (P)		
LOCK-S(Q)		
READ (Q)		
WRITE (P)		
UNLOCK (P)		
	LOCK – X (P)	
	READ (P)	
	WRITE (P)	
	UNLOCK (P)	
		LOCK-S(P)
		READ (P)

Fig. 11.6.3 : 2P locking is applied in schedule

11.7 Timestamp Based Protocols

Q. What do you mean by timestamp based protocol? Explain timestamp modeling protocol. (4 Marks)

(1) Introduction

- To achieve serializability order of transactions for execution can be decided in advance using its time at which transaction entered in system.
- A general method to achieve this is using time stamp ordering protocol.

(2) Concept of Timestamp

- A fixed timestamp is assigned at start of execution of the transaction. Every transaction T_j has been assigned a timestamp by database system denoted as TS (T_j).
- A transaction which has entered in system recently will have greater timestamp. If transaction T_j starts after T_i then,
- $$TS(T_i) < TS(T_j)$$
- **System clock** is used as timestamp i.e. system time when transaction T_i enters system or a **logical counter** can be used as timestamp and incremented after every assignment.
- If $TS(T_i) < TS(T_j)$, then system must ensure that serializable schedule is equivalent to a serial schedule as $< T_i, T_j >$

- Every data item X is with two timestamp values :

(a) W-timestamp (X)

- It denotes the largest timestamp of any transaction that executed WRITE (X) successfully on given data item X.
- That mean it is timestamp of recent WRITE (X) operation.
- On execution of next WRITE (X) operation (O_1), W-timestamps will be updated to new timestamp of O_1 i.e. TS (O_1).

(b) R-timestamp (X)

- Denotes the largest timestamp of any transaction that executed READ (X) successfully on data item (X).
- That mean it is timestamp of recent READ (X) operation.
- On execution of next READ (X) operation (O_1), R-timestamps will be updated to new timestamp of O_1 i.e. TS (O_1).

(3) Timestamp - ordering Protocol

- This protocol ensures any conflicting READ or WRITE is executed in order of timestamp.
- If by any reasons, if transaction is aborted then on restarting, new timestamp is assigned.

Working

(a) For transaction T_i to execute READ (X) Operation,

If TS (T_i) < W - timestamp (X)

Then T_i is trying to read value of X that is overwritten by other transaction.

W - timestamp (X) = 149 (Recent Write)			
TS	**T_i**	**T_x**	**Operations**
148	READ (X)	...	TS (T_i) < W - timestamp (X) i.e. 148 < 149 (W-Timestamp)
149	Unlock (X)	WRITE (X)	Record W - timestamp (X) = 149

So this READ is rejected (as T_x is already performing write operation on data so no other operation can be performed by any other transaction) and T_i is rolled back.

If TS (T_i) ≥ W - timestamp (X)

Then READ executed and set

R-timestamp(X) = max {TS (T_i), R-timestamp}			
TS	**T_i**	**T_x**	**Operations**
148	....	WRITE (X)	Record W - timestamp (X) = 148 (recent write)
149	READ (X)	....	TS (T_i) ≥ W - timestamp (X) i.e. 149 ≥ 148 (W-Timestamp) Record R-timestamp(X) = 149
150	....	....	....
151	READ (X)	....	TS (T_i) ≥ W - timestamp (X) i.e. 151 ≥ 148 (W-Timestamp) Record R-timestamp(X) = max{149,151} = 151

(b) If transaction T_i execute WRITE (X) Operation,

If TS (T_i) < R - timestamp (X)

Then T_i has produced value of X which is not needed now so rollback T_i.

TS	**T_i**	**T_x**	**Operations**
148	....	...	...
149	WRITE(X)	....	TS (T_i) < R - timestamp (X) i.e. 149 < 151 (R-Timestamp) Reject WRITE roll back T_i
150	....	....	....
151		READ (X)	Record R-timestamp(X) = 151 (Recent READ Operation)

If TS (T_i) < W - timestamp (X)

Then T_i is trying to write obsolete value of X, So rollback T_i

TS	T_i	T_x	Operations
148	....	...	...
149	WRITE (X)	....	TS (T_1) < W - timestamp (X) i.e. 149 < 151 (W-Timestamp) Cannot write as other transaction using data X for writing.
150	....	....	....
151		WRITE (X)	Record W-timestamp(X) = 151

(c) Otherwise, system executes WRITE (X) and sets

W-timestamp (X) = TS (T_i)

TS	T_i	T_x	Operations
150	....	....	....
151	WRITE (X)		Record W-timestamp(X) = 151

(4) Advantages

- This protocol ensures conflict serializability, as conflicting operations are processed in order of timestamp of operation.
- Ensures that it is free from deadlock.

(5) Disadvantages

- Starvation is possible for long transaction if short transaction conflicts with it causes restorative of long transaction again and again.
- It may not give recoverable schedules.

11.8 Thomas' Write Rule

Q. Explain Thomas write rule with algorithm. (4 Marks)

(1) Introduction

- Thomas Write Rule is also known as Modified timestamp protocol.

- Thomas Write Rule uses view serializability.
- It generates schedules which are not possible by other protocols.
- Generated schedules that are view equivalent to the serial schedule.

(2) Example :

T_1	T_2
READ (D)	
	WRITE (D)
WRITE (D)	

- The above schedule with 2 transactions T_1 and T_2.
- As T_1 starts before T_2 so timestamp TS (T_1) < TS (T_2). T_1 executes READ (D), and then T_2 executes WRITE (D). Now W-timestamp (D) = TS (T_2) so when T_1 executes WRITE (D), but TS (T_1) <W- timestamp (D), So WRITE (D) of T_1 is rejected and T_1 is rolled back.
- But this rollback is unnecessary. Since T_2 has already written value of D and T_1 is trying to write value which will never be read.
- So any time if T_i with TS (T_i) < TS (T_2) trying to READ (D) will be rollback as TS (T_i) < W - timestamp (D). Any transaction T_j with TS (T_j) > TS (T_2) should read value of D by T_2.
- So such WRITE (D) operations (as in T_2) are obsolete and should be ignored.

(3) Thomas' Write Rule

- To achieve this functionality timestamp ordering protocol is modified which is called as **Thomas Write Rule.**
- Rejects few write (D) operations by modifying check for WRITE(D).
- Suppose, T_i issues WRITE (D)

(a) If TS (T_i) < R - timestamp (D)

Then T_i is producing value of D, which needed previously not now Assuming it is never produced the system rejects WRITE and T_i is rollback.

