MSc-IT

GAUHATI UNIVERSITY
Institute of Distance and Open Learning

Semester- I

MSc-IT
Paper: INF 1056
(under CBCS)

Advanced Database Management System

www.idolgu.in

ADVANCED DATABASE MANAGEMENT SYSTEM

GAUHATI UNIVERSITY
Institute of Distance and Open Learning

First Semester

(under CBCS)

M.Sc.IT

Paper: M.Sc.-IT-19-I-1056

ADVANCED DATABASE MANAGEMENT SYSTEM

Contents:

Contributors:

Dr. Sisir Kumar Rajbongshi (Block I : Unit- 1)
Asstt. Prof., Dept. of Computer Science
PDUAM, Goalpara

Mr. Surajit Medhi (Block I: Units- 2 & 5)
Asstt. Prof., Dept. of B.Sc.(IT)
B. Borooah College, Guwahati, Assam

Dr. Ridip Dev Choudhury (Block I : Units- 3 & 4)
Associate Prof., HCB School of Science & Technology
KKHSOU, Assam

Dr. Sonia Sarmah (Block I : Unit- 6)
Asstt. Prof., Dept. of Computer Applications
Assam Don Bosco University, Assam

Mrs. Epsita Medhi (Block II: Unit- 1)
Research Asstt., Dept. of Information Technology
Gauhati University, Assam

Dr. Pranab Das (Block II: Unit- 2)
Asstt. Prof. (Sr.), Dept. of Computer Applications
Assam Don Bosco University

Dr. Manash Protim Bhuyan (Block II: Unit- 3)
Asstt. Prof., Dept. of Computer Science
& Engineering, Golaghat Engineering
College, Golaghat, Assam

Mr. Deepjyoti Kalita (Block II: Unit- 4)
Asstt. Prof., Dept. of Computer Science
Mangaldai College, Darrang, Assam

Dr. Dipen Nath (Block III: Unit- 1)
Asstt. Prof., Dept. of Computer Science
Mangaldai College, Darrang, Assam

Dr. Utpal Barman (Block III: Units- 2, 3 & 4)
Asstt. Prof., Dept. of Computer Science
& Engineering
GIMT, Guwahati, Assam

Course Coordination:

Prof. Dandadhar Sarma Director, IDOL, Gauhati University
Prof. Anjana Kakoti Mahanta Prof., Dept. Computer Science, G.U.

Cover Page Designing:

Bhaskar Jyoti Goswami IDOL, Gauhati University

ISBN:

August, 2021

Published on behalf of Institute of Distance and Open Learning, Gauhati University by the Director, and printed at Gauhati University Press, Guwahati-781014.

UNIT-1
RELATIONAL DATA MODEL ANDRELATIONAL DATABASE CONSTRAINTS

Structure

1.0 INTRODUCTION

In this unit, you will study about the relational model of database. You will be taught about the various components, characteristics and limitations of this model. The unit will also familiarize you with key terms related to relational model like- domain, attribute, tuple etc. You will also learn the various types of keys such as primary, alternate, foreign, candidate, logical and super key with example. The unit will also give emphasis on the various relational constraints. The different types of constrains- domain constraint, key constraint, entity integrity constraint and referential integrity constraint will be discussed. Finally, you will learn about the Entity-Relationship diagram.

1.1 UNIT OBJECTIVE

After going through this unit, you will be able to:

- Understand in detail the relational model, its advantages, limitations and applications
- Explain the key terms of relational model
- Structure of the relational model
- Learn various characteristic of relations
- Understand the different keys in relational model
- Learn about the relational constraints
- Operations in Relational Model with Constraint Violations
- Analyze the E-R diagram

1.2 RELATIONAL MODEL

Storing and managing information is one of the most important tasks for computers. The way in which information is organized can have a profound effect on how easy it is to access and manage. Perhaps the simplest but most versatile way to organize information is to store it in the forms of tables. Table is the backbone of the relational model.

The relational model is centered on this idea: the organization of data into collections of two-dimensional tables called "relations." The data and relationships are represented by collection of inter-relatedtables. Each table is a group of column and rows, where column represents attribute of an entityand rows represents records. The table name and column names are helpful to interpret themeaning of values in each row. In the formal relational model terminology, a row is called a tuple, acolumn header is called an attribute, and the table is called a relation. The data type describing thetypes of values thatcan appear ineach column isrepresented by adomain of possiblevalues.

Originally, the relational model of database was introduced in 1970, by English Computer Scientist **Edgar F. Codd**. The relational data model was developed for databases — that is, information stored over a long period of time in a computer system — and for database management systems, the software that allows people to store, access, and modify this information. Databases still provide us with important motivation for understanding the relational data model. They are found today not only in their original, large-scale applications such as train booking systems or hospital management systems, but in desktop computers handling individual activities such as maintaining expense records, homework grades, and many other uses.

1.2.1 Advantages of Relational Model

The relational model is the most dominant database model. It has lots of advantages:

1. **Simple Model**

Compared to other types of models, a relational database model is much simpler. It is free from query processing and complex structuring. As a result, it does not require any complex queries. A simple SQL query is sufficient enough for handling.

2. **Data Accuracy**

In the relational database system, there can be multiple tables related to one another with the use of primary key and foreign key concepts. Hence, there is no repletion of data. There is no chance for duplication of data. Hence the accuracy of data in the relational database is more than any other database system.

3. **Easy Access to Data**

In the Relational Database System, there is no pattern or pathway for accessing the data, as to another type of databases can be accessed only by navigating through a tree or a hierarchical model. Anyone who accesses the data can query any table in the relational database. Using join queries and conditional statements one can combine all or any number of related tables in order to fetch the required data. Resulting data can be modified based on the values from any column, on any number of columns, which permits the user to effortlessly recover the relevant data as the result. It allows one to pick on the desired columns to be incorporated in the

outcome so that only appropriate data can be displayed.

4. **Data Integrity**

Data integrity is a crucial characteristic of the Relational Database system. It ensures that all the data in the database confines within suitable arrangements and the data necessary for creating the relationships are present. This relational reliability amongst the tables in the database helps in avoiding the records from being imperfect, isolated or unrelated. Data integrity aids in making sure of the relational database's other significant characteristics like Ease of use, precision, and stability of the data.

5. **Flexibility**

A Relational Database system by itself possesses qualities for leveling up, expanding for bigger lengths, as it is endowed with a bendable structure to accommodate the constantly shifting requirements. This facilitates the increasing incoming amount of data, as well as the update and deletes wherever required. This model consents to the changes made to a database configuration as well, which can be applied without difficulty devoid of crashing the data or the other parts of the database.

A Data Analyst can insert, update or delete tables, columns or individual data in the given database system promptly and easily, in order to meet the business needs. There is supposedly no boundary on the number of rows, columns or tables a relational database can hold. In any practical application, development and transformation are restricted by the Relational Database Management System and the hardware contained by the servers. So these changes can create an alteration in other peripheral functional devices connected to the particular relational database system.

6. **Normalization**

The normalization process provides a set of regulations, characteristics, and purposes for the database structure and evaluation of a relational database model.Normalization aims at illustrating multiple levels of breaking down the data. Any level of normalization is expected to be accomplished on the same level, that is, before moving ahead to the next levels. A relational database model is usually confirmed to be normalized, only when it satisfies the necessary conditions of the third normalization form. Normalization offers an impression of reassurance on the database plan, to be extra strong and reliable.

7. **High Security**

As the data is divided amongst the tables of the relational database system, it is possible to make a few tables to be tagged as confidential and others not. This segregation is easily implemented with a relational database management system, unlike other databases. When a data analyst tries to login with a username and password, the database can set boundaries for their level of access, by providing admission only to the tables that they are allowed to work on, depending on their access level.

8. **Feasible for Future Modifications**

As the relational database system holds records in separate tables based on their categories, it is straightforward to insert, delete or update records that are subjected to the latest requirements. This feature of the relational database model tolerates the newest requirements that are presented by the business. Any number of new or existing tables or columns of data can be inserted or modified depending on the conditions

provided, by keeping up with the basic qualities of the relational database management system.

1.2.2 Limitations of Relational Database

The relational model suffers from certain limitations. The relational model has been developed to meet the requirements of business information processing. While applying the relational model to the application areas, such as Computer-Aided Design (CAD), simulation and image processing, many shortcomings have been noticed in this model also. The various shortcomings of this model may be discussed as follows:

1. **Cost**

The underlying cost involved in a relational database is quite expensive. For setting up a relational database, there must be separate software which needs to be purchased. And a professional technician should be hired to maintain the system. All these can be costly, especially for businesses with small budget.

2. **Performance**

Always the performance of the relational database depends on the number of tables. If there are more number of tables, the response given to the queries will be slower. Additionally, more data presence not only slows down the machine, it eventually makes it complex to find information. Thus, a relational database is known to be a slower database.

3. **Physical Storage**

A relational database also requires tremendous amount of physical memory since it is with rows and columns. Each of the operation depends on separate physical storage. Only through proper optimization, the targeted applications can be made to have maximum physical memory.

4. **Complexity**

Although a relational database is free from complex structuring, occasionally it may become complex too. When the amount of data in a relational database increases, it eventually makes the system more complicated.

5. **Information Loss**

Large organizations tend to use more number of database systems with more tables. These information can be used to be transferred from one system to another. This could pose a risk of data loss.

6. **Structure Limitations**

The fields that are present on a relational database has limitations. Limitations are in that sense that it cannot accommodate more information. Despite if more information are provided, it may lead to data loss. Therefore, it is necessary to describe the exact amount of data volume which the field will be given.

1.3 COMPONENTS AND RELATIONAL TERMINOLOGIES

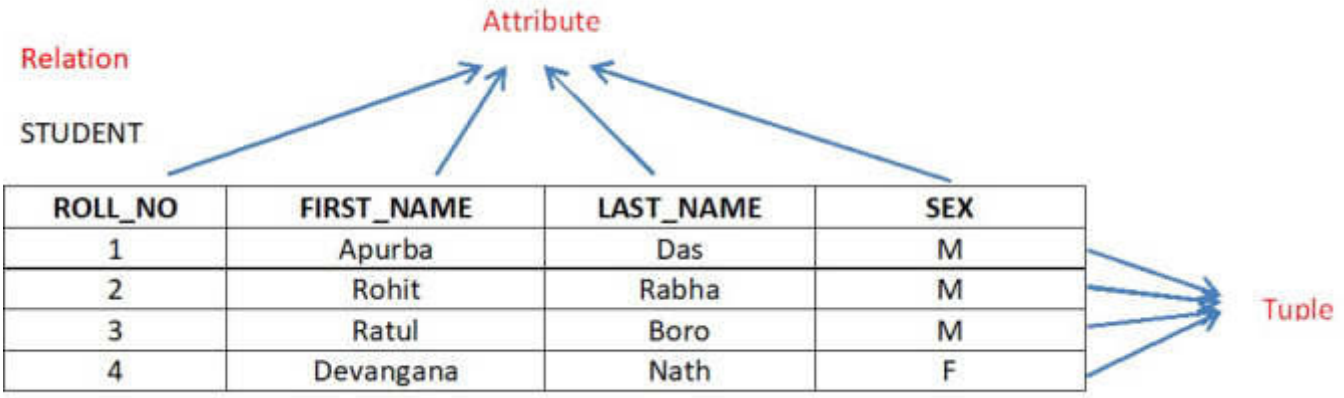
The main principle of the relational model is the information principle-all information is represented by data values in relations. The three components- structural, manipulative and integrity- make up this model. These

components are defined as follows:

- The structural component is concerned with how data is represented.
- The manipulative component is concerned with how data is operated upon.
- The integrity component is concerned with determining which states are valid for a database.

The various relational terminologies are described as follows-

Domain: A domain is a set of values permitted for an attribute in a table. Domain is atomic. Forexample, ROLL_NO can only be a positive integer. A data type or format is also specified for eachdomain.Itispossibleforseveralattributestohavethesamedomain.



