Database Management System

Chapter 1 : Introduction Database Concepts

Q. 1 Write a short note on : Data.

Ans. : Data

The facts and figures that can be recorded in system and that have special meaning assigned to it. The system 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.

Q. 2 Write a short note on : Database.

Ans. : Database

A database is a collection of data items stored in one place and having some common base (Background) between them. Like a college database contains 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 (Data + Base).

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 : Sample Student Database

Q. 3 Discuss Database system.

Ans. : Database System

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 to 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 IS is also called as Database Management System (DBMS).

Examples

1. MS Access, Fox Pro by Microsoft.
2. Oracle by Oracle corp.
3. SQL Server By Microsoft.
4. Ingres, DB2 by IBM.

Q. 4 Explain the features of DBMS.

Ans. : Features of DBMS

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

1. Data integrity

Integrity constraints provide a way of ensuring that changes made to the database by authorized users that do not result in a loss of data consistency and correctness. Database integrity concern 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 to database can be reverted back with help of rollback command. The changes can be saved successful with 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.

Q. 5 Discuss the advantages of Database system over Files system.

Ans. :

Advantages of DBMS over Files system

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 :** With the usage of database, it is assured 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. 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.

Q. 6 Discuss different database Users.

Ans. : 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.

Chapter 2 : Database Architecture

Q. 1 Explain three-level architecture of DBMS.

Ans. : Database architecture

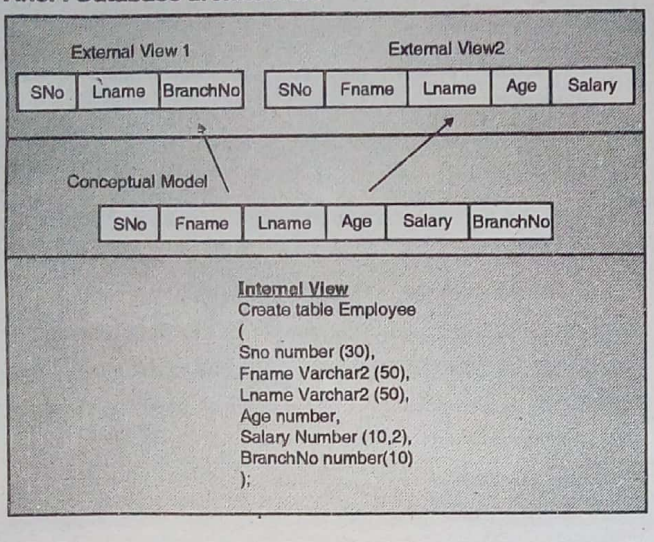


Fig. 2.1 : Database schema levels

(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.

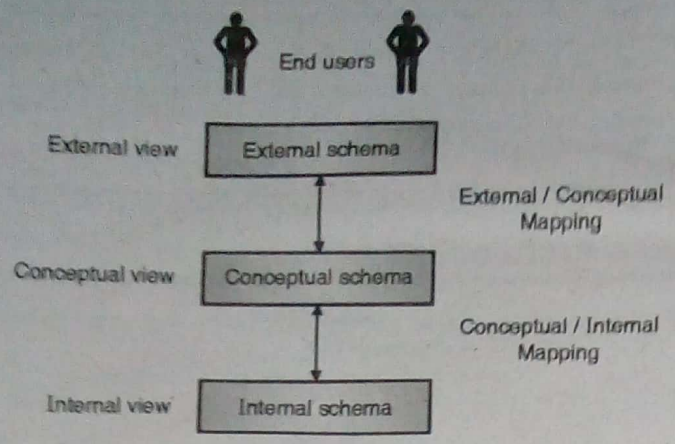


Fig. 2.2 : Three level schema architecture

(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.

Q. 2 Define data Independence and explain types of data Independence.

Ans. : Data Independence

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

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.

Q. 3 Discuss the role of Database Administrator.

Ans. : Role of Database Administrator

1. The DBA needs to performs many roles to keep the database up and running.
2. System Administrator / Designer
3. The database administrator need to manage DBMS software and server.
4. He is also responsible for deciding on the storage and access methods.
6. The DBA performs all data field updates or adding new fields into database.
7. Database Developer / Programmer
8. The DBA writes the programs to design database and to design the means of reorganizing databases periodically.
9. The DBA also determines and implement database searching strategies.
10. System Analyst
11. The DBA needs to analyse the system performance and fine tune the DBMS activities.
12. DBA needs to take care of system crashes by planning proper recovery procedures.
13. He will also specify techniques for monitoring database performance.

Q. 4 Write a short note on : Responsibilities of Database Administrator.

Ans. : Responsibilities of Database Administrator

The various responsibilities of DBA are as follows,

1. Designing overall Database schema
2. The DBA is responsible for designing overall database schema (tables and fields). Also responsible for deciding on the data storage and access methods.
3. Selecting and installing database software and hardware.
4. The DBA selects the suitable DBMS software like Oracle, SQL Server or MySQL.
5. Designing Authorization/Access Control
6. The DBA will decide the user access levels and security checks for access and data manipulations.
7. Designing Recovery Procedures
8. In order to take care of system crashes DBA needs to design the system recovery procedures and also specifying techniques for monitoring database performance.