TS	T_i	T_x	Operations
148	....	...	...
149	WRITE(X)	....	TS (T_1) < R - timestamp (X) i.e. 149 < 151 (R-timestamp) Reject WRITE roll back T_i
150	....	....	....
151		READ (X)	Record R-timestamp(X) = 151 (Recent READ Operation)

(b) If TS (T_i) < W-timestamp (D)

Then T_i is writing obsolete or outdated value of D so WRITE is ignored.

TS	T_i	T_x	Operations
148	....	...	...
149	WRITE (X)	....	TS (T_1) < W - timestamp (X) i.e. 149 < 151 (W-timestamp) Cannot write as other transaction using data X for writing.
150	....	....	....
151		WRITE (X)	Record W-timestamp(X) = 151

(c) Otherwise

WRITE (D) is executed and W-timestamp (D) = TS (T_j)

TS	T_i	T_x	Operations
150	....	....	....
151	WRITE (X)	....	Record W-timestamp(X) = 151

- Compare to Basic Timestamp ordering
  - Unlike, timestamp protocol, Thomas write rule asks to ignore write operation if T_i issues WRITE (D) and TS (T_i) < W - timestamp (D).

Review Questions

Q. 1 What do you understand by concurrency control ? Explain view serializability and conflict serializability with proper example.

Q. 2 Explain Two phase locking.

Q. 3 Explain time stamp based protocols.

Q. 4 Explain two phase locking protocol. Also list advantages of this protocol.

❑❑❑

CHAPTER 12

Recovery System

Module VI

Syllabus

Failure Classification, Log based recovery, ARIES, Checkpoint, Shadow paging, Deadlock handling.

12.1 Database Recovery Concepts - System Recovery

Q. Explain concept of database recovery and also explain the techniques. (4 Marks)
Q. Explain method of database recovery. (4 Marks)

1. Database recovery is the process of restoring the database to original (correct) state as it was before database failure occurs.
2. The process of solving any type of database failures, quickly and without data loss and keep database available is called database recovery.
3. The main element of database recovery is the most recent database backup. If you maintains database backup efficiently, then database recovery is very straight forward process.

Example :

To recover data from system having full backup option is as below,

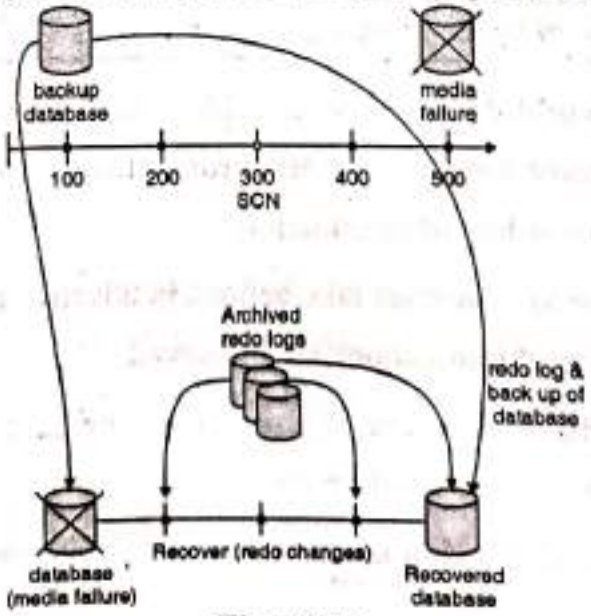


Fig. 12.1.1

Restoring entire system to a certain point may require time, depends on when last full backup taken and incremental backups which covers the period of time between the full backup and restore point.

4. **To understand concept of database recovery we need to understand terms**
   (a) Database Backup
   (b) Database Failure

5. **Database recovery algorithms**
   (a) During normal transaction executions ensure that enough information is backed up to allow recovery from possible failures.
   (b) Data should ensure that database consistency, transaction atomicity, and durability.

6. **Types of database recovery / recovery phases**
   (a) Forward database recovery
   (b) Backward database recovery

7. **Database recovery techniques**
   (a) Log based recovery
   (b) Shadow paging recovery
   (c) Checkpoints

12.2 Database Recovery Concepts - System Failure

Q. Explain various types of system failure. (4 Marks)

- Like any other real world database systems also suffer from failure from a variety of causes: disk crash, power outage, software error, a fire in the machine room even rain. The result of any failure is loss of information.
- Therefore, the database system must take actions in advance to ensure that the atomicity and durability properties of transactions are preserved.
- An integral part of a database system is a recovery scheme that can restore the database to the consistent state that existed before the failure.
- Provide high availability; that is, it must minimize the time for which the database is not usable after a crash.

12.2.1 Failure Classification

Q. Explain system failure classification. (4 Marks)

A computer system, like any other electrical or mechanical system is tends to failure. There are many causes, including disk crash, power failure, software errors, a fire in the machine room, or even sabotage. Whatever the cause, information may be lost. There are various types of failure that may occur in a system, each of these needs to be dealt with a different manner.

1. **Hardware Failure / System crash**

- There is a hardware malfunction that causes the loss of the content of volatile storage, and brings transaction processing to a halt.
- The content of non volatile storage remains intact, and is not corrupted or changed.

2. **Software Failure**

- The database software or the operating system may be corrupted or failed to work correctly, that may causes the loss of the content of volatile storage, and brings about database failure.

3. **Media failure**

- A disk block loses its content as a result of either a head crash or failure during a data-transfer operation.
- Copies of the data on other disks such as tapes, CDs are used to recover from the failure.

4. **Network Failure**

- The problem with network interface card can cause network failure.
- There may be problem with network connection.

5. **Transaction failure**

There are two types of errors that may cause a transaction to fail :

(a) Logical error

The transaction can no longer continue with its normal execution because of some internal condition, such as wrong input values, data not found in database, data overflow, or resource limit exceeded etc.

(b) System error

The system has entered an undesirable state like deadlock; as a result transaction cannot continue with its normal execution.

6. Application software error

- The problem with software accessing the data from database.
- This may cause database failure as data cannot be updated using such application to it..

7. Physical disasters

The problem caused due to flood, fire, earthquake etc..

8. Application software error

These are some logical errors in the program that is accessing database, which cause one or more transactions failure.

12.3 Log-Based Recovery

Q.	Explain concept of log and log based recovery algorithm.	(4 Marks)
Q.	How does log based protocol works ?	(4 Marks)
Q.	Write short notes Log Based Recovery.	MU - Dec. 18, 10 Marks

(1) Introduction

- There can be problem in accessing database due to any reason causes a database system failure.
- The most widely used structure for recording database modifications is **Transaction log (or log).**
- The log is a sequence of log records, recording all the update activities done on the database by all database users.