ROLL_NO	FIRST_NAME	LAST_NAME	SEX
1	Apurba	Das	M
2	Rohit	Rabha	M
3	Ratul	Boro	M
4	Devangana	Nath	F

Fig. 1.1 *Relational Model Concepts*

Attribute: Attributes are the characteristics of a relation. Each column in a table is the attribute. Attributes are the properties which define a relation. e.g.,ROLL_NO, FIRST_NAME etc of the relation Student. Each attribute in a relational model must have domain information. Domain information contains the following:

- ***Data Type***: Databases provide support for different types of data and their variants. For example integer, float etc.
- ***Length***: Length means number of characters or digits that an attribute value has. For example, when we assign a PIN code, it has 6 digits.
- ***Date format***: A date contains day, month and year. These three must be given in combination. Such as DD/MM/YYYY or MM/DD/YYYY or YYYY/MM/DD etc.
- ***Range***: A range is specified by lower and upper bounds of data values that an attribute may have.
- ***Constraints***: These are particular type of conditions that put restrictions on values that are allowed.
- ***NULL support***: There is a support for NULL values in relational model. Some particular attribute may remain blank. For example, in a relation the column "PAN_Number" may be blank as a person may not have a PAN number.
- ***Default Value***: If nothing is entered, database assigns a default value. Relational model supports the facility that the default value may be set for every attribute.

Attributes can be of many types:

- ***Composite vs Simple attribute***: Composite attributes can be subdivided into smaller attributes. For example the "Name" of a student can be divided into "First_Name", "Middle_Name" and

"Last_Name". On the otherhand simple attributes are the attributes that can't be subdivided into smaller attributes. For example the "Roll_No" of a STUDENT.

- ***Single Valed vs Multivalued***: Single valued attributes are the attributes that have a single value for a particular entity. For example the "Last_Name" of an EMPLOYEE. But the multivaled attributes can have more than one value for a particular entity. For example "Mobile _Number" of an EMPLOYEE.
- ***Derived vs Stored attributes***: Derived attributes are the attributes whose vales can be derived from the value of some other attributes. For example the age of a student can be derived from date of birth of the student. The "Date_of_Birth" is called the stored attribute from which you can derive some other attribute.
- ***NULL attribute***: A certain entity may not possess a value for an attribute. This will mean "not applicable" or that the value is unknown" or "non-existent". For example there may be chance when a student has no phone no. In that case the "Phone_No" attribute is called NULL attribute.

Tuple – Itis nothing buta single rowof a table,which contains asingle record.

Relations- are in the table format. It is stored along with its entities. A table has two propertiesrows and columns. Rows represent records andcolumns represent attributes.

Relation Schema- A relational schema is the design for the table. It includes none of the actualdata, but is like a blueprint or design for the table, so describes what columns are on the table andthe data types. It may show basic table constraints (e.g. if a column can be null) but not how itrelates to other tables.

A relation schema R, denoted by R (A1,A2, ..., An), is made up of a relation name R and alist of attributes, A1,A2, ...,An. Each attribute Ai is the name of a role played by some domain D intherelationschemaR.Dis called the domain of Ai and is denoted by dom(Ai). The relationschema R(A1,A2, ...,An), also denoted by r(R),is a set of n -tuplesr= {t1,t2, ...,tm}.

Degree- (or arity) of a relation is the number of attributes n of its relation schema. A relation ofdegree four, which stores information about college students, would contain four attributesdescribing each student as follows:

STUDENT(Roll_No, First_Name, Last_name, Sex)

Cardinality: Total numberof rowspresent inthe Table.

RelationInstance–Relationinstanceisafiniteset oftuplesata giventime. Relation instances do not have duplicate tuples.

Null Value: Afield with aNULL value isa field withno value. Primarykey can't bea null value.

Characteristics ofRelations

- **Ordering of Tuples in a Relation:** A relation is defined as a set of tuples.The tuples in arelation do not have any particular order. In other words, a relation is not sensitive to theordering of tuples.

However, in a file, there always is an order among the records. Tupleorderingis notpartofa relation definition because a relation attempts to represent facts at a logical or abstractlevel. Many tuple orders can be specified on the same relation. For example, tuples in theSTUDENTrelationcouldbeorderedbyvaluesofName,Roll,Age etc. The definition of a relation does not specify any order: There is no preference forone ordering over another.

- **Ordering of attributesin a Relation:** The ordering of attributes is not important, becausethe attribute name appears with its value. There is no reason to prefer having one attributevalue appear before another in a tuple. When a relation is implemented as a file, theattributes and the values within tuples are ordered.

- **Values in a tuple**: All values are consideredatomic. A special null value is used torepresentvaluesthatare unknown or inapplicable to certain tuples. In general, NULLvalues, means value unknown or value exists but is notavailable.

Table 1.1 *STUDENT*

Roll	Name	Phone	Age
1	Nipun	1234567890	26
2	Nava	1234567891	28
3	Mohit	1234567892	19

- **Interpretation(Meaning)ofaRelation:**Therelationschemacanbeinterpreted asa declaration or a type of assertion. Each tuple in the relation can then be interpreted as afactoraparticularinstanceoftheassertion.For example, the first tuple in above tableassertsthefactthatthereisaSTUDENTwhoseRoll Number is 1, Name isNipun,Phoneis1234567890 and Age is 26, and so on.

Relational ModelNotation

We will usethe followingnotation inour presentation:

- A relation schema Rof degree n isdenoted by R(A 1, A 2 ,..., A n).
- The uppercase lettersQ, R,S denoterelation names.
- The lowercase lettersq, r,s denoterelation states.
- The letters t,u, v denotetuples.
- In general, the name of a relation schema such as STUDENT also indicates the current setof tuples in that relation—the current relation state—whereas STUDENT (Roll , Name , ...)refers only to the relation schema.
- An attribute A can be qualified with the relation name R to which it belongs by using the dotnotation R.A—for example, STUDENT . Name or STUDENT . Age . This is because thesame name may be used for two attributes in different relations. However, all attributenames in a particular relation must be distinct.
- An n-tuple t in a relation r(R) is denoted by $t = <v_1, v_2, ..., v_n>$, where v_i is the valuecorresponding to attributeA_i.

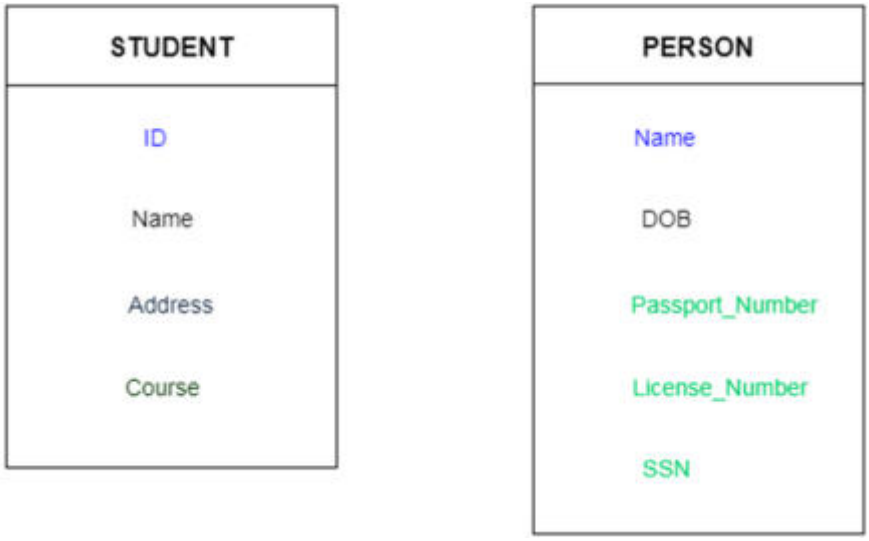


***Fig 1.2** Illustration of Relational Schema*

1.4 KEYS IN RELATIONAL MODEL

Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.

A Key can be a single attribute or a group of attributes, where the combination may act as a key.

Why we need a Key?

In real world applications, number of tables required for storing the data is huge, and the different tables are related to each other as well.

Also, tables store a lot of data in them. A table generally extends to thousands of records stored in them, unsorted and unorganised.

Now to fetch any particular record from such dataset, you will have to apply some conditions, but what if there is duplicate data present and every time you try to fetch some data by applying certain condition, you get the wrong data. How many trials before you get the right data?

To avoid all this, **Keys** are defined to easily identify any row of data in a table.

For example: In STUDENT table of fig 1.2, The attribute ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

Types of key: Different types of keys are shown in the following figure:

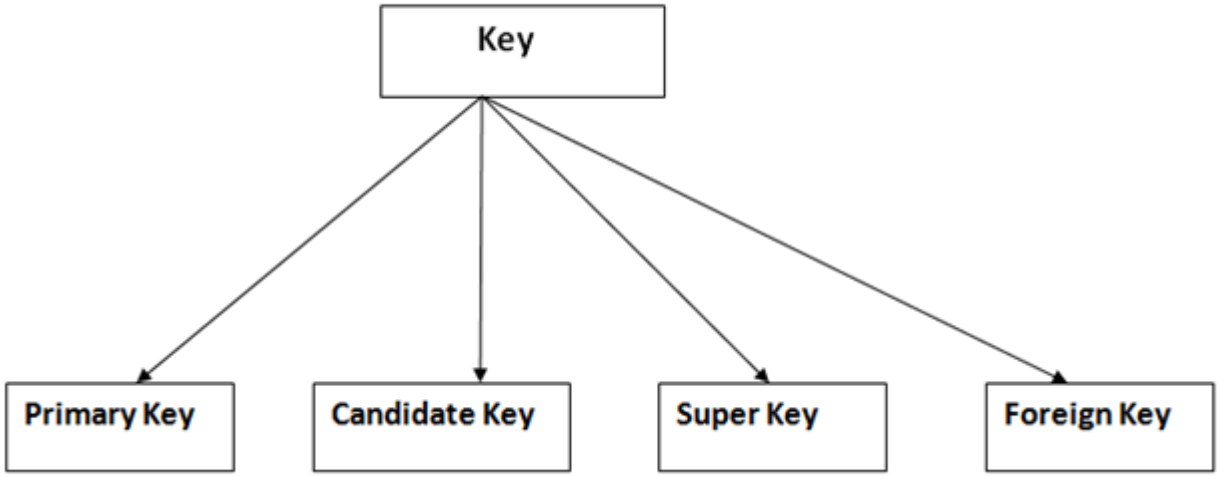


***Fig 1.2** Illustration of Relational Schema*

1. Primary key

- It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.
- In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique.
- For each entity, selection of the primary key is based on requirement and developers.

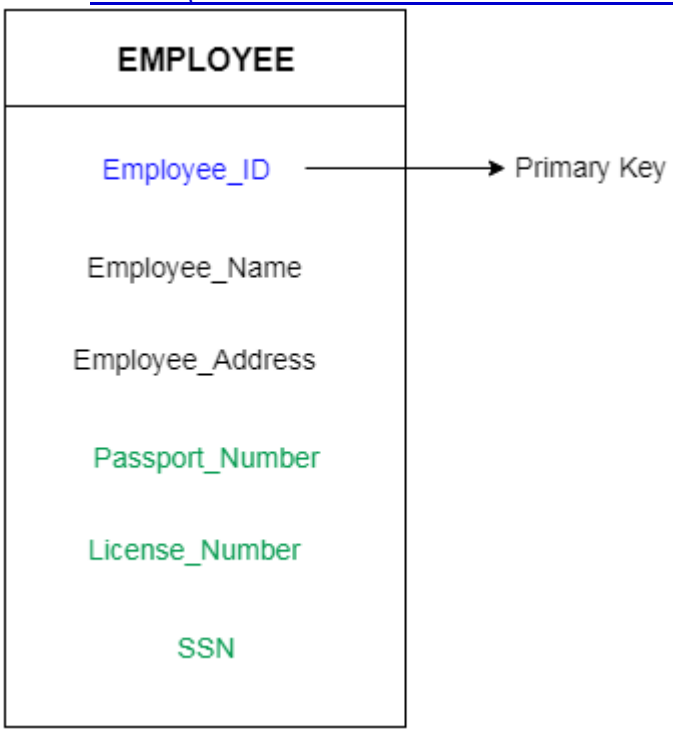


***Fig 1.3** Illustration of Primary Key*

2. Candidate key

Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table. There can be more than one candidate key.

For example: In the EMPLOYEE table, Employee_id is best suited for the primary key. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.