9. Operations Management
10. The operations management of database administration deals with data problems arising on a day-to-day basis. Specifically, the responsibilities include
    (i) Investigation of errors found in the data.
    (ii) Supervision of restart and recovery procedures in the event of a failure.
    (iii) Supervision of reorganization of databases.
    (iv) Initiation and control of all periodic dumps of data.

Q. 5 List the important skills required to a Database Administrator (DBA).

Ans. : Skills of Database Administrator (DBA)

The various programming and soft skills are required to DBA are as follows,

1. Good communication skills
2. Excellent knowledge of databases architecture and design and RDBMS
3. 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.

Q. 6 Write a short notes on : Query processor.

Ans. : Query processor

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

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.

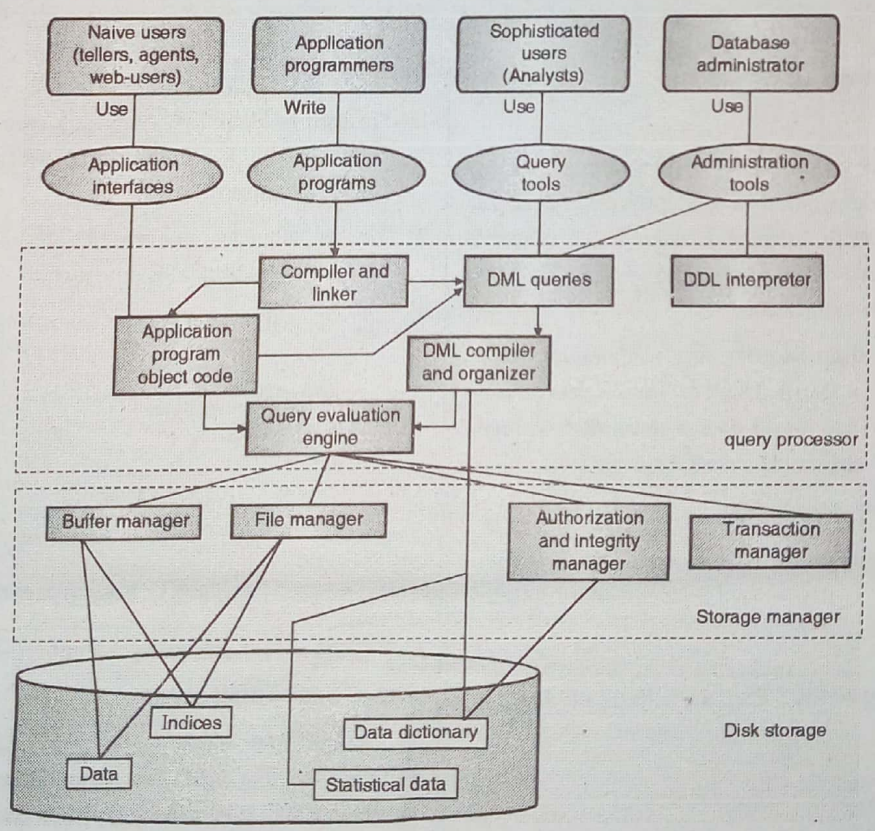


Fig. 2.3 : Components of DBMS

Q. 7 Write a short note on : Storage management.

Ans. :

Storage management

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 :

1. **Authorization and integrity manager :** Checks for integrity constraints and authority of users to access data.
2. **Transaction manager**, which ensures that the database remains in a consistent (correct) state although there is system failures.
3. **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
4. **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,

1. **Data files :** Stored in the database itself.
2. **Data dictionary :** Stores metadata about the structure of the database.
3. **Indices :** Provide fast access to data items.

Q. 8 Write a short note on : Transaction management.

Ans. : Transaction management

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.

Chapter 3 : Entity Relationship Data Model

Q. 1 What do you mean by ER model.

Ans. : ER Model

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,

1. Entity
2. Attributes
3. Relationships

Q. 2 What is strong entity? Explain with example.

Ans. :

Strong Entity

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

Example

1. In case of Employee entity any specific employee can be identified by his Employee_id which is primary key of employee entity.
2. In case of student in class each student identified by unique roll number which is his primary key.
3. Strong entity type is represented by single rectangle.

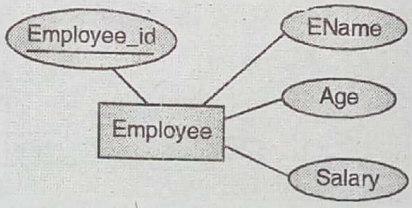


Fig. 3.1 : Employee entity

Q. 3 What is weak entity ? Explain with example.

Ans. : Weak Entity

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.

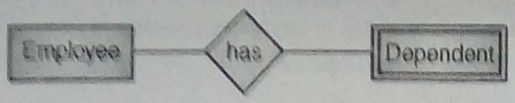


Fig. 3.2 : Weak entity "dependent"

Q. 4 Explain different types of attributes in ER Model.

Ans. : Types of Attributes

(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.

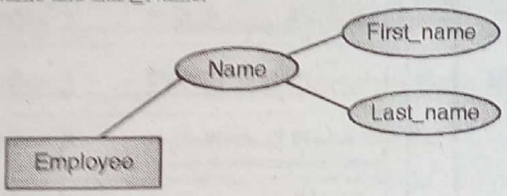


Fig. 3.3 : Composite attributes

(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.