(2) Transaction log

- Transaction log records are maintained to record various events during transaction processing.
- Transaction log file is recorded when transaction performs a write operation to record data changes.
- Log records to be useful for recovery from system and disk failures, the log must reside in stable storage.
- We assume that every log record is written to the end of the operation on stable storage as soon as log is created.
- Transactional log contains following data :

(i) **Transaction Identifier (T_i) :** This is the unique identifier of the transaction that is recorded at write operation.

(ii) **Data item Identifier(X) :** unique identifier to recognize data item written.

(iii) **Old Value (V_1) :** Value of the old data item.

(iv) **New Value (V_2) :** Value of new data item after the write is done.

- There are several types of log records.

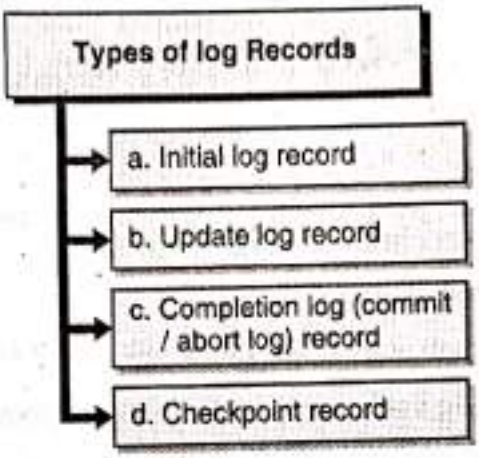


Fig. 12.3.1 : Types of log records

(a) Initial log record

- To start a transaction initial transaction log is recorded.
- This log indicates that recording log file is started.
- Log Record :< T_n Start>
- Transaction (T_n) has started.

Example :

<T_1 Start> : Transaction T_1 is started.

(b) Update log record

- An update log record describes a single database write.
- It also includes the value of the bytes of page before and after the page change.
- **Log record :**< T_n, X, V_1, V_2>
- Transaction (T_n) has performed a write on data item X.
- It modify current value V_1 of data item X to updated value V_2 after the write.

Example :

<T_1 , A, 100,500> : Transaction T_1 changed value of A to 500.

Or <T_1 , A, 500> : Transaction T_1 changed value of A to 500.

(c) Completion log (commit / abort log) record

- If transaction completes operations successfully then commit a transaction or if any problem while executing transaction decision to abort and hence rollback a transaction.
- **Operational Update Log :** $< T_n$ Commit $>$ or $< T_n$ Abort $>$
- It commits or rolls back the operations performed by transaction.

Example :

$<T_1$ Commit$>$: Transaction T_1 is committed to server.

Or $<T_1$ Rollback$>$: Transaction T_1 abort its operations.

(d) Checkpoint record

- Records a point when checkpoint has been made.
- These are used to speed up recovery.
- It also record information that eliminates the need to read a log's past.
- The time of recording varies according to checkpoint algorithm.
- **Log record :** $< T_n$ Checkpoint A $>$
- It marks transaction status.

Example :

$<T_1$ Checkpoint A$>$: Transaction T_1 is committed to server.

(3) Recovery actions

Redo operation

- If a transaction has recorded commit operation before failure occurs, then transaction must be redone.
- Recovery Action : **REDO** (T_i)

Example :

REDO (T_1)

Undo operation

- If a transaction has not recorded commit operation before failure occurs, then transaction must be undone.
- Recovery Action : **UNDO** (T_i)

Example : UNDO (T_1)

12.3.1 Deferred-Modification Technique (REDO Algorithm)

Q. Explain Deferred log based in detail. (4 Marks)
Q. What is deferred modification technique ? (4 Marks)

(1) Introduction

- The deferred-modification technique will record database modifications in the transaction log files after transaction partially commits.
- **Partial commit :** When the final action of the transaction has been executed a transaction said to be partially committed.
- The version of the deferred-modification technique that we describe assumes that serial execution of transactions.
- Transaction perform partially commit,
  - Transaction log is recorded.
  - If the system crashes before the transaction commit or if the transaction aborts, then the information on the log will not be recorded.

(2) Working

The execution log based recovery of transaction T_n as follows,

(a) T_n starts its execution by writing < **T_n start**> to transaction log file.

(b) If Transaction performs (T_n) performs write(X) operation it will record a **new operational log.**

(c) If T_n **commits data to server,** than a record < T_n commit> is written to transaction log.

(d) Ensure that all the transactional log record are written to stable storage. As it is possible to that database may fail at any point of time.

(e) Once they have been written, the actual updating takes place to server then it will be in committed state.

(3) Example

- Consider a simple banking system. Let T_0 be a transaction that transfers Rs.50 from account A to account B :

T_0
read(A)
A :=A -50
write(A)
read(B)
B:= B + 50
write(B)

- Let T_1 be a transaction that withdraws Rs. 100 from account C.

T_1
read(C)
C:= C + 100
write(C)

- Suppose that these transactions are executed serially, in the order T_0 followed by T_1, and that the values of accounts A, B, and C before the execution took place were Rs.100, Rs.100, and Rs.100, respectively.
- The portion of the log containing the relevant information on these two transactions appears as below,

<T_0 start>
<T_0, A, 50>
<T_0,B, 150>
<T_0 commit>
<T_1 start>
<T_1, C, 200>
<T_1 commit>

Fig. 12.3.2 : Database log corresponding to T_0 an T_1

- There are various orders in which the actual outputs can take place to both the database system and the log as a result of the execution of T_0 and T_1. as shown above.
- The value of A is changed in the database only after the record <T_0, A, 50> has been placed in the log.

Log	Database
<T_0 start>	
<T_0, A, 50>	
<T_0, B 150>	

Log	Database
<T_0 Commit>	
	A = 50
	B = 150
<T_1 Start>	
<T_1 C, 200>	
<T_1 Commit>	
	C = 200

Fig. 12.3.3

- Transaction log are used to handle any failure that results in the loss of data.
- The recovery scheme uses the following recovery procedure.

(I) Method 1

- Use two lists of transactions: the committed transaction since the last checkpoint and the active transaction.
- Apply the REDO operation to all the WRITE operations of the committed transactions from the log in the order in which they were to the log.
  - Restart the active transaction. REDO(T_n) sets the value of all data items updated by transaction T_n to the new values.
  - The data items updated by T_n and their new value is updated to the log file.
  - The redo operation must be executing once.
  - After a failure, the recovery system contacts log based recovery to check for recovery action.
  - Transaction T_n will redone if and only if the log contains **both** the record < T_n start> and the record < T_n commit>.
- Thus, if the system crashes after the transaction completes its execution, the recovery scheme uses the information in the log to restore the system to a previous consistent state after the transaction had completed.
- Let us return to our banking example with transactions T_0 and T_1 executed one after the other in the order T_0 followed by T_1. Fig. 12.3.3 shows the log those results from the complete execution of T_0 and T_1.