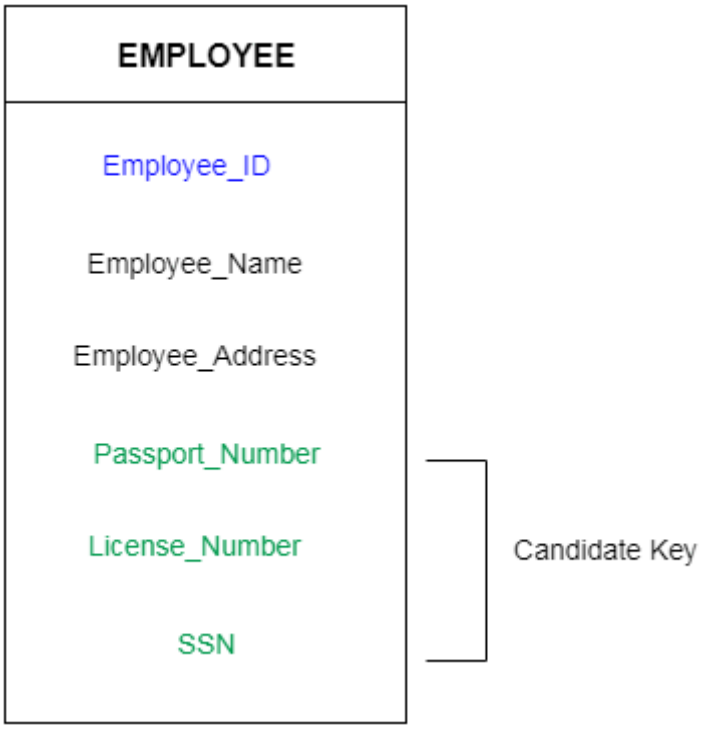


***Fig 1.3** Illustration of Candidate Key*

3. Super Key

Super Key is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

Let's take a simple **STUDENT** table, with the attributes **student_id, name**, **phone** and **age** as shown below-

Table 1.2 *STUDENT*

Student_id	name	phone	age
1	Rohit	1234567890	17
2	Rohit	1234567891	18
3	Mohit	1234567892	19

In the table defined above super key would include student_id, (student_id, name), phone etc.
Confused? The first one is pretty simple as student_id is unique for every row of data, hence it can be used to identity each row uniquely.

Next comes, (student_id, name), now name of two students can be same, but their student_id can't be same hence this combination can also be a key.

4. Foreign key

- Foreign keys are the column of the table which is used to point to the primary key of another table.If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refers. The relation which is being referenced is called referenced relation and the corresponding attribute is called referenced attribute and the relation which refers to the referenced relation is called referencing relation and the corresponding attribute is called referencing attribute
- In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table.
- Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

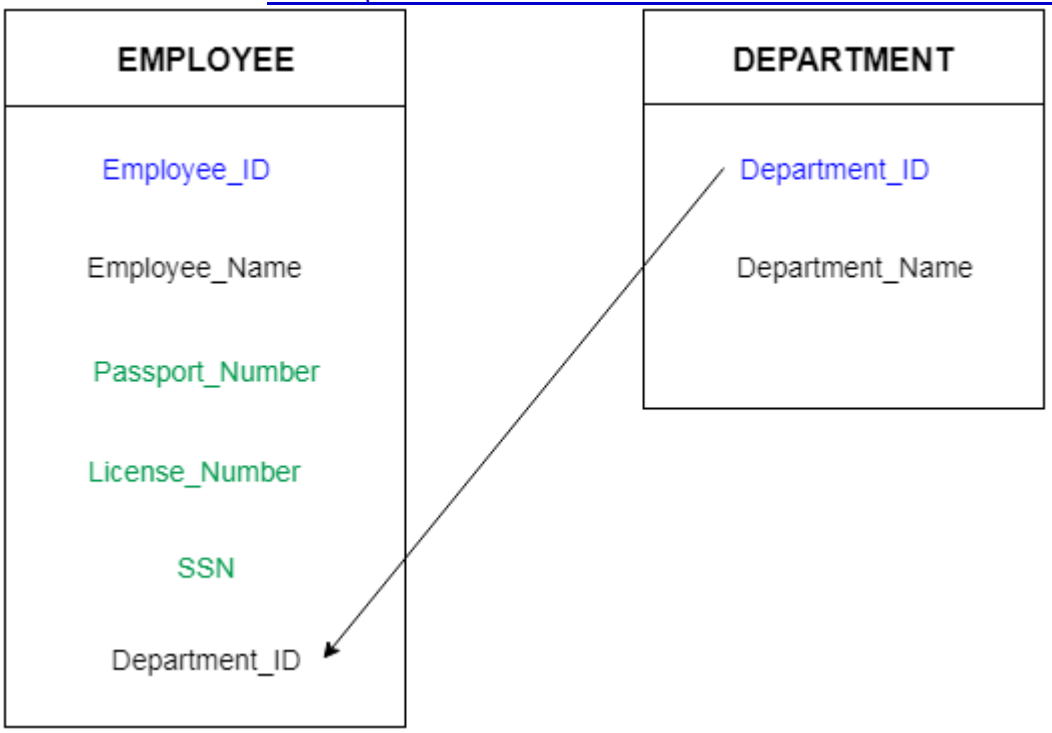


***Fig 1.4** Illustration of Foreign Key*

Moreover the above main keys, we can have the following also:

Composite Key
Key that consists of two or more attributes that uniquely identify any record in a table is called **Composite key**. But the attributes which together form the **Composite key** are not a key independently or individually.

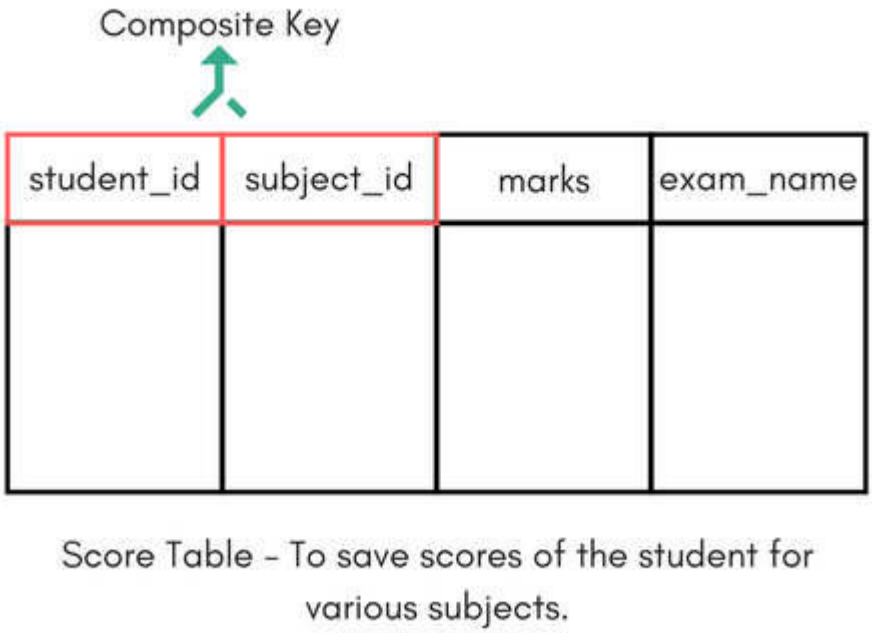


***Fig 1.5** Illustration of Composite Key*

In the above picture we have a Score table which stores the marks scored by a student in a particular subject. In this table student_id and subject_id together will form the primary key, hence it is a composite key.

Secondary or Alternative key
The candidate keys which are not selected as primary key are known as secondary keys or alternative keys.

Non-key Attributes
Non-key attributes are the attributes or fields of a table, other than **candidate key** attributes/fields in a table.

Non-prime Attributes
Non-prime Attributes are attributes other than Primary Key attribute(s)**.**

1.5 RELATIONALMODELCONSTRAINTS

Constraintsenforcelimitstothedataorrestrictionofdatathatcanbeinserted/updated/deleted from a table. The whole purpose of constraints is to maintain the dataintegrity during an update/delete/insert into a table. Constraints on databases can generally bedivided into three main categories:

1. Constraints those are inherent in the data model, We call these inherent model-based constraintsor implicit constraints.

2. Constraints that can be directly expressed in schemas of the data model, typically by specifyingthem in the DDL.

3. Constraintsthatcannotbedirectlyexpressedin the schemas of the data model, and hencemust be expressed and enforced by the application programs. This is known as application-basedor semantic constraints or business rules.

The schema-based constraints include ***domain constraints***, ***key constraints***, ***entity integrity constraints***, and ***referential integrity constraints***.

1.5.1 Domain constraints:

Eachtablehascertainsetofcolumnsandeachcolumnallowsasametypeofdata,basedonitsdata type. The column does not accept values of any otherdata type.

Domain constraints are**user defineddata type**and wecan definethem like this:

DomainConstraint=datatype+Constraints(NOTNULL/UNIQUE/ PRIMARYKEY /FOREIGNKEY / CHECK / DEFAULT).
Let us consider the following STUDENT relation:

Table 1.3

ROLL	NAME	AGE
1	Rahul Sarmah	23
2	Smriti Gogoi	22
3	Shahidul Khan	24
4	Rupak Chetri	A

Here, value **'A'**is notallowed since onlyinteger valuescan be takenby theage attribute.

1.5.2 Key constraints

Anattributethatcanuniquelyidentifyatupleinarelationiscalledthekeyofthetable.Allthevalues of primary key must be unique.

Table 1.4

ROLL	NAME	AGE
1	Rahul Sarmah	23
1	Smriti Gogoi	22
3	Shahidul Khan	24
4	Rupak Chetri	22

This relation does not satisfy the key constraint as here all the values of primary key are notunique.

1.5.3 Entity IntegrityConstraint

Entity integrity constraint specifies that no attribute of primary key must contain a null valuein any relation. This isbecause the presence of nullvalue in the primarykey violates theuniqueness property.

Table 1.5 STUDENT

ROLL	NAME	AGE
1	Rahul Sarmah	23
2	Smriti Gogoi	22
3	Shahidul Khan	24
	Rupak Chetri	22

Thisrelationdoesnotsatisfytheentityintegrityconstraintasheretheprimarykeycontainsa NULLvalue.

1.5.4 Referential IntegrityConstraint-

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a keyattributeofarelationthat canbereferredinotherrelation.

A set of attributes FK in relation schema R 1 is a foreign key of R 1 that references relation R 2 if itsatisfies. R 1 is called the referencingrelation and R 2 is the referencedrelation.

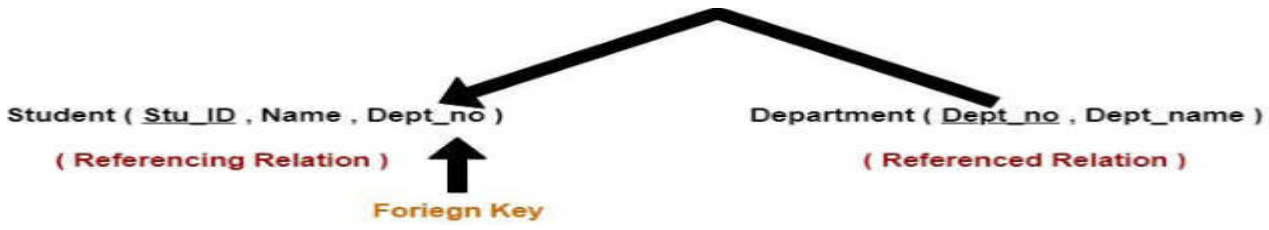


***Fig 1.5** Illustration of the concept of Referential Integrity Constant*

Table 1.6 STUDENT

Roll	Name	Dept_No
1	Rahul Sarmah	D1
2	Smriti Gogoi	D1
3	Shahidul Khan	D2
4	Rupak Chetri	D5

Table 1.7 DEPARTMENT

Dept_No	Dept_Name
D1	Mathematics
D2	Physics
D3	Chemistry
D5	Computer Science

Here,

- The relation‘Student’ doesnot satisfythereferential integrityconstraint.
- This is because in relation ‘Department’, no value of primary key specifies department no.14.
- Thus, referentialintegrity constraintisviolated.