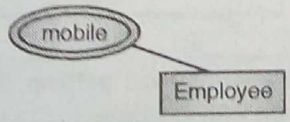


Fig. 3.4: Multi valued attributes

(c) Derived Attributes

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.

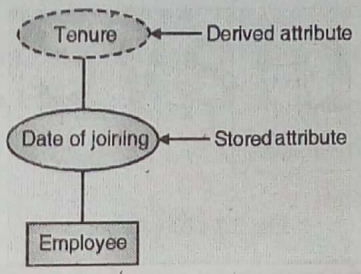


Fig. 3.5 : Derived attributes

(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.

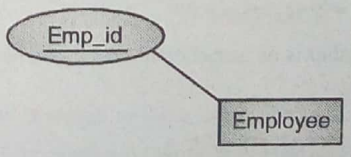


Fig. 3.6 : Key attributes

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)

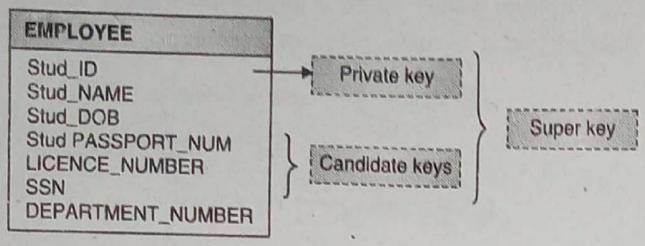


Fig. 3.7 : Key Concept

Q. 5 Explain different types of keys.

Ans. : Types of Keys

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

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
2. **Composite Key :** Any key with more than one attributes that uniquely identifies a single tuple in entity.
3. **Candidate Key :** A super key with minimum number of attributes is a candidate's key. No subset of candidate key van be key.
4. **Primary Key :** A selected key of strong entity which uniquely identify tuple in entity is a primary key of that entity.
5. **Alternate Key :** A Candidate key which is not selected as primary 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

Q. 6 What is relationship set?

Ans. :

Relationship Set

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.

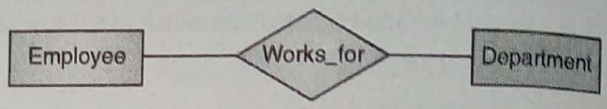


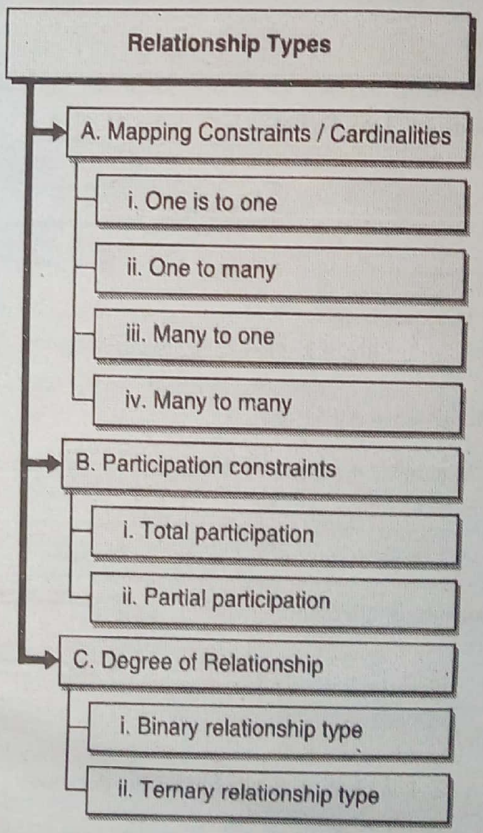
Fig. 3.8 : ER Diagram for Works_for

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.

Q. 7 List various constraints of relationship.

Ans. :



Q. 8 Explain the terms total participation and Partial participation with example.

Ans. :

(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.

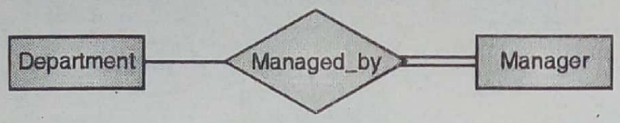


Fig. 3.9 : Total participation

(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.

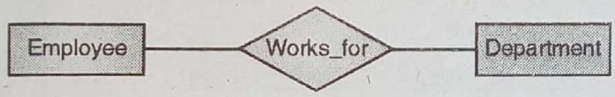


Fig. 3.10 : Partial participation

Q. 9 Explain Specialization with the help of an example.

Ans. :

Specialization

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.

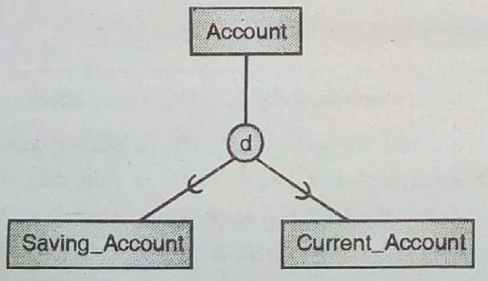


Fig. 3.11 : Specialization

Q. 10 Explain Generalization with the help of an example.

Ans. :

Generalization

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

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.