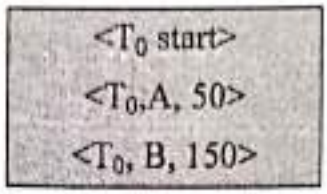


Fig. 12.3.4(a) : Same log at different times - no commit recorded

(II) Method 2

(A) System crashes before the completion of the transactions

(a) When the system starts after failure, no redo action is required. As, no commit record appears in the log.

(b) All data values of A & B remain same.

(c) The log records of the incomplete T_0 can be deleted from the log.

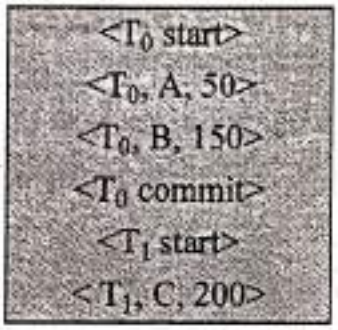


Fig. 12.3.4(b) : < T_0 commit > recorded

(B) The crash comes just after the log record

(a) For the step WRITE(C) of transaction T_1 has been written to stable storage.

(b) When the system comes back up, the operation redo (T_0) is performed since the record < commit> appears in the log on the disk.

(c) After this operation is executed T_0, the values A and B are Rs.50 and Rs.150, respectively.

(d) The value of account C remain before, the log records of the incomplete transaction T_1 can be deleted from the log.

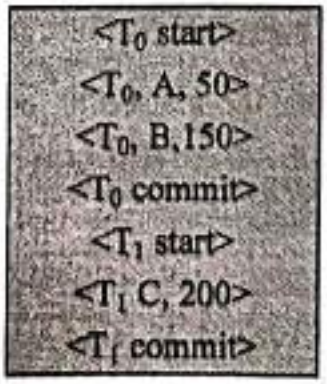


Fig. 12.3.4(c)

(C) A crash occurs just after the log record < T_1 commit > is written to is stable storage

(a) The log at the time of this crash is as in Fig.12.3.4(c) the system comes back up.

(b) Two commit records are in the log: one for T_1 and one for T_0.

(c) Therefore, the system must perform operations redo (T_0) and redo (T_1) in the order in which their commit records appear in the log.

(d) After the system executes these operations, the values of accounts A, B, and C, are Rs.50, Rs.150, and Rs.200.

Finally, let us consider a case in which a second system crash occurs after recovery from the first crash.

(4) Summary

- For each commit record <T_n commit>found in the log, the system performs the operation redo (T_n). In other words, it restarts the recovery actions from the beginning.
- Since redo writes values to the database independent of the values currently in the database, the result of a successful second attempt at 'redo' is the same as though 'redo' had succeeded the first time.

12.3.2 Immediate-Modification Technique (UNDO Algorithm)

Q. Explain Immediate log base in detail.	(4 Marks)
Q. What is immediate modification technique.	(4 Marks)

(1) Introduction

- The **immediate-modification technique** writes database modifications to be written to the database as soon as it is performed or in the active state.
- In the event of transaction failure, the system makes use of old-value of the log records to restore the data items.

(2) UNDO operation

- Before a transaction T_n starts its execution, the system writes the record <T_n start> to the log.
- During transaction execution, any **WRITE(X) operation by T_n** is preceded by the writing of the appropriate new update record to the log.
- When **T_n partially commits**, the system writes the record <T_n commit> to the log.
- Since the information in the log is used in restoring the original database state, we cannot

allow the actual update to the database to take place before the corresponding log record is written out to stable storage.

- We therefore require that, before the execution of an output (B) operation its log records is written to stable storage.
- This procedure is defined as follows :
  1. Use two lists of transactions maintained by the system;
     (a) Committed transactions since the last checkpoint
     (b) Active transactions
  2. Undo all the WRITE operations of the active transaction from the log, using the UNDO procedure
  3. Redo the WRITE operations of transaction which are committed.
- UNDO Operation

  Undo WRITE operation : It consists of check its log entry of write T and reading Old value and setting the value of item X in the database to old value.

(3) Example :

- Consider a simple banking system, with transactions T_0 and T_1 executed one after the other in the order T_0 followed by T_1.
- Fig. 12.3.5(a) shows possible order of execution in both the database and the log as a result of the execution of T_0 and T_1.

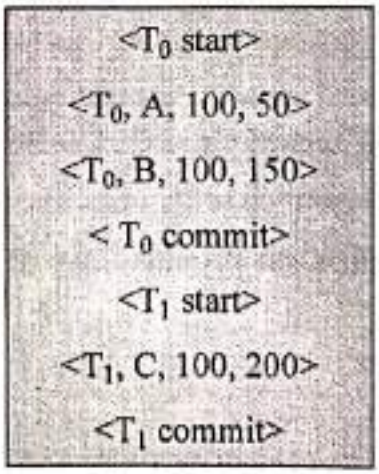


Fig. 12.3.5(a)

Log	Database
<T_0 start>	
<T_0, A, 1200, 950>	
<T_0, B, 2000,2050>	

Log	Database
	A = 950 B = 2050
<T_0 commit> <T_1 start>	
<T_1, C, 700, 800> < T_1 commit >	C = 800

Fig. 12.3.5(b)

- Using the log file, the system can handle any failure that does not result in the loss of information in non volatile storage.
- The recovery scheme uses two recovery procedures.

(I) Method 1

(a) Undo(T_n) restores the value of all data items updated by transaction T_n to the old values and redo(T_n) sets the value of all data items updated by transaction T_n to the new values.

(b) The set of data items updated by T_n and their respective old and new values can be found in the log.

(c) The undo and redo operations must guarantee to correct behaviour, even if a failure occurs during the recovery process.

(d) After a failure has occurred, the recovery scheme consults the log to determine which transactions need to be redone, which need to be undone :

- Transaction T_n needs to be undone if the log contains the record < T_n start>, but does not contain the record <T_n commit>.
- Transaction T_n needs to be redone if the log contains both the record < T_n start> and the record <T_n commit>.

- In our banking example, with transaction T_0 and T_1 executed one after the other in the order T_0 followed by T_1.