1.5.5 Operations inRelationalModel withConstraintViolations

Four basic operations performed on relational database model are insert, update, delete andselect.

- **Insert Operation:** Insert is used to insert data into the relation Insert can violate any of thefour types of constraints. Domain constraints can be violated if an attribute value is giventhat does not appear in the corresponding domain or is not of the appropriate data type. Keyconstraints can be violated if a key value in the new tuple t already exists in another tuple intherelationr(R).Entityintegritycanbeviolatedifanypartoftheprimarykeyofthenewtupletis NULL.Referentialintegritycanbe violated if the value of any foreign key in trefers to a tuple that does not exist in the referenced relation. If an insertion violates one ormore constraints, the default option is to reject the insertion.If the insertion is not rejectedthen, the insertion violation cancause cascade in the relation. A foreign key with **cascade**deletemeansthatifarecordintheparenttableisdeleted,then thecorresponding recordsin the child table will automaticallybe deleted. This is called a**cascade** delete.

- **Delete Operation:** is used to delete tuples from the table. The Delete operation can violateonly **referential integrity**.This occurs ifthe tuplebeing deleted isreferenced byforeignkeys from other tuples in the database. Here are someexamples.

-

Operation:DeletetheDepartmenttuplewithDeptNO=1.Result:Thisdeletio nisacceptableanddeletesexactlyonetuple.

-Operation:Delete the Studenttuple withDept No= 1.

Result: This deletion is not acceptable, because there are tuples in Department thatrefer to this tuple.

Several options are available if a deletion operation causes a violation. The first option,called restrict, is to reject the deletion. The second option, called cascade. A third option, called setnull or set default, is to modify the referencing attribute values that cause the violation.And alsocombinations of these three options are also possible.

- **Update Operation:**Modify allows you to change the values of some attributes in existingtuples. Consider two tables in figure- EMPLOYEE(Ssn, name, salary, Dno) and DEPARTMENT(Dno,Dname)
  -Operation:Update the salary of the EMPLOYEE tuple with Ssn = '123' to 2800.Result: Acceptable.
  -Operation:Update the Dnoof theEMPLOYEE tuplewith Ssn= '123' to7.
  Result: Unacceptable,becauseit violatesreferentialintegrity.
  -Operation:Update the Ssn of the EMPLOYEE tuple with Ssn = '123' to '321'.Result: Unacceptable, because it violates primarykey constraint

Updating an attribute that is neither part of a primary key nor of a foreign key usually causes noproblems.

- **The Transaction Concept:** A transaction is an executing program that includes somedatabase operations, such as reading from the database, or applying insertions, deletions,or updates to the database. At the end of the transaction, it must leave the database in avalidorconsistentstatethatsatisfiesalltheconstraintsspecifiedon thedatabase schema.A single transaction may involve any number of retrieval operations C and any number ofupdate operations. For example, a transaction to apply a bank withdrawal will typically readtheuseraccountrecord,checkifthereisasufficientbalance,andthenupdatetherecordby the withdrawal amount.

1.6 SYNTHESIZINGERDIAGRAMTORELATIONALSCHEMA

An entity type within ER diagram is turned into a table. Each attribute turns into a column(attribute) in the table. The key attribute of the entity is the primary key of the table which is usuallyunderlined. Derived attributes are ignored and multivalued attributes are represented in anothertable. Taking the following simple ER diagram of **strong entity,**

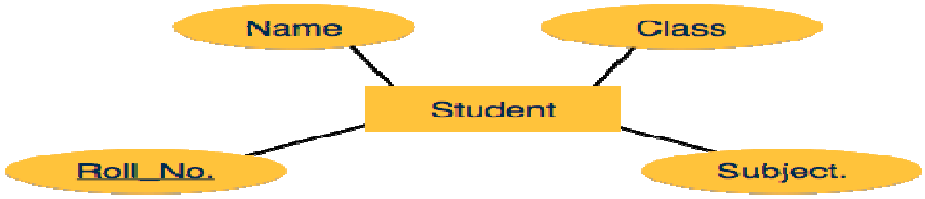


***Fig 1.6** Simple ER diagram*

Theinitialrelationalschemaisexpressedinthefollowingformatwritingthetablenameswith the attributes list inside a parentheses as shown below:For

STUDENT (Roll_No ,Name, Class,Subject)

- Create table for each entity.
- Entity's attributes should become column of tables with their respective data types.
- Declare primary key.

A **multi-valuedattribute** isusually representedwitha double-lineoval.

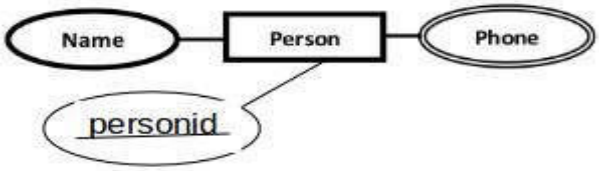


Fig 1.7*ER diagram with multivalued attribute*

Ifyouhaveamulti-valuedattribute,taketheattributeand turnitinto anewentity ortable ofits own.

PERSON(personid ,
name)PHONE(***personid***,ph
one)

PHONE
PhoneID(pk) Personid (fk)phone

PERSON
Personid (pk)name

A **compositeattribute**,

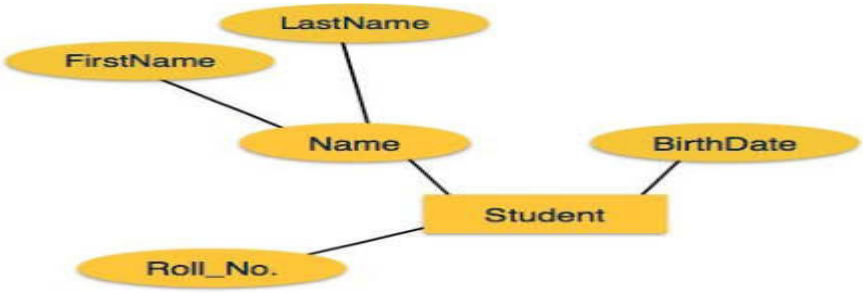


Fig 1.8 *Simple ER diagram showing composite atrribute*

The firstnameand lastnamebecome anindividual attributesin arelational table

STUDENT(Roll_No,BirthDate,FirstName,LastName)

Mapping Relationship

A relationshipis anassociation amongentities.

-One to many

-many to many

-many to one

Many to Many

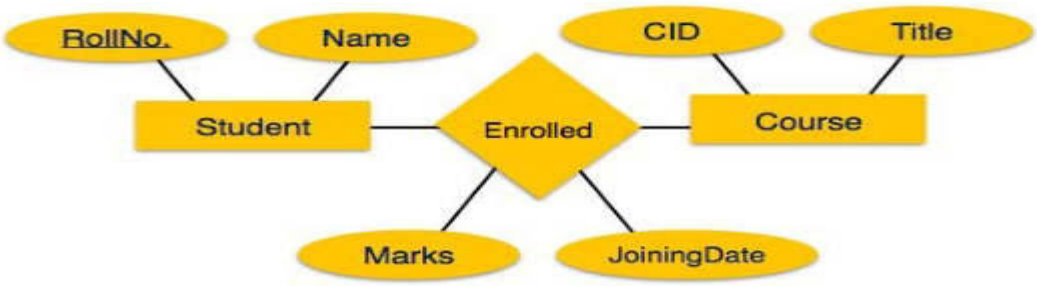


Fig 1.9 ER diagram showing many to many relationship

- Create table foreach entity.
- Entity's attributesshould becomecolumn oftables withtheirrespective datatypes.
- Declare primarykey.
- Create table fora relationship.
- AddtheprimarykeysofallparticipatingEntitiesascolumnoftablewiththeirrespectivedata types.
- If relationship hasany attribute,add each attributeascolumn oftable.

- Declare aprimary keycomposing allthe primarykeys ofparticipating entities.
- Declare allforeign keyconstraints

STUDENT(
RollNo,Name)COURSE(CID, Title)
ENROLLED(RollNo,CID,Marks,JoiningDates)

One to manyand Many toOne

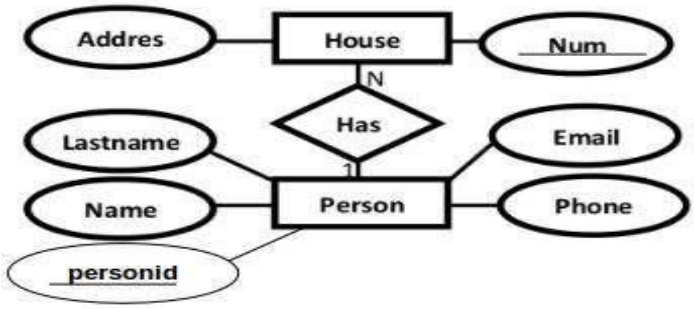


Fig 1.10 ER diagram showing one to many and many to onerelationship

- Create table foreach entity.
- Entity's attributesshould becomecolumn oftables withtheirrespective datatypes.
- Set primary Key of a table as a Foreign key of other tableHOUSE(Num,Address, personid)

PERSON(personid,Name,Lastname,Email,Phone)Or
HOUSE(Num,Address)
PERSON(personid,Name,Lastname,Email,Phone,Num)

Mapping WeakEntity Sets

A weak entityset is onewhich doesnot have anyprimary keyassociated with it.

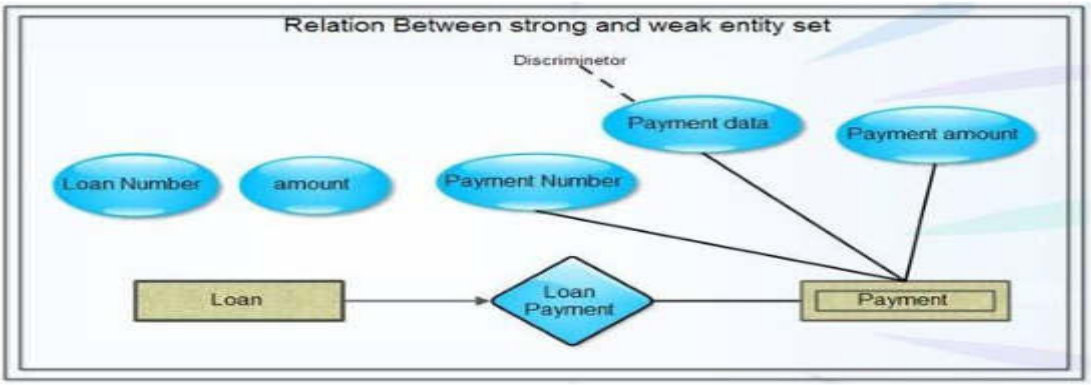


***Fig 1.11 E**R diagram showing relationship between strong and weak entity*

- Create table forweak entityset.
- Add all itsattributes to tableas field.
- Add the primarykey oflinked strong entityset.
- Declare allforeign keyconstraints

LOAN(Loan Number,amount,PaymentNumber)
PAYMENT(LoanNumber,PaymentNumber,Payment Data,Paymentamount)

1.7 SUMMARY

In this unit, you have learnt that the relational model of a database is based on the set theory which provides a simple, yet rigorously defined concept of the manner in which data is perceived by users. Relation is a central concept of the relational model which was first proposed by E.F. Codd in 1969.

You have also been familiarized with key terms related to relational model. Domain, in context of a database, means 'set of possible values' an attribute can acquire. In a database, information is kept in a table in a table in rows and columns. Records are placed in rows attributes are placed in columns. In this model a row is termed as a tuple that gives complete information of the entity. The various types of keys, i.e, primary, alternate, foreign, candidate, logical and super key have been explained in detail.

The unit also laid emphasis on the various relational constraints. Constraints define a condition, which needs to be satisfied while storing data in a database. The different types of constrains- domain constraint, key constraint, entity integrity constraint and referential integrity constraint are explained. Finally, you were taught about the Entity-Relationship diagram. This diagram graphically represents the overall logical structure of the database.