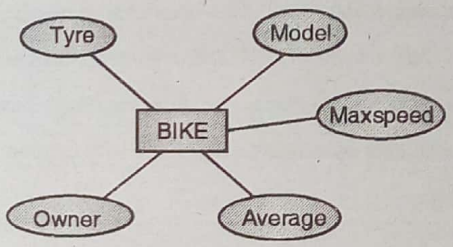


Fig. 3.12 : BIKE entity

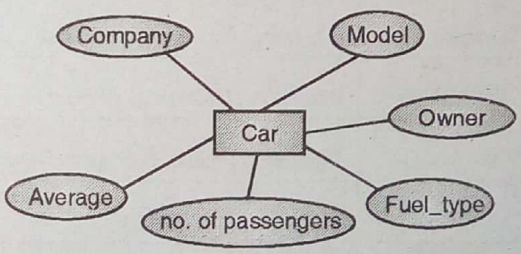


Fig. 3.13 : Car entity

Q. 11 Explain aggregation with the help of an example.

Ans. :

Aggregation

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.

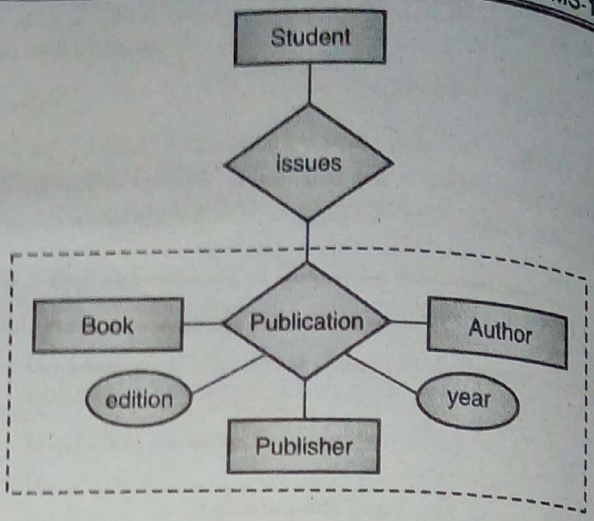


Fig. 3.14 : Aggregation

Chapter 4 : Relational Data Model

Q. 1 Write a note on Relational Database Schema.

Ans. :

Relational Database Schema

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. 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. 4.1 : Sample Relational Schema

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

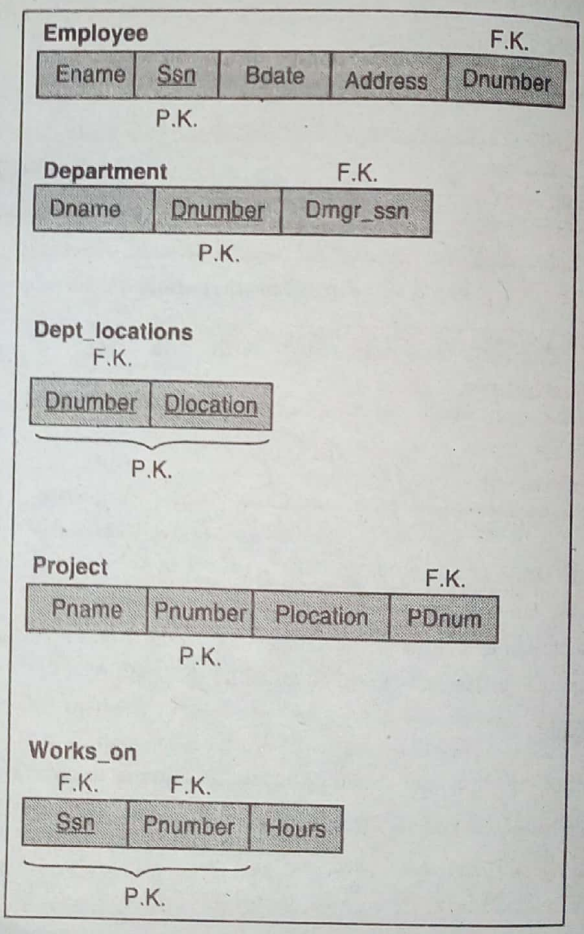


Fig. 4.2. : Sample Relational Database Schema

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,

$R_1, R_2, ... R_n$ = Relation or Tables

A, B, C = Attributes or Columns

Q. 2 Explain types of Domain constraints.

Ans. : 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'
```

Q. 3 Explain types of Entity constraints.

Ans. :

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
```

Chapter 5 : Relational Algebra

Q. 1 Describe the following Relational algebra operation : Project.

Ans. : Project Operator

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.

Syntax

$$\pi_{<column\_list>}(\text{Input\_Table\_Name})$$

Example

Query : Find salary and age of all Employees.

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

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

Age	Salary
24	50000
24	15000
23	41000

Q. 2 Explain Rename Relational algebra operators with suitable examples.

Ans. :

Rename Operation

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).

Syntax

$$\rho_{<New\_Name>}(\text{Input\_Table\_Name})$$

Example

Query : Find salary and age of all Employees.

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

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

Q. 3 Explain Set Union Relational algebra operators with suitable examples.

Ans. : Union Operator

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.