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>

Fig. 12.3.6(a)

(II) Method 2

(a) If system crashes before the completion of the transactions

- First, let us assume that the crash occurs just after the log record for the step of transaction T_0 has been written to stable storage.
- When the system comes back up, it finds the record <T_0 start> in the log, but no corresponding <T_0 commit> record.
- Thus, transaction T_0 must be undone, so an undo (T_0) is performed.
- As a result, the values in accounts A and B (on the disk) are restored to Rs.100 and Rs.100, respectively.

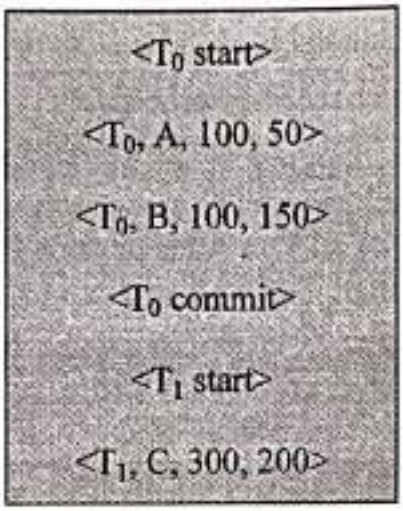


Fig. 12.3.6(b)

(b) If crash comes occurs after the log record for the step write (C) of transaction T_1 has been written to stable storage

Recovery Action

(i) UNDO(T_1)

Record <T_1 start> appears in the log, but there is no commit record for transaction <T_1 commit>. So, Transaction needs to be Undone its effects.

(ii) REDO(T_0)

- Log contains both the record <T_0 start> as well as <T_0 commit>. Than transaction should be written again to show its effect on transaction.

- In this example, the same outcome would result if the order were reversed.

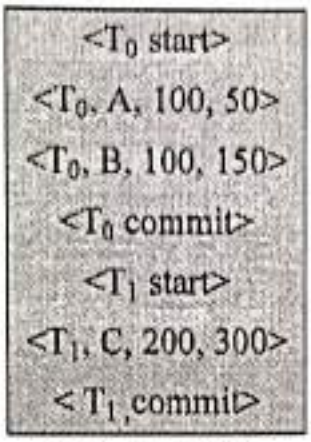


Fig. 12.3.6(c)

(c) If crash occurs just after the log record <T_1 commit> has been written to stable storage.

- When the system comes to backup, both T_0 and T_1 need to be redone, since the records <T_0 start> and <T_0 commit> appear in the log, as do the records <T_1 start> and <T_1 commit>.
- After the system performs the recovery procedures REDO(T_0) and REDO(T_1), the values in accounts A, B, and C, are Rs.50, Rs.150, and Rs.300, respectively.

12.4 Recovery Related Structures - Checkpoint

Q.	Explain concept of checkpoint with suitable example.	(4 Marks)
Q.	What is checkpoint ? What is its use ?	(4 Marks)
Q.	How does checkpoint works ?	(4 Marks)
Q.	What are advantages and disadvantages	(4 Marks)

(1) Introduction

- A database checkpoint is where all committed transactions are written to the redo/audit logs.
- The database administrator determines the frequency of the checkpoints based on volume of transactions.
- When a system fails, It check log to determine recovery action.
- Too frequent checkpoints can affect the performance.

(2) Problems in this approach

- The search process is time-consuming.

- Most of the transactions that, according to our algorithm, need to be redone as they have already written their updates into the database.

(3) Need of check points

- To record status of transaction execution, the system maintains the log, using one of the two techniques.
- The system periodically performs checkpoints, with following sequence of actions :
  1. Output all log records onto stable storage which are currently stored in main memory.
  2. Output to the disk.
  3. Output onto stable storage a log record <checkpoint>.
- Transactions are not allowed to perform any update actions, such as writing to a buffer block or writing a log record, while a checkpoint is in working state.
- The presence of a <checkpoint> record in the log allows the system to restructure its recovery procedure.

(4) Working

- Consider a transaction T_n that committed prior to the checkpoint.
- For such a transaction, the <T_n commit> record appears in the log before the <checkpoint> record.
- Any database modifications made by transaction T_n must have been written to the database either prior to the checkpoint or as part of the checkpoint itself. Thus, at recovery time, there is no need to perform a redo operation on T_n.
- After a database failure has occurred, the recovery scheme examines the log to determine the most recent transaction T_n that started executing before the most recent checkpoint took place.
- It can find such a transaction by searching the log backward, from the end of the log, until it finds the first <checkpoint> record (since we are searching backward, the record found is the final <checkpoint> record in the log); then it continues the search backward until it finds the next <T_n start> record. This record identifies a transaction T_n.
- Once the system has identified respective transaction T_n, the redo and undo operations need to be applied to only for transaction T_n and all transactions that started executing after transaction T_n.
- The exact recovery operations to be performed depend on the modification technique being used.
- For the immediate-modification technique, the recovery operations are :

- For all transactions T_k in T that have no <T_k commit> record in the log, execute UNDO(T_k).
- For all transactions T_k in T such that the record <T_k commit> appears in the log, execute REDO(T_k).

- Obviously, the undo operation does not need to be applied when the deferred-modification technique is being employed.
- Consider the set of transactions {T_0, T_1 .... , T_{10}} executed in the order of the subscripts. Suppose that the recent checkpoint took place during the execution of transaction T_6. Thus, only transactions T_6, T_7, ... , T_{10} need to be considered during the recovery scheme. Each of them needs to be redone if it has committed; otherwise, it needs to be undone.

(5) Advantages

1. A database checkpoint keeps track of change information and enables incremental database backup.
2. A database storage checkpoint can be mounted, allowing regular file system operations to be performed.
3. Database checkpoints can be used for application solutions which include backup, recovery or database modifications.

(6) Disadvantages

1. Database storage checkpoints can only be used to restore from logical errors (E.g. a human error).
2. Because all the data blocks are on the same physical device, database storage checkpoints cannot be used to restore files due to a media failure.

12.5 Shadow Paging

Q. Explain concept of shadow paging. (4 Marks)
Q. Explain page table using shadow paging and describe process of recovery. (4 Marks)

(1) Introduction

- It is not always convenient to maintain logs of all transactions for the purposes of recovery.
- An alternative is to use a system of shadow paging.
- This is where the database is divided into pages that may be stored in any order on the disk.

– In order to identify the location of any given page, we use something called a **page table**.

(2) Method

(a) During the life of a transaction two page tables are maintained as below,

(i) Shadow page table

(ii) Current page table.

(b) When a transaction begins both of these page tables point to the same locations (are identical).