1.8 KEY TERMS

- **Domain**: Set of possible values an attribute can acquire.
- **Tuple**: A row of a relation in a relational data model that gives complete information of an entity.
- **Relational Schema**: Description of the database that is specified during database design.
- **Atomic values**: The values that cannot be divided into subcomponents.
- **Attribute**: It is a column header in a relation which is the properties of entity.
- **Cardinality**: The number of tuples in a relation.
- **Relation**: It is a tabular structure defined by the heading and the data is entered in the body containing a set of rows.
- **Foreign key**: it is an attribute of one relation R_2 whose values are required to match those of the primary key of some relation R_1.
- **Referencing relation**: Relation that contains the foreign key.
- **Super key**: It is the superset of primary key that can uniquely identify any data row in the table
- **Candidate ke**y: It is the set of keys that is minimal and can uniquely identify any data row in the table.
- **Composite key**: Combination of two or more attributes to form a key.
- **Secondary or alternate key**: Only one candidate key is selected as the primary key. The rest of them

are known as alternate key.

- **Constraints**: Constrains enforce limits to the data or type of that data that can be inserted/updated/deleted from a table.
- **E-R diagram**: It describes interrelated things of interest in a specific domain of knowledge.

1.9 QUESTIONS AND EXERCISES

Multiple Choice Questions

1. What is the instance of a Database?
   a) The logical design of the database system
   b) The entire set of attributes of the database put together in a single relation
   c) The data or collection of information stored in a database at a particular moment of time.
   d) The initial values inserted into the database
2. An attribute is a _____ in a relation.
   a) Row
   b) Column
   c) Value
   d) Tuple
3. Constraints define a condition, which needs to be satisfied while storing data in a _____.
   a) Data
   b) Database
   c) Attribute
   d) Task
4. An alternate key is a candidate key that is not the _____.
   a) Entity
   b) Attribute
   c) Secondary Key
   d) Primary Key
5. Advantages of relational model are
   a) Simplicity
   b) Data Integrity
   c) Flexibility
   d) All of the above
6. The name of the relation along with necessary attributes is represented with the help of a
   a) Relation Schema
   b) ER diagram
   c) flowchart
   d) Venn diagram
7. Candidate key of a relation can be
   a) Only One
   b) Exactly Two
   c) Zero
   d) More than One
8. Attribute is properties of an/a

a) Data
b) Instance
c) Attribute
d) Entity

9. An alternate key is a/an
   a) Candidate key
   b) Super key
   c) Foreign Key
   d) Primary Key
10. The pre-defined value and scope that present in every attribute is called
    a) Main
    b) Primary
    c) Domain
    d) None of the above

Answers: 1. (c), 2.(b), 3.(b), 4(d), 5(c), 6(a), 7(d), 8(d), 9(a), 10(c)

Fill in the Blanks

1. Primary key of a table never contains NULL and _____ values.
2. A _____ key allows us to identify uniquely an entity in the entity set.
3. What is the degree of a table with 1000 rows and 10 columns?
4. In a relational database a referential integrity constraint can be specified with the help of _____.
5. The format or data type must be specified for _____.
6. _____ is the basic building block of RDBMS.
7. Multivalued attribute is capable of storing more than one value in a single _____.
8. _____ attribute is represented with double ovals in an ER diagram.
9. _____ key is primary key of another table.
10. Four basic operations performed on relational database model are Insert, update, delete and _____.

Answers: 1. Duplicate, 2. Super, 3. 10, 4. Foreign key, 5. Domain, 6. Entity, 7. Attribute, 8. Multivalued, 9. Foreign, 10. Select.

State whether TRUE or FALSE

1. Relational model is a database is based on the set theory of mathematics.
2. An alternate key is a candidate key that is not the primary key.
3. The Relation Schema describes the relation name, attribute and their names.
4. A column in a table represents a relationship among a set of values.
5. The term tuple is used to refer to a row.
6. An entity is the basic building block of RDBMS.
7. A relation can't have more than one candidate key.
8. The relational schema is the name of the relation only.
9. Insert operation is not performed in the relational model.
10. Attributes are the characteristics of entity.

Answers: 1. True, 2. False, 3. True, 4. False, 5. True, 6. True, 7. False, 8. False, 9. False, 10. True

Match Column A with Column B

1.	Relation that contains the foreign key	A.	E-R diagram
2.	The number of tuples in a relation is termed as	B.	Entity Integrity
3.	A mechanism that allows uniquely identify rows in a table is termed as	C.	Cardinality
4.	Redundant value means	D.	SQL
5.	Example of an RDBMS language	E.	Domain
6.	A set of possible values an attribute can acquire can be termed as	F.	E.F Codd
7.	The Relational model was proposed by	G.	Duplicate vale
8.	The graphical representation of the overall logical structure of a database	H.	Referencing relation
9.	Entity	I.	Oval symbol in E-R diagram
10.	Attribute	J.	Rectangle symbol in E-R diagram

Answers: 1. G, 2. C, 3. B, 4. G, 5. D, 6. E, 7. F, 8. G, 9. J, 10. I

Short-Answer Questions

1. What are the different features of relational model?
2. What are the advantages of the relational model?
3. State the various features of relations.
4. Discuss the various components of domain information.
5. What is key? What is the importance of key in a relation?
6. With the help of example define a tuple in a relation.
7. What is constraint? Why they are important?
8. With the help of an example define a super key.
9. With the help of an example define foreign key.
10. With the help of an example define candidate key.

Long-Answer Questions

1. Discuss the relational model.
2. Explain the different types of constraints with example.
3. Explain the different types of keys with example.
4. With the help of illustrations explain the E-R diagram.
5. What are domains? Explain its constraints.
6. Discuss the E-R diagram in the context of relational model.
7. Explain the different types of attributes with example.
8. Explain the different types of entity with example.
9. Discuss the advantages of relational model.
10. Discuss the tabular structure that is used to represent a relation in relational model.

UNIT-6
NORMALIZATION AND FUNCTIONAL DEPENDENCIES

Contents:

6.0 INTRODUCTION

A relational database schema comprises of a number relational schemas, where each relational schema is designed by grouping the related attributes. While there are numerous groupings possible for the same set of attributes, not all the groupings lead to a ”good” design. A good design can be easily understood by the users, follows a logical organization of the attributes, and minimize redundancy. In this model, we are going to discuss some informal guidelines to measure the ”goodness” of a relation. Another important concept- functional dependency, which refers to the constraints that exist among the attributes of a relation, is also introduced in this module. Functional dependency is an important tool to measure how appropriate is the grouping of the attributes in a relation. This module also discusses normal forms and the process of normalization. A relational schema is said to be in a normal form if it meets certain desirable properties. The process of converting a relation into a normal form is called normalization. Functional dependency and constraints on key attributes can be used to analyze which normal form relation is and also help in further normalizing the relation if possible. Some other advanced concepts like - multivalued dependency, join dependency and lossless join property are also presented in this module.

6.1 OBJECTIVE

After completion of this module, you will be able to -

- *list* the informal measures to assess the quality of relational schema design.
- *describe* the various functional dependencies and normal forms.
- *understand* the concept of null values, redundant information, and spurious tuples and how to eliminate these by performing normalization.
- *apply* the concept of database normalization (1 NF, 2NF, 3 NF, etc.) to create an efficient relational schema design to organize the data logically and meaningfully and eliminate redundancy.
- *analyze* whether a given relational schema design follows the basic guidelines of design or not.
- *evaluate* in which normal form a given relational schema is, and if possible, convert it to a higher normal form.
- *create* good relational schema designs by applying the algorithms for losses join properties.

6.2 INFORMAL DESIGN OUTLINES FOR RELATIONAL DATABASES

A relational schema can be defined as a set of relational tables and associated items related to each other. While it is possible to design multiple relational schemas for the same problem, the challenging task is to choose the good one. The design guidelines help us to assess the quality of the relational schemas and thus enable us to achieve good quality relational schema designs. The following are the four informal design guidelines-

- The semantics of the Relation
- Minimizing redundancy
- Reduction of the null values in tuples.
- Discarding the possibility of generating spurious tuples.

6.2.1 Semantics of a Relation

When we arrange attributes to construct a relation schema, we presume that each attribute has a specific meaning. This meaning, or semantics, describes how to interpret the attribute values recorded in a tuple of the relation, or how the attribute values in a tuple relate to one another. For example, in figure 1.1, the relations FACULTY, COURSE, and DEPARTMENT have distinct semantics. The attributes in the relations are also self-explanatory. The relation DEPARTMENT represents details of a department- *department number* (D_NO), *name of the department* (D_NAME), and the *department email id* (D_EMAIL). D_NO is the primary key of the relation, inferring that each department has a unique department number. The relation FACULTY, on the other hand, outlines the details of a faculty, like- *name*(F_NAME), *id*(F_ID), *gender*(F_GENDER), and *date of birth*(F_DOB). F_ID is the primary key of the relation. The attribute, D_NO in FACULTY is the foreign key from the relation DEPARTMENT, indicating the implied relationship between the two relations. Similarly, the relation COURSE also has a distinct meaning. It depicts course details like - *name of the course* (C_NAME), *course code* (C_CODE), and *credit* (CREDIT). The attributes, F_ID and D_NO, in COURSE are the foreign keys from FACULTY and DEPARTMENT respectively and the attribute C_CODE is the primary key.

FACULTY

F_NAME	F_ID	F_GENDER	F_DOB	D_NO
	p.k.			

DEPARTMENT

D_NO	D_NAME	D_EMAIL
	p.k.	

COURSE

C_CODE	C_NAME	C_CREDIT	F_ID	D_NO
p.k.			f.k.	f.k.

Fig 6.1: Relational Schema design with clear semantics

Guideline 1: Create a relationship schema that is self-explanatory and thus simple to understand. If a relation schema relates to a single entity type or relationship type, the meaning is usually obvious. However, in a single relation, if attributes from different entity types and relationship types are combined, the relation becomes semantically unclear.

6.2.2 Minimization of Redundancy

Redundancy is the repetition of the same fact again and again across multiples places in the same database. Apart from wastage of storage space, redundancy also results in various other side effects. In designing a relational schema, therefore, one of the most important goals is to minimize redundancy across. Proper grouping of the attributes in a relation schema helps significantly in minimizing redundancy. This can be illustrated with the example in figure 6.2. The relations FACULTY_DEPT and COURSE_DEPT are being designed to represent the faculties and the courses. The relations cover all the aspects the such as - which faculty works for which department and which course is offered by which department. However, if compared with the design in figure 6.1, the design in figure 6.2 consumes more storage space. In figure 6.1, the department number and department email id have been mentioned only once for a particular department in the DEPARTMENT relation. However, in figure 6.2, in the FACULTY_DEPT relation, these two details are repeated for every employee that belongs to a particular department. The same is also the case with the COURSE_DEPT relation.

Apart from wastage of storage space, redundancy leads to another serious issue of update anomalies. Insertion, deletion, and modification anomalies are the three categories of update anomalies. A

FACULTY_DEPT

F_NAME	F_ID	F_GENDER	F_DOB	D_NO	D_NAME	D_EMAIL
	p.k.					

COURSE_DEPT

C_NAME	C_CODE	CREDIT	F_ID	D_NO	D_NAME	D_EMAIL
	p.k.		f.k.			

Fig 6.2: Relational Schema design with redundancy

brief discussion of each is presented in this section.

6.2.2.1 INSERT ANOMALIES

Consider table 6.1, which is the populated table for the relation FACULTY_DEPT. Every time we enter a faculty detail, we must also enter the corresponding department details. While entering these details, one must be careful about entering all the fields correctly. For example, two faculty members working for department number 1, must have the same values for the attributes D_NAME and D_EMAIL. However, as we can observe from table 6.1, the faculty members with id 123 and 124 work for the same department but D_NAME and D_EMAIL values are different. This results in the inconsistency of the database.

Table 6.1: Populated FACULTY_DEPT table

F_ID	F_NAME	F_GENDER	F_DOB	D_NO	D_NAME	D_EMAIL
123	Ravi Singh	Male	06-02-1972	1	Geography	geo@gmail.com
124	Rahul Bose	Male	05-04-1980	1	English	eng@gmail.com
125	P. Joseph	Male	12-07-1979	2	Social Science	Sssc@gmail.com
126	Sima Mishra	Female	08-11-1985	3	Mathematics	maths@gmail.com

Another difficulty is that there is no option to enter the details of a department which has not appointed any faculty yet. The department details can be entered into FACULTY_DEPT relation, only where there is at least one faculty who is working in that department. These issues will not occur in the design of figure 6.1 as the department details are not clubbed with the faculty details and thus there is no redundancy.