Syntax

(Query Expression 1) $\cup$ (Query Expression 2)

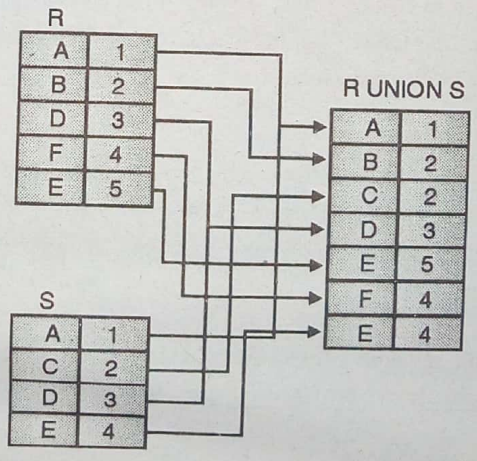


Fig. 5.1 : SET Operation

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) ∪ (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

Q. 4 Explain Set Intersection operation with suitable examples.

Ans. :

Intersect Operator

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.

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

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

Q. 5 Express Join in terms of basic relational algebra operations.

Ans. :

Joins

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

Syntax

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

There are various types of joins possible in relational algebra.

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

1. The two table are joined using Cross join
2. DBMS will look for a common column with same name and data type
3. 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 values 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 $\bowtie_{\text{employee.did= Department.did}}$ 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 $\bowtie=_{\text{employee.did= Department.did}}$ 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.

Q. 6 Explain Division Relational algebra operators with suitable examples.

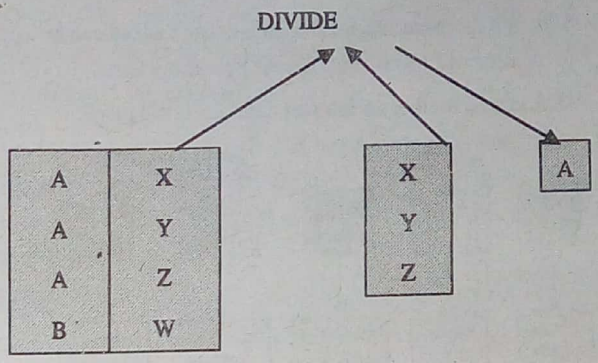
Ans. : Division Relational Operator

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.

Syntax

(Query Expression 1) (Query Expression 2)



DIVIDE Operator

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

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

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

Chapter 6 : Structured Query Language

Q. 1 Explain role of SQL with example.

Ans. : Role of SQL

1) SQL is an interactive query language which can be used to retrieve data from database.

2) SQL is a database programming language which can be used along with programming language to access data from database.

3) SQL is a database administration language which can be used to monitor and control data access by various users.

4) SQL can be used as an Internet data access language.

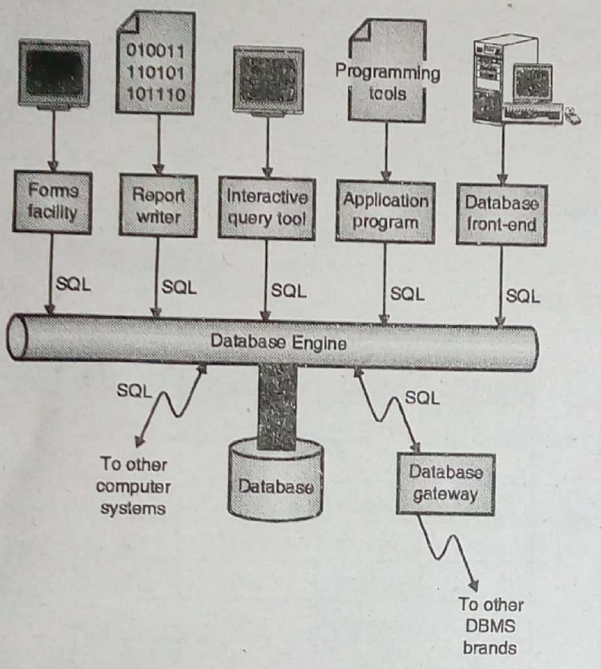


Fig. 6.1: Role of SQL in DBMS

Q. 2 Explain various SQL data types with example.

Ans. : SQL Data Types

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.

Q. 3 Explain uses of DML commands with syntax.

Ans. : Data Manipulation language

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.

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|
+----+---------+
```

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  |
+----+---------+
```

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   |
+----+---------+
```

Q. 4 Explain all types of privileges.

Ans. : Types of Privileges

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.

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

Object privileges	Authorized to
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.

Q. 5 Write a short note on Revoking of privileges.

Ans. : Revoking of Privileges

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

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.

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;
```

Chapter 7 : SQL Security

Q. 1 Describe view.

Ans. : View

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.

Base table

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

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)

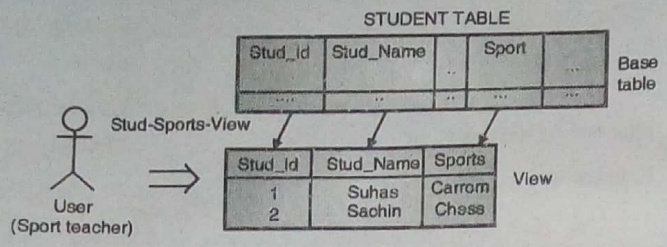


Fig. 7.1 : Overview of view

Q. 2 What are the types of views ?

Ans. :

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.