(c) During the lifetime of a transaction the shadow page table doesn't change at all.

(d) However during the lifetime of a transaction update values etc. may be changed.

(e) For pages updated by the transaction, two versions are kept. The old version is referenced by the shadow directory and the new version by the current directory.

(f) So whenever we update a page in the database we always write the updated page to a new location.

(g) This means that when we update our current page table it reflect the changes that have been made by that transaction.

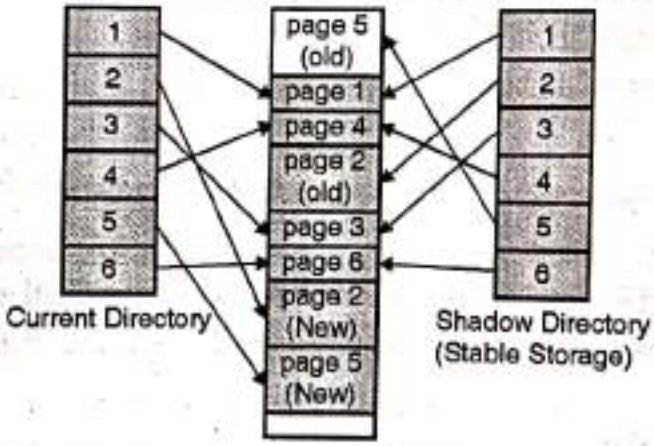


Fig. 12.5.1 : Pagetable-shadow paging

(h) As we can see the shadow page table shows the state of the database just prior to a transaction, and the current page table shows the state of the database during or after a transaction has been completed.

(3) Process of recovery

– We now have a system whereby if we ever want to undo the actions of a transaction all we have to do is recover the shadow page table to be the current page table.

– As such this method makes the shadow page table particularly important, and so it must always be stored on stable storage.

- On disk we store a single pointer location that points to the address of the shadow page table.
- This means that to swap the shadow table for the current page table (commiting the data) we just need to update this single pointer (very unlikely to fail during this very short fast operation).
- In case of no failure means while committing transaction just discard the shadow directory.
- In case of multi-user environment with concurrent transaction, logs and checkpoints must be incorporated in shadow paging.

(4) Advantages

- Shadow page method does not require any Undo or Redo algorithm for recovery purpose.
- Recovery using this method will be faster.
- No overhead for writing log records.

(5) Disadvantages

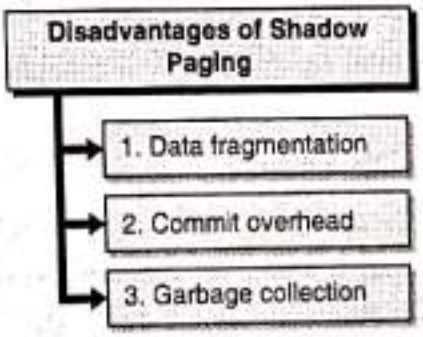


Fig. 12.5.2 : Disadvantages of shadow paging

1. Data fragmentation

The main disadvantage of this technique is the updated Data will suffer from fragmentation as the data is divided up into pages that may or not be in linear order for large sets of related hence, complex storage management strategies.

2. Commit overhead

If the directory size is large, the overhead of writing shadow directories to disk as transaction commit is significant.

3. Garbage collection

- Garbage will accumulate in the pages on the disk as data is updated and pages lose any references. For example if i have a page that contains a data item X that is replaced with a new value then a new page will be created. Once the shadow page table is updated nothing will reference the old value of X.

- The operation to migrate between current and shadow directories must be implemented as an atomic mode.

12.6 ARIES - Algorithm

Q. Explain ARIES algorithm with different steps. Enlist all advantages of ARIES algorithm **(6 Marks)**

1. Introduction

- ARIES is a recovery algorithm that is designed for no force type of backup approach.
- Recovery manager is generally called when there is a crash.
- Restart can be proceeded in three different phases as below

2. Principles of ARIES algorithm

(a) Write-ahead logging

- Any change to a database object is first recorded in some log file.
- The record in the log must be written to any of stable storage before the change in database is written to disk.

(b) Repeating history during Redo

- ARIES finds all operations done by DBMS before the crash and restores system back to the same state that it was in at the time of the crash.
- Then, it aborts all the actions of transactions those are still there in active state at the time of the crash.

(c) Logging changes during Undo

- We make changes to the database during restoring database all transactions are logged in same order.
- So, it ensures same action is not repeated in the event of repeated restarts.

3. Phases of ARIES algorithm

(a) Analysis Phase

- It first finds dirty pages(data changes those are not committed to database) in the available buffer pool
- It also identifies all active transactions at the time of the system crash.
- Identify Redo LSN from which redo should start.

(b) Redo

- In order to restore database system will repeats all actions performed on database from start of log or from any selected point in log or from RedoLSN.

- Then it restores the database state to state at which it was at the time of the system crash.
- RecLSN and RedoLSN avoid redo action already reflected on page.

(c) **Undo**

- It reverts or undoes all operations of transactions which are not committed.
- So after above action now database only reflects actions which are committed transactions.

4. Example

a. Consider the crash recovery example illustrated in Fig 12.6.1.

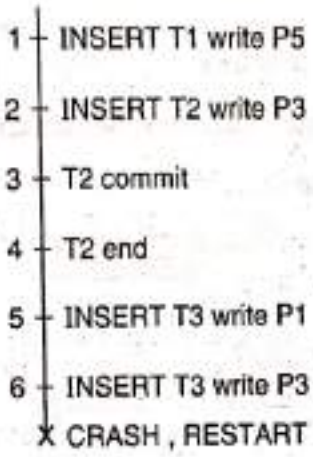


Fig. 12.6.1 Example of crash recovery using ARISE

b. When the system is restarted, the Analysis phase identifies transactions T1 and T3 are active at the time of the crash, and therefore to be rollback or revert.

c. Transaction T2 is committed; therefore, it needs to be written to disk; and P1, P3, and P5 are may be dirty pages.

d. All the update operations (including those of T1 and T3) are applied once again in the same order as shown during the Redo phase.

e. Then, the actions of T1 and T3 are reverted in reverse order during the as in Undo phase; means first T3's write of P3 is undone and then T3's write of P1 is undone, then finally T1's write of P5 is undone. As only T_2 is committed.

5. Advantages of ARIES algorithm

(a) ARIES algorithm is simple and flexible as compared to other recovery algorithms.

(b) ARIES support concurrency control protocols

(c) Fine locking at lower granulation.

(d) Independent recovery of every page.