6.2.2.2 DELETION ANOMALIES

This can be inferred from the second issue in insertion anomaly. From table 6.1, if we delete the faculty information with id 125, then we will lose all the details of department 2. This is because the faculty, with id 125, is the only faculty working in department 2. The same will be the problem if we delete the tuple with id 126. This problem does not occur in the database of figure 6.1, as deleting a tuple from the FACULTY relation will not cause any deletion of tuples from the DEPT table. Thus all the tuples in DEPT will still be intact.

6.2.2.3 MODIFICATION ANOMALIES

This again can be inferred from the first issue of insertion anomaly. If there are some changes made in one department details- such as the department name, the same has to be updated in all the tuple of the FACULTY_DEPT relation with that department number. For example, if we wish to change the department email id of department number 1, then we need to update the same in all the faculty tuples that are working in department 1. Even if we forget to update it in one tuple, the database will be inconsistent.

Guideline 2: Design a schema with minimum redundancy, so that there are no update anomalies. In case of any unavoidable redundancy, the program must be designed to tackle all the related insertion, deletion, and modification anomalies.

6.2.3 Reducing the NULL values in tuples

A NULL value for an attribute in a tuple can have multiple interpretations, such as-

- The attribute doesn't apply to this tuple.
- The value of the attribute is unknown for this tuple
- The value of the attribute is known but has not been recorded yet.

The NULL values not only result in wastage of space but also creates problems in many operations such as JOIN operations, aggregate operations such as COUNT or SUM, etc.

GUIDELINE 3: While designing a relational schema we should group the attributes in such a way that produces as few NULL values as possible.

6.2.4 SPURIOUS TUPLES

Many a time, a relational schema has to be decomposed into smaller relations. Inappropriate decomposition of the relation may result in some information that originally did not exist in the original relation. For example, let a relation R be decomposed into two smaller relations R1 and R2. If the natural join of R1 and R2 produces any extra tuple that does not exist in the original relation R, then that tuple is called the spurious tuple. Let's consider the relation R in table 6.2(a). Decomposition of R into relations R1(A, B) and R2(B, C) will result in the following two relations as shown in tables 6.2(b) and 6.2(c) respectively. Now the natural join over R1 and R2 will result in table 6.2(d). As we may observe, table 6.2(d) has two extra tuples that are originally not present in R. These are called spurious tuples. Spurious tuples represent wrong or invalid information and thus leads to the inconsistency of the database.

Table 6.2 (a): Spurious tuples: Relational schema R

A	B	C
a1	b1	c1
a2	b1	c2

Table 6.2 (b):Spurious tuples: Relational schema R1

A	B
a1	b1
a2	b1

Table 6.2 (c):Spurious tuples: Relational schema R2

Table 6.2 (c):Spurious tuples: Relational schema R2

B	C
b1	c1
b1	c2

Table 6.2 (d):Spurious tuples: R1*R2

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a1	b1	c2

Guideline 4: Decompose a relation into multiple relations in such a way that the natural join of the smaller relations does not produce any spurious tuple. This can be done by having relations, where the common attributes are either the primary key or foreign key of the relations. In case of an unavoidable situation where two relations have common non-key attributes, extra care should be taken not to join such relations.

Check Your Progress

1. Redundancy in a database leads to _____, _____ and _____ anomalies.
2. If the natural join of two relations results in extra tuples that are not in the original relation, then those tuples are called as _____ tuples.
3. State true or false
    a. NULL vales in a relation always have specific meaning.
    b. Spurious tuples represent wrong or invalid information.
    c. A good relation always has complex semantics.

SAQ

1. Briefly discuss the four informal guidelines for designing a good relational schema.
2. List the problems that occur due to redundancy in a database.
3. What are the spurious tuples? Give one example.
4. List any three reasons, why Null values are not desirable in a relation.

6.3 FUNCTIONAL DEPENDENCIES

In Database Management System (DBMS), functional dependency (FD) refers to the relationship between two attributes in a table or relation. For any relation, if the value of the set of attributes Y is determined by the value of the set of attributes X, then Y is said to be functionally dependent on X. This is symbolically represented by $\boldsymbol{X \rightarrow Y}$. This notation can be read as "*X functionally determines Y*" or "*Y is functionally determined by X*". If this functional dependency holds, then for every valid instance of X there will be a unique value of Y in the table. Usually, the functional dependency exists between a prime key attribute and a non-key attribute(s). Thus, in a relation R, if two tuples, say t1 and t2 have the same values of X, then they must have the same values of Y as well. A functional dependency is the property of the attributes in a relation. It must hold for every tuple in a relation. The concept of functional dependency was introduced by E.C. Codd. It helps in avoiding bad design and in avoiding data redundancy. To better understand functional dependency, let us consider the relations in figure 6.1. Tables 6.3, 6.4, and 6.5 are the populated tables for the relations FACULTY, DEPARTMENT, and COURSE respectively.

Table 6.3: Populated FACULTY relation

F_ID	F_NAME	F_GENDER	F_DOB	D_NO
123	Ravi Singh	Male	06-02-1972	1
124	Rahul Bose	Male	05-04-1980	1
125	P. Joseph	Male	12-07-1979	2
126	Sima Mishra	Female	08-11-1985	3

Table 6.4: Populated DEPARTMENT relation

D_NO	D_NAME	D_EMAIL
1	Geography	geo@gmail.com

Table 6.4: Populated DEPARTMENT relation

2	Social Science	Sssc@gmail.com
3	Mathematics	maths@gmail.com

Table 6.5: Populated COURSE relation

C_NAME	C_CODE	CREDIT	F_ID	D_NO
Physical geography	230	3	123	1
Anthropology	231	3	125	2
Graph Theory	232	3	126	3
Real Analysis	233	4	126	3

We can see from table 6.4, D_NAME is uniquely determined by D_NO. Thus in this relation the functional dependency, D_NO→ D_NAME holds. Similarly, a few other functional dependencies that hold in the relations DEPARTMENT, FACULTY, and COURSE are-

- F_ID → F_GENDER
- F_ID → F_GENDER
- D_NO → D_EMAIL
- C_NAME → C_CREDIT
- C_CODE → F_ID
- C_NAME → F_NAME

6.3.1 TYPES of FUNCTIONAL DEPENDENCIES

Functional dependencies can be classified into three forms-

- Trivial functional dependencies
- Non-trivial functional dependencies
- Multivalued functional dependencies
- Transitive functional dependencies
- Full Functional dependencies

6.3.1.1 Trivial functional dependencies

A functional dependency X→Y is said to be trivial if Y is a subset of X. For example, the functional dependency {F_ID, F_NAME}→F_NAME is a trivial functional dependency since F_NAME is a subset of {F_ID, F_NAME}. Similarly, {D_NUMBER, D_NAME}→DNAME is also another example of trivial functional dependency.

6.3.1.2 Non-trivial functional dependencies

Unlike in trivial function dependency, in non-trivial functional dependency, the set of attributes on the right-hand side is not a subset of the attributes on the left-hand side. In other words, if X→Y, and Y is not a subset of X, then the functional dependency is said to be non-trivial. For example, the functional dependency, F_ID→ {F_NAME} in table 6.3 is an example of non-trivial functional dependencies. Another example of non-trivial functional dependency from table 6.5 is COURSE_ID →C_NAME.

6.3.1.3 Multivalued functional attributes

If there exists a functional dependency of the form X→ {Y, Z} such that there is no dependency between Y and Z, then the FD is said to be a multivalued functional dependency. In other words, multi values dependency occurs when two or more attributes in a table are functionally independent of each other but are functionally determined by a specific attribute. Multivalued dependency is represented by the symbol "→→". For multivalued dependency, we must have at least three attributes in the relation. A more detailed discussion of the multivalued attribute is presented in section 6.5.

6.3.1.4 Transitive functional dependencies

In a relation, if the functional dependencies X→Y and Y→ Z exist, then the functional dependency X→Z also exists. This is called transitive dependency. For transitive dependency to exists, there must be at least three attributes in the relation. For example, consider table 6.6. In this table, S_ID determines S_Name and S_Name determines S_Age. Due to transitive dependency, we can also state that S_ID determines S_Age. Thus we can summarise as-

- S_ID→S_Name
- S_NAME→S_Age
- S_ID→S_Age [due to transitivity]

Table 6.6: Example of transitive dependency

S_ID	S_Name	S_Age
1	Ravi	20
2	Rohan	19
3	Sukanya	21
4	Puja	20

6.3.1.5 Full Functional dependencies

A functional dependency X→Y is said to be a full functional dependency if removal of any attribute from X means the functional dependency doesn't exist any longer. For example, consider table 6.7, showing the number of hours (per week) assigned to the employees for different projects.

In this table {E_ID, PROJECT_ID}→HOURS. If we remove any attribute from the left-hand side then the dependency no longer holds as neither E_ID→HOURS nor PROJECT_ID→HOURS. Thus, it is an example of full functional dependency.

Table 6.7: Example of full functional dependency

E_ID	PROJECT_ID	HOURS
1	3	16
1	2	20
2	1	12
3	3	10

6.3.2 INFERENCE RULES for FUNCTIONAL DEPENDENCIES

While designing a relational schema R, the designer also specifies a set of functional dependencies. Let's consider that this set of functional dependencies is denoted by F. Usually, the schema designers list only the functional dependencies that are semantically obvious. Apart from the functional dependencies in F, it is possible to infer several other functional dependencies that hold in any legal relation instances in R. For example, one of the functional dependencies that hold for the FACULTY relation in figure 6.1 is F_ID→ {F_NAME, F_GENDER, F_DOB, D_NO}. We can easily infer a number of other functional dependencies from the given FD. Some of these are-

- F_ID→ {F_NAME}
- F_ID→{F_GENDER, F_DOB}
- F_NAME→{F_DOB}

It is not practically possible to mention all the functional dependencies that hold in a relation schema. However, we can systematically infer the other functional dependencies with the help f the inference rules. This set of inference rules were first introduced by William W. Armstrong in 1974. These rules are thus also called Armstrong's axioms. These axioms define a set of rules which, if applied repeatedly, generate all the other functional dependencies that can be inferred from a set of functional dependencies originally specified by the designer.

Armstrong's Axioms:

- **IR1: Axiom of reflexivity:**

  If Y is a subset of X, i.e, Y⊆X, the X→Y.

- **IR2: Axiom of augmentation:**

  If in a relation the functional dependency X→Y holds, then the functional dependency XZ→YZ also holds in that relation.

- **IR3: Axiom of transitivity:**

  In a relation if the two functional dependencies X→Y and Y→Z hold, the functional dependency X→Z also holds.

Armstrong showed that the inference rules from IR1 to IR3 are sound and complete. Soundness means - if we consider any relational schema R with a set of functional dependencies as F, then for any legal relational instance r of R, which satisfies the functional dependencies in F also satisfies the functional dependencies inferred using IR1 to IR3. On the other hand, the rules are complete in the sense that, application of the rules IR1 to IR3 over F until no additional functional dependencies are generated will result in all the possible dependencies that can be inferred from F.
Some other important secondary rules that can be derived from the above inference rules are-

- **IR4:Decomposition or Projective rule**

If X→YZ holds in a relation, then the functional dependencies X→Y and X→Z also hold.

Proof:

Step 1: X→YZ	[Given]
Step 2: YZ→Y	[Applying IR1, as Y⊆YZ]
Step 3: X→Y	[Applying IR3 on step 1 and step 2]

Similarly, we can prove that X→ Z.

- **IR5: Union or additive rule:**

If the functional dependencies X→Y and X→Z hold in a relation, then the functional dependency X→YZ also holds in the relation.

Proof:

Step 1: X→Y	[Given]
Step2: X→Z	[Given]
Step3: XY →YZ	[Applying IR2 on step2]
Step4: XX→XY	[Applying IR2 on step1]
Step5: X→XY	[As XX=X]
Step6: X→YZ	[Applying IR3 over steps 5 and 3]

- **IR6: Pseudo Transitivity**

If the functional dependencies X→Y and WY→Z hold in a relation, then the relation WX→Z also holds in the same relation.