Q. 3 Explain syntax for creating views.

Ans. : Creating View

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.

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.

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.
```

Q. 4 How to drop view ? Give Syntax.

Ans. : Drop View

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.

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.

Example

Remove a view created in above step.

```
SQL>     DROP VIEW JobBelow3K;

View Dropped;
```

Q. 5 What are advantages of views ?

Ans. : 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.

Q. 6 What are disadvantages of views ?

Ans. : 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.

Chapter 8 : Trigger

Q. 1 Describe Trigger, components of it and types of trigger.

Ans. : Trigger

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.

Components of Trigger (E-C-A model)

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

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 : Trigger parameters

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.

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;
```

Q. 2 Explain solution for mutating table error

Ans. : Mutating Table Error

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.

Chapter 9 : Relational Database Design

Q. 1 Describe various design guidelines for relational schema.

Ans. :

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.

In above 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.

In above 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.

In above 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,

1. Not applicable data
2. Invalid data
3. 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.

Q. 2 What is Normalization ?

Ans. : Normalization

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.

Normalization is a process of designing a consistent database by minimizing redundancy and ensuring data integrity through decomposition which is lossless.

Q. 3 Explain concept of functional dependency.

Ans. : Functional Dependencies

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.

Example

Consider an employee table with columns as shown in Table 9.1

Table 9.1 : 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.

Q. 4 What are types of functional dependencies?

Ans. : Types of Functional 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 9.2

Table 9.2 : 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, 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 9.3

Table 9.3 : 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

1. Consider an employee table schema,
2. Employee_Id → Department_Id
3. Department_Id → Dname

Table 9.4 : 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) $X \rightarrow Y$ and Y is a subset of X, then it is called as a trivial FD. Functional dependency $X \rightarrow Y$ 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 $X \twoheadrightarrow Y$ 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 9.5 : 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 $\twoheadrightarrow$ Car

$\alpha \twoheadrightarrow \beta$ says relationship between α and β independent of relationship between α and R-β. That means **Employee_Id $\twoheadrightarrow$ Car** relationship is independent of Employee_Id and EName relation.

i.e. **Employee_Id $\twoheadrightarrow$ Car** is independent of Employee_Id $\twoheadrightarrow$ Ename

Q. 5 What are goals of decomposition?

Ans. : Goals of Decomposition

(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

1. Employee (Employee_Id, Ename, Salary, Department_Id, Dname)
2. Can be decomposed using lossless decomposition as,
3. Employee_desc (Employee_Id, Ename, Salary, Department_Id)
4. Department_desc (Department_Id, Dname)
5. Alternatively the lossy decomposition would be as joining these tables is not possible so not possible to get back original data.
6. Employee_desc (Employee_Id, Ename, Salary)
7. Department_desc (Department_Id, Dname)
8. 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.

Q. 6 Write a note on : Super Key.

Ans. : 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.

Q. 7 Write a note on : Candidate Key.

Ans. : Candidate Keys

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 are 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.

Q. 8 Write a note on : Secondary Key.

Ans. :

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,

$AB \rightarrow C, BC \rightarrow DE, E \rightarrow AF$

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.

Q. 9 Explain 1NF.

Ans. : 1NF

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).

Definition

A relation is in 1NF, if every row contains exactly one value for each attribute. 1NF states that all attributes in relation must have atomic (indivisible) values and all attribute in a tuple must have a single value from the domain of that attribute.

In short rules for data in 1NF is,

A column in a table should contain only indivisible data.

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 9.6 : (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 9.7 : 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

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.

Q. 10 Explain 2NF.

Ans. : 2NF

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.

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.

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 9.8 : 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 9.9 : 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 9.10 : 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

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.

Q. 11 Explain 3NF.

Ans. :

3NF

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.

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.

Example

1. Consider an employee table with columns as shown in diagram,
2. The relational schema **not in 2 NF** is represented as,
3. Consider an Employee table with following FDs,
4. Employee_Id → Ename, Salary, Department_Id
5. Department_Id → Dname
6. The state of Employee relational is,

Table 9.11 : 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 : 3NF Employee Table

Employee_Id	Ename	Salary	Department_id
10	Mahesh	50000	C1
12	Suresh	25000	E2

Employee_Id	Ename	Salary	Department_id
15	Ganesh	26000	C1
18	Mahesh	50000	E2

Department (Department_Id, Dname)

Employee_Id → Dname

Table 9.13 : 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.

1. All attributes are full functionally dependent on primary key.
2. Hence, Relation R is **in 2NF**.
3. But, non-prime attributes B, D, E and F are transitively depends on key.
4. So, Relation R is **not in 3NF**.

(ii) The 3NF Relation Schema is,

1. R1 (A, B, C) with FDs A → BC
2. R2 (B, D) with FDs B → D
3. R3 (D, E, F) with FDs D → EF

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.

Q. 12 Describe BCNF in detail.

Ans. : BCNF

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.

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.