(e) Transaction records save points and roll back up to that rollback.

12.7 Concept of Deadlock

Q. What is database deadlock? Explain various deadlock handling techniques. **(4 Marks)**
Q. What is dead lock ? **(4 Marks)**
Q. Define Deadlock. Explain Deadlock Detection, Prevention and Recovery.
MU - Dec. 18, 10 Marks

1. A system is said to be in a state of deadlock if there exists a set of transactions such that every transaction in the set is waiting for another transaction to complete its execution.

Example :

Consider two transactions given below,

T_1 : This transaction first read and then write data on data item X, then after that performs read and write on data item Y.

T_1
Read (X)
Write (X)
Read (Y)
Write (Y)

T_2 : This transaction first read and then write data on data item Y, then after that performs read and write on data item X.

T_2
Read (Y)
Write (Y)
Read (X)
Write (X)

Consider above two transactions are executing using locking protocol as below,

T_1	T_2	
Lock-X(X)		
Read (X)		
Write (X)		
	Lock-X(Y)	

T_1	T_2	
	Read (Y)	
	Write (Y)	
Lock-X(Y)		Wait for transaction T_2 to Unlock Y
Read (Y)		
Write (Y)		
	Lock-X(X)	Wait for transaction T_1 to Unlock X
	Read (X)	
	Write (X)	

So in above schedule consider two transactions given below transaction $\mathbf{T_1}$ is waiting for transaction $\mathbf{T_2}$ to Unlock data item Y and transaction $\mathbf{T_2}$ is waiting for transaction $\mathbf{T_1}$ to Unlock data item X.

So system is in a deadlock state as there are set of transactions $\mathbf{T_1}$ and $\mathbf{T_2}$ such that, both are waiting for each other transaction to complete. This state is called as **Deadlock state**.

2. There are **two principle methods** to handle deadlock in system,

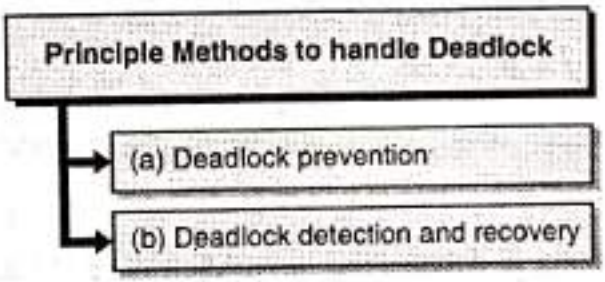


Fig. 12.7.1 : Principle Methods to handle Deadlock

(a) Deadlock prevention

We can use deadlock prevention techniques to ensure that the system will never enter a deadlock state.

(b) Deadlock detection and recovery

We can allow the system to enter a deadlock state and then we can detect such state by **deadlock detection techniques** and try to recover from deadlock state by using and **deadlock recovery scheme**.

3. Both of above methods may result in transaction rollback.
4. Deadlock prevention is commonly used if the probability that the deadlock occurs in system is high else detection and recovery are more efficient if probability of deadlock occurrence is less.

12.7.1 Approaches for Deadlock Prevention

Q. Explain dead lock prevention. (4 Marks)

(1) Approach 1

A simplest form, in which a transaction acquires lock on all data items which will be required by transaction at the start of execution. It is effective as other transactions can't hold lock on those data items till first unlocks data item.

Disadvantages

(i) It is difficult to know in advance which data items need to be locked.

(ii) Data utilization is very low as may be unused data items locked by transaction.

(2) Approach 2

This approach uses timestamp ordering of data items. It is something like tree protocol and every transaction have to access data item in given sequence only. The variation of this approach with two phase protocol assures deadlock prevention. Order of data items must be known to every transaction.

(i) To have concurrency control, two phase locking protocol is used which will ensure locks are requested in right order.

(ii) Timestamp ordering is determined by validation (T_i) i.e. TS (T_i) = validation (T_i) to achieve serializability.

(iii) The validation test for transaction T_j requires that for all transaction T_i with TS (T_i) < TS (T_j) then one of the following conditions must be hold.

(a) Finish (T_i) < start (T_j) as T_i finishes before T_j starts the serializability should be maintained.

(b) T_i completes its write phase before T_j starts its validation phase (Start (T_j) < finish (T_i) < Validation (T_j)).

(iv) This ensures writes of both transactions do not take place at same time.

(v) Read of T_j not affected by writes of T_j and T_j can't affect read of T_i so serializability is maintained.

(3) Approach 3 : Prevention and transaction rollbacks

Pre-emptive technique

(i) Pre-emption means, if transaction T_i wants to hold lock on data item hold by T_j. then system may preempt (UNLOCK all previous locks) T_i by rolling it back and granting lock to T_j on that data item.

(ii) To control this pre-emption every transaction is assigned a unique timestamp.

(iii) System uses this timestamp to decide whether to wait or rollback the transaction.

(iv) The transaction retains its old time stamp if it is rollback and restarted.

(v) Various deadlock prevention techniques using timestamps are as follows :

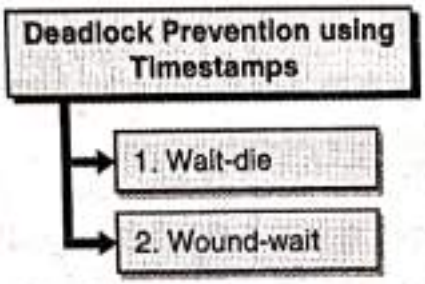


Fig. 12.7.2 : Deadlock Prevention using Timestamps

1. Wait-die

- This is non pre-emptive technique of deadlock prevention.
- When transaction T_i wants to hold data item, currently hold by T_j, then T_i is allowed to wait if and only if it is older than T_j otherwise T_i is rolled back (die).
- If T_1 requests for data item hold by T_2 then as TS (T_1) < TS (T_2) so T_1 should wait.

2. Wound-wait

- This is preemptive technique of deadlock prevention.
- When T_i wants to hold data item, currently hold by T_j, then T_i is allowed to wait if and only if T_i is younger to T_j otherwise T_j is rollback (wounded). Consider T_1, T_2, T_3 transactions.
- If T_1 requests for data item hold by T_2 then data item is pre-empted from T_2 and given to T_1 and T_2 is rollback.
- If T_3 requests for data item hold by T_2 then T_3 need to roll back.

(vi) Prevention may lead to starvation of transaction if they rollback again and again. But both schemes wait-die and wound-wait avoid starvation by making use of timestamps.