Proof:

Step 1: X→Y	[Given]
Step 2: WX→WY	[Applying IR2 on step 1]
Step 3: WY→ Z	[Given]

Check Your Progress

4. The functional dependency {ISBN, Book_Name}→Book_Name is an example of _____ functional dependency.
5. If in a relation named COMPANY, C_name→ C_location and C_location→ C_pincode, then we can say infer that C_name→ C_pincode due to _____.
6. If A→B and A→ C then, due to additive rule we can infer that _____ .
7. State true or false
   a. Functional dependency represents the relation between two tables.
   b. If A and B are two sets of attributes and A is a subset of B, then we can say that B → A.
   c. The three axioms- reflexivity, augmentation and transitivity represent the complete and sound sound set of inference rules.

Step 4: WX→Z [Applying IR3 on step2 and step 3]

6.3.3 CLOSURE of FUNCTIONAL DEPENDENCIES

For any relational schema R, if the set of functional dependencies is specified as F, then the set of all the functional dependencies that can be inferred from F, is called the closure of F. The closure of F is denoted as F^+. The closure of a set of functional dependencies F, F^+,can be derived by repeatedly applying the inference rules over F unless a point is reached where no additional functional dependencies are generated.

To find closure of a functional dependency F systematically, first, we need to identify each set of attributes X that occurs as the left-hand side of any functional dependence in F. The next step is to identify the set of all attributes that are dependent on X. As a result, for each such set of attributes X, we find the set of attributes that are functionally determined by X based on F; this is referred to as the closure of X under F and is denoted by X^+.

Example: Let's consider a relation STUDENT (ID, NAME, CGPA, LOCATION) with the set of functional dependencies specified is F={ID → NAME, NAME→CGPA, ID →LOCATION}

For finding the closure of F, we need to find the closure of the attribute present in the left-hand side of the functional dependencies. Thus, we need to find ID^+ and $NAME^+$

Step 1: ID → ID [Due to IR1]
Step 2: ID →NAME [Given]
Step 3: ID →{ID, NAME} [Applying IR5 on step1 and 2]
Step 4: ID →LOCATION [Given]
Step 5: ID →{ID, NAME, LOCATION} [Applying IR5 on step3 and 4]
Step 6:NAME→CGPA [Given]
Step7: ID → CGPA [Applying IR3 on step 2 and step 6]
Step 8: ID→{ID, NAME, LOCATION, CGPA} [Applying IR5 over step 5 and 7]

Thus ID^+={ID, NAME, LOCATION, CGPA}
Similarly, we can find that
$NAME^+$={NAME, CGPA}
Thus the closure set with respect to F is:
ID^+={ID, NAME, LOCATION, CGPA}
$NAME^+$={NAME, CGPA}

6.3.4 EQUIVALENT SETS of FUNCTIONAL DEPENDENCIES

Let F and G be two sets of functional dependencies for a relational schema R. G is said to be **covered** by F if all the dependencies in G can be inferred from F. In other words, F **covers** G if G^+ is a subset of F^+i.e. $G^+ \subseteq F^+$. On the other hand, F and G are said to be **equivalent** if the following contains are satisfied:

- All the functional dependencies in F can be derived from functional dependencies in G.
- All the functional dependencies in G can be derived from functional dependencies in F.

In other words, if the closure of is equal to the closure of G, i.e., if $F^+ = G^+$, then F and G are said to be equivalent. Alternatively, we can also say that F and G are equivalent if F covers G and G covers F.

Example: A relation R (P, Q, R, S, T) is has two set of FDs F and G specified as follows-
F={P → Q, PQ → R, S → PR, S → T}
G= {P → QR, S → PT}

Determine whether F covers G:

Step-1:

- $(P)^+$ = { P, Q, R } // closure of left side of P → QR using set G
- $(S)^+$ = { P, Q, R, S, T } // closure of left side of S → PT using set G

Step-2:

- $(P)^+$ = { P, Q, R } // closure of left side of P → QR using set F
- $(S)^+$ = { P, Q, R, S, T } // closure of left side of S → PT using set F

From Step-1 and Step-2, we can conclude that F covers G i.e. F ⊇ G, as the FDs in F can determine all the attributes that are determined by the FDs in G.

Determining whether G covers F

Step-1:

- $(P)^+$ = { P, Q, R } // closure of left side of P → Q using set F
- $(PQ)^+$ = { P, Q, R } // closure of left side of PQ → R using set F
- $(S)^+$ = { P, Q, R, S, T } // closure of left side of S → PR and S → T using set F

Step-2:

- $(P)^+$ = { P, Q, R } // closure of left side of P → Q using set G
- $(PQ)^+$ = { P, Q, R } // closure of left side of PQ → R using set G
- $(S)^+$ = { P, Q, R, S, T } // closure of left side of S → PR and S → T using set G

From Step-1 and Step-2, we can conclude that G covers F i.e. G ⊇F, as the FDs in G can determine all the attributes that are determined by the FDs in F.

Thus we can conclude that F=G

6.3.5 MINIMAL COVER of FUNCTIONAL DEPENDENCIES

A set of FDs, F_{min}issaid to be the minimal cover of another set of functional dependency F if-

- $F_{min}$is the minimal set of functional dependencies and
- $F_{min}$is equivalent to F.

A set of functional dependency F_{min} is said to be minimal if the following conditions are satisfied-

- Each functional dependency in the set has only one attribute to the Right Hand Side (RHS)
- Any dependency in F_{min},say X→Z, can not be replaced by some other functional dependency Y→Z, where Y ⊂ X.
- If we remove any dependency from F_{min}, the resultant set will no longer be equivalent to F_{min}.

Algorithm 6.1:Finding minimal cover of a functional dependency

Step1: Identify the functional dependencies that have more than one attribute to the RHS. Transform them into a series of functional dependencies having only one attribute to the RHS.

Step2: Remove the redundant attributes on the left-hand side.

Step3: Eliminate the redundant functional dependencies.

Example:

Let's consider the functional dependency F={P → R, PQ → R, R → SU, RS → U, TR → PQ, TU → R}

Step 1: F1={P→R, PQ→R, R→S, R→U, RS→U, TR→P, TR→ Q, TU→R}

Step 2: To find the redundant attributes, we need to first find the closure of each attribute.

i. P^+-=PRSU
ii. Q^+ =Q
iii. R^+=RSU
iv. S^+=S
v. T^+=T

We can see from (i) that P^+ includes R. Thus Q is extraneous in PQ→R and thus Q can be removed. So, this dependency can be rewritten as P→R.

From (iii), we can see that R^+ includes U, thus in RS→U, S is extraneous. So, we can rewrite it as R→U.

Thus the new reduced set of functional dependencies can be written as,

F2={P→R, R→S,R→U, TR→P, TR→Q, TU→R}

Step 3: The last step is to eliminate the redundant dependencies

Here, TU→R is redundant as R can be determined using P due to the functional dependency P→R. Thus the final set of minimal cover for F is

F_{min}={P→R, R→S,R→U, TR→P, TR→Q}

Check Your Progress

8. If a and B are two sets of functional dependencies such that A covers B and B covers A, then A and B are said to be _____.
9. If G is a set of functional dependencies, then its closure is denoted by _____ .
10. In a relation R, the functional dependencies A→BC and B→D hold. The closure of the attribute A in that relation is _____.
11. State true or false
    a. Two sets of functional dependencies F and G are said to be equivalent if $F^+=G^+$.
    b. F is the minimal cover of G. If we remove any functional dependency from F

SAQ

1. Define functional dependency and it's types.
2. List the Armstrong's axioms for functional dependency.
3. Define multivalued attribute with an example.
4. Write the algorithm to find minimal cover of a set of functional dependencies.

6.4 NORMALIZATION and NORMAL FORMS

In DBMS normalization is used to minimize redundancy. As we have already discussed in section 6.2.2, redundancy in relations may lead to insert, delete, and modification anomalies. Normalization helps in breaking down big relations into smaller relations and ensures that data is stored logically with minimal redundancy.

Normal form a relation reflects its degree of normalization. It refers to the highest normal form condition that the relation satisfies. The normal forms that will be discussed in this unit are- first normal form (1NF), second normal form (2NF), third normal form (3NF), Boyce- Codd normal form (BCNF), and fourth normal form (4NF). In practice, the database designer has to normalize the relations to the highest normal form possible (usually up to 3NF, BCNF, or 4NF).

6.4.1 DEFINITION of KEYS

- **Super Key**: For a relational schema R={A, B, C,….., J}, a super key, S, is a set of attributes such that S⊆R and for each legal tuple in R, S has a unique value.
- **Key**: A key, K is a minimal super key. This implies that if we remove any attribute from K, then K will no longer hold the super key property.
- **Candidate Key:** A relational schema sometimes may have more than one key. In that case, each key is referred to as a candidate key. The designer may assign any of the candidate keys as the **primary key**. The other candidate keys are then referred to as **a secondary keys**.

- Prime and non-prime attributes: If an attribute is a member of a candidate key, it is referred to as a prime attribute, otherwise as a non-prime attribute.

6.4.2 FIRST NORMAL FORM

The first normal states that each attribute in a relation must have atomic values. For a relation to be in1NF, each tuple in that relation must have single values for each attribute. Thus, 1NF disallows multivalued and composite attributes.

Let's consider the relation shown in table 6.8(a). It is not in 1NF as E_PHONE_NO is a multivalued attribute. We can convert it to 1NF by distributing the multiple values of phone number across the rows and making E_ID and E_PHONE_NO a combined primary key as shown in table 6.8 (b). By definition, each relation in a relational model by default is in 1NF.

Table 6.8(a): Example of a relation that is not in 1NF

E_ID	E_NAME	E_PHONE_NO
1	Ravi Sharma	912346795
2	Erica Swift	8123456745
3	Rahul Nath	{6712387453, 6532478456}
4	Prabin Kumar	77345546734

Table 6.8 (b): Normalized version of the relation in table 6.8(a)

E_ID	E_PHONE_NO	E_NAME
1	912346795	Ravi Sharma
2	8123456745	Erica Swift
3	6712387453	Rahul Nath
3	6532478456	Rahul Nath
4	77345546734	Prabin Kumar

6.4.3 SECOND NORMAL FORM

The second normal form is based on full functional dependency. A relation is in second normal form it is already in 1NF and all the non-prime attributes in the relation are fully functionally dependent on the prime key. Let's consider the following relation in table 6.9(a)-

Table 6.9(a): Example of a relation that is not in 2NF

S_ID	COURSE_ID	S_NAME	COURSE_NAME	GRADE
1	1	Ravi	Java	A+
1	2	Ravi	Python	B
2	1	Rahul	Java	A
2	2	Rahul	Python	A+

The above table stores the grade scored by the students in different subjects. The primary key of the table is {S_ID, COURSE_ID}. The following are some of the functional dependencies that hold in the above relation-

i. {S_ID,COURSE_ID}→GRADE
ii. {S_ID}→S_NAME
iii. {COURSE_ID}→COURSE_NAME

The FD (i) is full functional dependency as GRADE is dependent on the S_ID and the COURSE_ID. Neither S_ID nor COURSE_ID alone can determine the GRADE. However, as we can see that the FDs (ii) and (iii) are partial dependencies as S_NAME can be determined by S_ID alone. Similarly, COURSE_NAME can be uniquely identified by COURSE_ID only. Due to these partial dependencies, the relation is not in 2NF.

A relation that is not in 2NF, can be converted to 2NF by breaking it into multiple relations where the nonprime attributes are fully dependent on the primary key. For example, the relation in table 6.9 (a) can be normalized to 2NF by breaking it down into Grade, Student and Course tables as shown in tables 6.9(b)-6.9(d).

Table 6.9 (b): Grade table which is in 2NF

S_ID	COURSE_ID	GRADE
1	1	A+
1	2	B
2	1	A
2	2	A+

Table 6.9(c): Student table(2NF)

S_ID	S_NAME
1	Ravi
2	Rahul

Table 6.9(d): Course table (2NF)

COURSE_ID	COURSE_NAME
1	Java
2	Python

6.4.4 THIRD NORMAL FORM

The third normal form is based on transitive dependency. To be in 3NF, a relation must also be in 2NF, and no non-prime attribute should be transitively dependent on the primary key. An attribute, Z, in a relation is transitively dependent on the primary key X, if the functional dependencies X→Y and Y→Z hold, where Y is neither a candidate key nor a subset of any key in that relation.
Consider the relation in table 6.10(a) that stores the information of the students and the corresponding programs they are enrolled in. The primary key of the table is S_ID.