Example

1. Consider an employee table in which employee can work in more than one department,
2. The relational schema **not in 2 NF** is represented as,
3. Consider an Employee table with following FDs,
4. Employee_Id → Ename, Salary, Department_Id
5. Department_Id → Dname
6. The state of Employee relational is,

Table 9.14 : 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 9.15 : 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 9.16 : 3NF Department Table

Department_Id	Dname	Dtype
C1	IT	Technical

Department_Id	Dname	Dtype
E2	HR	Skill
E1	TS	Technical

Emp_Dept (Employee_Id, Department_Id)

Table 9.17 : 3NF Emp Dept Table

Employee_Id	Department_Id
10	C1
12	E2

Employee_Id	Department_Id
12	C1
10	E2
13	E1

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.

Chapter 10 : Transaction

Q. 1 What is transaction? Explain concept of transaction using example.

Ans. : Transaction

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

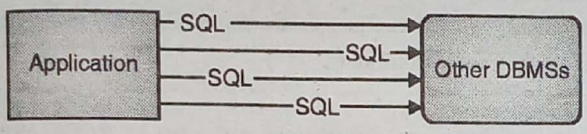


Fig. 10.1 : Executing single operation in DBMS

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.

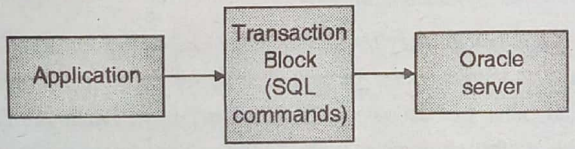


Fig. 10.2 : Executing transaction in DBMS

Types of operations that can be done inside transaction,

a. Read operation

Read operation transfers data item from (e.g. table) the database memory to a local buffer 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 that 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.

Q. 2 Explain Transaction structure and boundaries

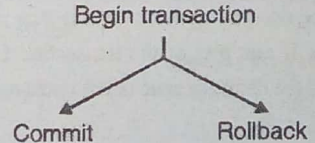
Ans. :

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.



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.

Q. 3 Explain ACID properties of transaction.

Ans. :

ACID 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 (Ti) all other transactions has finished before transactions (Ti) started, or other transactions are started execution after transactions (Ti) 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.

Q. 4 Explain transaction state diagram.

Ans. :

Transaction states

A transaction must be in one of the following states :

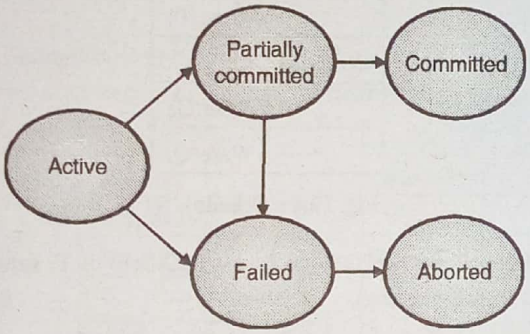


Fig. 10.3 : State diagram of a transaction

(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 :

1. **Restart the transaction :** A restarted transaction is considered to be a new transaction which may recover from possible failure.
2. **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.

Q. 5 What is transaction schedule?

Ans. : Transaction Schedule

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.

Q. 6 Explain advantages of concurrency

Ans. :

Advantages of concurrency

(1) 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.

(2) 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.

(3) 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.

Q. 7 Explain conflict serializability, Describe using example.

Ans. :

Conflict serializability

The database system must control concurrent execution of transactions which ensure that the database state remains in consistent state.

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_j	
		Read(D)	**Write(D)**
Transaction T_i	**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)

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.

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. 10.4 : Schedule S

Instruction WRITE(P) of T_1 and READ(P) of T_2 cannot be swapped as they conflict.

	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. 10.5

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.

Can be swapped

T_1	T_2
Read(P)	
Write(P)	
	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. 10.6

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

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. 10.8

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 $< T_1, T_2>$.

Q. 8 Explain view serializability, describe using example.

Ans. : View Serializability

View equivalence is less strict than conflict equivalence, but it is like conflict equivalence based on only the read and write operations of transactions.

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 $<T_1, T_2, T_3>$.

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. 10.9

Chapter 11 : Concurrency Control

Q. 1 Explain the concept of concurrency control.

Ans. :

Concurrency Control

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.

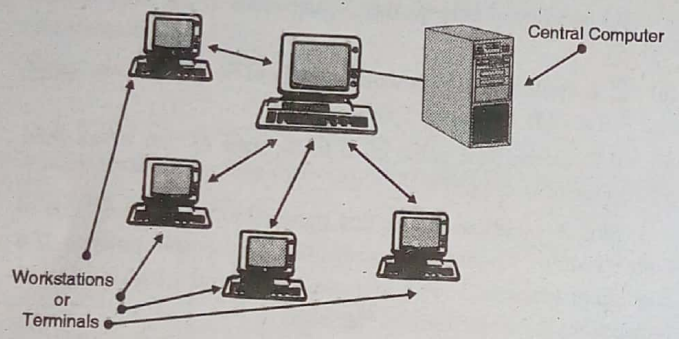


Fig. 11.1 : Concurrent database access

Concurrency control should provide a mechanism for avoiding and managing conflict between various database users operating same data item at same time.

Q. 2 Explain various problems of concurrent executions.

Ans. :

Problems of Concurrent Executions

(1) Lost update problem

Update made by one transaction is overwritten by other transaction or other user. This may loss updates of one transaction.