12.7.2 Deadlock Detection and Recovery

Q. Explain deadlock detection. (4 Marks)

Q. Explain recovery from dead lock. (4 Marks)

(1) Introduction

(i) When system is having deadlock detection and recovery techniques, the detection is done periodically to check whether the system is in deadlock, if yes then the recovery techniques are used to resolve this deadlock.

(ii) This can be done with help of some deadlock detection algorithms.

(iii) To achieve this, system must do the following :

(a) System should maintain information about current data items allocation to transactions and requests to be satisfied.

(b) Algorithm that uses this information to detect deadlock.

(c) Recovery techniques to be applied after detection of deadlock.

(2) Deadlock detection

(i) Deadlock can be detected using directed graph called as **wait-for-graph.**

(ii) The graph G = (V, E) can be seen as V is of vertices i.e. set of transaction in execution concurrently and E is set of edges.

(iii) Such that edge $T_i \rightarrow T_j$ if $T_i \rightarrow T_j$ is present in graph it shows that, transaction T_j is waiting for transaction T_i to release a data item it needs.

(iv) If cycle is present in wait-for-graph then deadlock is present and transactions in cycle are deadlock.

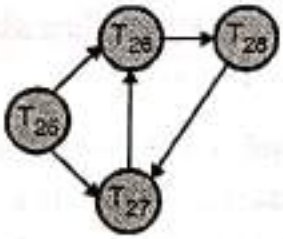


Fig. 12.7.3 : Wait-for graph with a cycle

(v) To detect deadlock, system must maintain wait-for-graph and periodically invoke an algorithm that searches cycle in the graph.

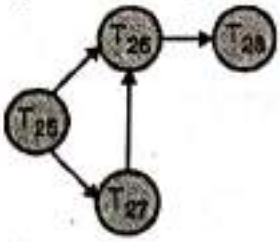


Fig. 12.7.4 : Wait-for graph with no cycle

(vi) If no cycle is present it mean no deadlock in system.

(vii) If deadlock occurs frequently, then detection algorithm should be invoked more frequently problem in this scenario is the data items locked by deadlock transaction will be unavailable until the deadlock is problem.

(viii) This may tend to more cycles in graph degrading capacity of granting lock requests.

(3) Recovery from deadlock

- When deadlock is detected in the system, then system should be recovered from deadlock using recovery schemes.
- A most common solution is rollback one or more transaction to break the deadlock.
- Methods for recover from deadlock,

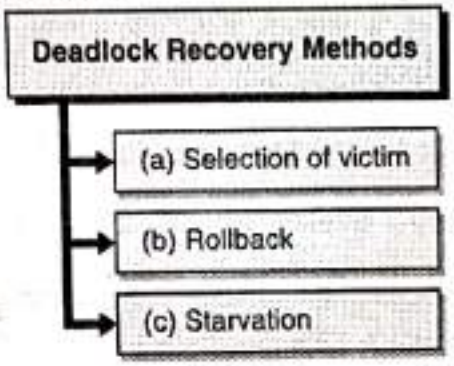


Fig. 12.7.5 : Deadlock Recovery Method

(a) Selection of victim

- If deadlock is detected then a transaction one or more transactions (victims) to be selected to break the deadlock.
- Transactions with minimum cost should be selected for rollbacks.
- Cost can be detected by following factors.
  - For how much time transaction has computed and how much time it needs to do.
  - How many data items it has locked.
  - How many data items it may need ahead.
  - Number of transaction to be rollback.

(b) Rollback

- Once victims are decided then there are two ways to do so :
  - (i) **Total rollback :** The transaction is aborted and then restart its.
  - (ii) **Partial rollback :** Rollback the only transaction which is needed tobreak the deadlock.
- But this approach needs to record some more information like state of running transactions, locks on data item held by them, and deadlock detection algorithm

specifies points up to which transaction to be rollback and recovery method has to rollback.

- After sometime transaction resume; partial transaction.

(c) Starvation

- It may happen every time same transaction is selected as victim and this may lead to starvation of that transaction (minimum cost transaction it selected every time).
- System should take care that every time same transaction should not be selected as victim. So it will not be starved.

Review Questions

Q. 1 What is recovery ? Explain types of recovery? (Forward, Backward Recovery)

Q. 2 What is log based recovery ? Explain all log based recovery technique with example?

Q. 3 What is checkpoint? How is checkpoint information is used in recovery operation following System crash explain with example?

Q. 4 List and explain the three steps of Checkpointing in ARIES.

Q. 5 Describe the use of logs and check points in a database.

Q. 6 What is deadlock? What are its prevention and avoidance methods?

Q. 7 Explain wait-die and wound-wait for deadlock prevention.

❑❑❑

About the Author

Prof. Mahesh Mali is assistant professor and research scholar performing his Doctoral research. He holds **Masters of Computer Engineering** and **Bachelors of Computer Engineering** degree from Mumbai University with **distinction**. Has wide teaching experience of more than 15 Years. He has a **good experience of IT industry** as a Database Developer for 3 years with Mastek Ltd. He was working as visiting faculty with Xavier's Institute, St. Francis Institute, St. John College, SIWS (Wadala), Sardar Patel Institute, Vidyalanka[illegible]nd many other institutes of good repute. He is **examination panel [illegible]mber** of Mumbai and NMIMS University.

- Authored **61 Books** and **5 Research Papers** in International Journals.
- Winner of **"INSPIRE : Teaching Excellence Award"** for year 2013 and 2014 from Infosys Ltd. for his contribution in field of education.
- Appointed as Judge for interstate science exhibition by government of Maha[illegible]ra.
- **Oracle certified** International Trainer for databases and also **SAS** certified International Trainer for **SAS Data Analytics.**
- Guided many projects of Degree, Diploma and Post-Graduation engineeri[illegible]uents.

Tech Knowledge Publications
Head Office :
B/5, First Floor, Maniratna Complex, Taware Colony, Aranyeshwar Corner,
Pune - 411009. Maharashtra State, India. Tel. : 91-20-24221234, 91-20-24225678

Distributors

Student's Agencies (I) Pvt. Ltd. T. : (022) 40496161, 91672 90777
Vidyarthi Sales Agencies T. : (022) 23867279, 98197 76110
Bharat Sales Agency T. : (022) 23819359, 86572 92797

Our Branches : Pune | Mumbai | Kolhapur | Nagpur | Solapur | Nashik

For Library Orders Contact - Ved Book Distributors M : 80975 71421 / 92208 77214

Email : info@techknowledgebooks.com | Website : www.techknowledgebooks.com | Like us at. TechknowledgePublications

ISBN : 978-93-89424-23-2
Price ₹ 225/-
MO44A