(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
t_6	A=A – 10	Rollback	100
t_7	write (A)		190
t_8	commit		190

(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

(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**.

Q. 3 Explain types of lock.

Ans. :

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;
```

Q. 4 Explain versions of two phase locking protocol.

Ans. :

Versions of Two Phase Locking

(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.

Q. 5 Explain Thomas write rule with algorithm.

Ans. :

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	....	....	....

TS	T_i	T_x	Operations
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	....	....	....

TS	T_i	T_x	Operations
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).

Chapter 12 : Recovery System

Q. 1 Explain concept of database recovery and list the techniques.

Ans. : Database Recovery

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,

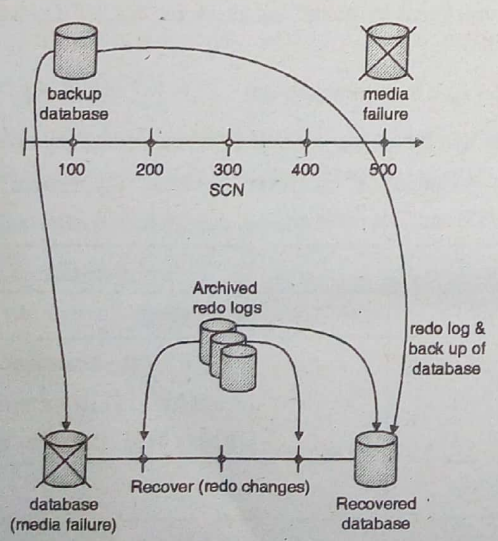


Fig. 12.1

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.

Database recovery techniques

(a) Log based recovery

(b) Shadow paging recovery

(c) Checkpoints

Q. 2 Explain system failure classification.

Ans. :

System Failure Classification

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.

Q. 3 Explain types of log records

Ans. : 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.

Q. 4 Explain concept of shadow paging.

Ans. :

Shadow Paging

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**.

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.

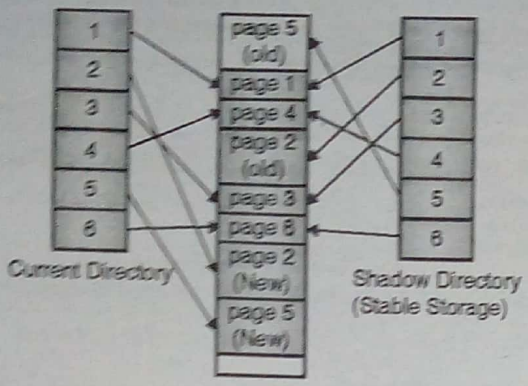


Fig. 12.2 : Pagetable-shadow paging

(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.

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.

Advantages

1. Shadow page method does not require any Undo or Redo algorithm for recovery purpose.
2. Recovery using this method will be faster.
3. No overhead for writing log records.

Disadvantages

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.

Q. 5 Explain ARIES algorithm with different steps. Enlist all advantages of ARIES algorithm

Ans. :

ARIES Algorithm

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

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.

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.

Example

a. Consider the crash recovery example illustrated in Fig 12.3

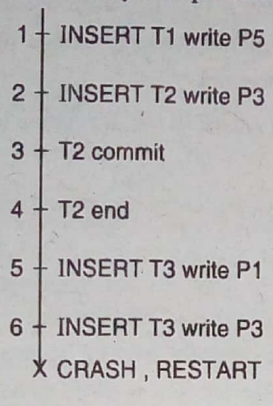


Fig. 12.3: Example of crash recovery using ARISE

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.

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.

Q. 6 What is database deadlock? Explain various deadlock handling techniques.

Ans. : Database Deadlock

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)	
	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 T_1 is waiting for transaction T_2 to Unlock data item Y and transaction T_2 is waiting for transaction T_1 to Unlock data item X.

So system is in a deadlock state as there are set of transactions T_1 and T_2 such that, both are waiting for each other transaction to complete. This state is called as **Deadlock state**.

There are **two principle methods** to handle deadlock in system,

(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. Both of above methods may result in transaction rollback. 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.

Q. 7 Explain approaches for Deadlock Prevention.

Ans. :

Deadlock Prevention Approaches

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.

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_i and T_j can't affect read of T_i so serializability is maintained.

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 :

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.

Q. 8 Explain Deadlock detection in detail.

Ans. :

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.

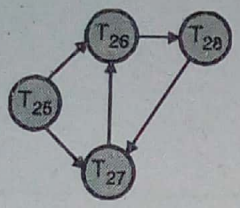


Fig. 12.4 : Wait-for graph with a cycle

(v) To detect deadlock, system must maintain wait-for-graph and periodically invoke an algorithm that searches cycle in the graph.

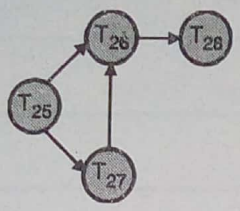


Fig. 12.5 : Wait-for graph with no cycle

(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.

Q. 9 Explain Deadlock recovery methods

Ans. :

Deadlock Recovery Methods

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,

(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.

1. For how much time transaction has computed and how much time it needs to do.
2. How many data items it has locked.
3. How many data items it may need ahead.
4. 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.