Table 6.10(a): Example of a relation violating 3NF

S_ID	S_NAME	PROGRAM_NAME	PROGRAM_DURATION
1	Ravi	B. Tech	4
2	Rahul	BCA	3
3	Arati	BCOM	3
4	Arif	MCA	2

Some of the functional dependencies that hold in the relation are-

i. S_ID→S_NAME
ii. S_ID→PROGRAM_NAME
iii. PROGRAM_NAME→PROGRAM_DURATION
iv. S_ID→PROGRAM_DURATION

We can see that the FD (iv) is a transitive dependency that can be inferred from FDs (i) and (ii) using the IR3. However, PROGRAM_NAME is not a candidate key in this relation neither it is a subset of any key. Thus, it can be concluded that the relation is not in 3NF as PROGRAM_NAME is transitively dependent on the primary key S_ID via the non-prime attribute PROGRAM_NAME. We can covert it to 3NF by breaking it into two relations as shown in table 6.10(b) and 6.10(c).

Table 6.10(b): Student relation which is in 3NF

S_ID	S_NAME	PROGRAM_NAME
1	Ravi	B. Tech
2	Rahul	BCA
3	Arati	BCOM
4	Arif	MCA

Table 6.10(c): Program relation which is in 3NF

PROGRAM_NAME	PROGRAM_DURATION
B. Tech	4
BCA	3
BCOM	3
MCA	2

6.4.5 BOYCE CODE NORMAL FORM (BCNF)

The Boyce code normal form is a stricter version of 3NF. Originally, it was proposed to simplify the definition of 3NF, however ended up putting more constraints on the relation. Every relation to be in BCNF must be in 3NF.

The general definition of 3NF states that- whenever a non-trivial functional dependency X→Y holds then it must satisfy either of the following conditions-

i. X is a super key in the relation.
ii. Y is a prime attribute.

In BCNF the condition (ii) is eliminated. This implies that a relation is in BCNF if, for each functional dependency X→Y, X is a super key of R.

Example: To better understand the concept let's consider the relation in table 6.11(a).

Table 6.11(a): Example of a relation which is not in BCNF

S_ID	COURSE	INSTRUCTOR
101	Java	Rupam
101	DBMS	Priya
103	DBMS	Trisha
104	Python	Ashmita
105	Java	Kalyan

The constraints in the above table are-

- A student can enrol in multiple courses
- For each course, an instructor is assigned to the student.
- A course can be thought by multiple instructors.

The primary key in this relation is {S_ID, COURSE}, as it uniquely determines the INSTRUCTOR. Another point to be noted is that the course is dependent on the instructor as one instructor can teach only one subject. Thus the functional dependencies are-

i. {S_ID, COURSE} →INSTRUCTOR
ii. INSTRUCTOR → COURSE

The table is in 3NF as there are both the FDs either the right-hand side is a key or the left-hand side is a prime attribute. However, it is not in BCNF as in the FD (ii) INSTRUCTOR is not a prime attribute. To covert the relation to BCNF, we may decompose the relation into two other relations as shown in table 6.11(b) and 6.11(c).

Table 6.11(b): Student_instructor table

S_ID	INSTRUCTOR
101	Rupam
101	Priya
103	Trisha
104	Ashmita
105	Kalyan

Table 6.11(c): Instructor_course table

Table 6.11(c): Instructor_course table

INSTRUCTOR	COURSE
Rupam	Java
Priya	DBMS
Trisha	DBMS
Ashmita	Python
Kalyan	Java

Consider another example. The relation in table 6.11(d) stores employee details- id, name, pan number, and age. The candidate keys for this relation are- E_ID and PAN_NO. From these E_ID has been chosen as the primary key. Following are some of the dependencies that exist in the relation-

iii. E_ID→ E_NAME
iv. E_ID→PAN_NO
v. PAN_NO→ AGE
vi. PAN_NO→E_ID

As we can see that in all the FDs the left-hand side is a candidate key, so the relation is in BCNF.

Table 6.11(d): Example of a relation which is in BCNF

E_ID	E_NAME	PAN_NO	AGE
1	Xavier Mavely	ABCFX985B	48
2	Manish Pandey	PQRET654X	34
3	Rakesh Sharma	GFRTE4563B	54
4	Shilpi Molohtra	FDREU004T	43

SAQ

1. Define normalization.
2. Define the terms- Superkey, key, candidate key and primary key.

Check Your Progress

12. A legal relational schema by default is in _____ normal form.
13. If a relation is in 2NF, then non no-prime attribute can be _____ dependent on the key.
14. For a relation R to be in BCNF, if the functional dependency A → B holds in R then A must be a _____.
15. In a relation, which is in 3NF, no non-prime attribute is _____ dependent on the primary key.
16. State true or false
    a. BCNF is stricter than 3NF.
    b. Normalization is a tool to minimize NULL values.

6.5 MULTIVALUED DEPENDENCY AND FOURTH NORMAL FORM

In section 6.3.1.3, we introduced the concept of multivalued dependency. In this section, we will present an elaborate discussion on multivalued attributes and the fourth normal form (4NF).

6.5.1 FORMAL DEFINITION OF MULTIVALUED DEPENDENCY

Lets X and Y be two attributes in a legal relation and let t1 and t2 be any two legal tuples in that relation such that-

t1(X)=t2(X) .

The multivalued dependency X→→ Y holds in the relation, if there exists another two tuples t3 and t4 with the following conditions-

t1(X)=t2(X)=t3(X)=t4(X)

t1(Y)=t3(Y)

t2(Y)=t4(Y)

Example: Consider the relation in table 6.12. We may observe that the students Ravi and Rahul have interests in multiple indoor and outdoor games. In the first four tuples, S_Name is the same, i.e, Ravi. As the tuple (Ravi, Badminton, Cricket) and (Ravi, Table Tennis, Football) exist in the relation, another two tuples (Ravi, Badminton, Football) and (Ravi, Table Tennis, Cricket) also exist in the same relation. Sam can be observed from the last four tuples as well. Thus, we can say that-

S_Name→→ IndoorGame

S_Name→→ OutdoorGame

Table 6.12: Example of multivalued dependency

S_Name	Indoor Game	Outdoor Game
Ravi	Badminton	Cricket
Ravi	Table Tennis	Cricket
Ravi	Badminton	Football
Ravi	Table Tennis	Football
Rahul	Chess	Cricket
Rahul	Volleyball	Football
Rahul	Chess	Football
Rahul	Volleyball	Cricket

6.5.2 FOURTH NORMAL FORM

The definition of the fourth normal form is based on multivalued dependency. For a relation to be in 4NF, it must be in BCNF and should not have any multivalued dependency. The relation in table

6.12 is in BCNF but not in 4NF as it contains multivalued dependencies. To transform the said relation into 4NF, we may decompose it into tables 6.13(a) and 6.13(b)-

Table 6.13(a): Student_Indoor Games relation

S_Name	IndoorGame
Ravi	Badminton
Ravi	Table Tennis
Rahul	Chess
Rahul	Volleyball

Table 6.13(b): Student_Outdoor Games relation

S_Name	OutdoorGame
Ravi	Cricket
Rahul	Cricket
Ravi	Football
Rahul	Football

Both the relations in table 6.13(a) and 6.13(b) are in 4 NF as there is no multivalued dependency.

6.6 RELATIONAL DECOMPOSITION AND ITS PROPERTIES

In the earlier sections, we have discussed the normal forms- 1NF, 2NF, 3NF, BCNF, and 4NF. In all the cases we have seen a relation can be upgraded to a higher normal form by decomposing it into multiple relations. Decomposition helps in removing redundancies and inconsistencies.

While decomposing a relational schema into multiple relational schemas one must make sure that the decomposition preserves all the original attributes. If a relational schema, R, is decomposed into multiple relations D={R1, R2, R3,……, Rn}, then each attribute in R must appear in at least one relation Ri in D. This property is called the ***attribute preserving*** property of decomposition. Another additional aim of decomposition is that each relation Ri in D must be at least in either 3NF or BCNF. Unfortunately, these two properties alone don't guarantee a good database design. In the following sections, we discuss some additional criteria that must hold in a decomposition.

6.6.1 DEPENDENCY PRESERVATION PROPERTY of a DECOMPOSITION

Let's consider the decomposition of the relational schema R into D as discussed above. The dependency preserving property of a decomposition states that- every functional dependency X→Y specified in R, must appear either directly in one of the relations Ri ∈ D or can be inferred from other dependencies specified in some relation Ri ∈ D. Let F be the set of FDs specified in R. Let F1, F2… Fn is the set of functional dependencies specified in R1, R2…. Ri respectively. The decomposition D is said to be dependency preserving if-

$(F1 \cup F2 \ldots \cup Fn)^+ = F^+$

Example: Let's consider a relation R (W, X, Y, Z). The specified set of functional dependencies for this relation is F= {WX → Y, Y → Z, Z → W}. R is decomposed into two relations - R1(W, X, Y) and R2(Y, Z). Let F1 and F2 be the set of functional dependencies for R1 and R2 respectively.First, we will find the closure of F1. To do so, we will consider the combinations- W, X, Y, WX, XY, and XY.

W^+ = { W } // Trivial

X^+ = { X } // Trivial

Y^+= {Y, W, Z} = {Y, W} [As Z is not in R1, it has been removed from the closure]

Y→ W [Removing Y from right side as it is trivial attribute]

WX^+ = {W, X, Y, Z} [As Z is not in R1, it has been removed from the closure]
= {W, X, Y}

WX → Y [Removing WX from right-side as these are trivial attributes]

XY^+ = {X, Y, Z, W}
= {W, X, Y}

XY → W [Removing XY from right side as these are trivial attributes]

WY^+ = {W, Y, Z}

WY → Z [Removing WY from right side as these are trivial attributes]

Thus, F1 ={Y→ W, WX → Y, XY → W}.

Similarly, F2= { Y→ Z }

In the original relation R, F={ WX→ Y, Y → Z, Z → W}.

WX → Y is present in F1.

Y → Z is present in F2.

But, Z → W is not preserved.

F1 U F2 is a subset of F. So, the given decomposition is not dependency preserving.

6.6.2 LOSSLESS (NON-ADDITIVE) JOIN PROPERTY of a DECOMPOSITION

Another important property that a decomposition should satisfy is the lossless or non-additive join property. This ensures that if we reconstruct the relation R by performing natural join (*) on R1, R2 … and Rn, then it should not produce any spurious tuples. To put it in another way, the decomposition is -

- lossy if R1*R2*…..*Rn ⊃R
- Lossless if R1*R2*…..*Rn = R

The problem of spurious tuples has already been discussed in section 6.2.4.

To illustrate this concept letsconsider the following relational schema in table 6.14(a)-

Table 6.14(a): Employee_Department relation

E_ID	E_NAME	E_AGE	DEPT_ID	DEPT_NAME
1	Xavier	42	1	Sales
2	Pallavi	34	1	Sales
3	Arun	42	2	Marketing
4	Susane	28	3	HR

If we decompose this relation into two smaller schemas R1(E_ID, E_NAME, E_AGE, DEPT_ID) and R2(DEPT_ID, D_NAME), then following tables 6.14(b) and 6.14(c) will be the result of the decomposition-

Table 6.14(b): Decomposition of Employee_Department relation to relation R1

E_ID	E_NAME	E_AGE	DEPT_ID
1	Xavier	42	1
2	Pallavi	34	1
3	Arun	45	2
4	Susane	28	3

Table 6.14(c): Decomposition of Employee_Department relation to relation R2

DEPT_ID	DEPT_NAME
1	Sales
2	Marketing
3	HR

If we perform natural join over R1 and R2 then we will get back exactly the tuples present in the relation R. Neither will any extra tuple be generated nor will there be any missing tuple. Thus this decomposition is lossless.

However, if the same relation R is decomposed into two other relations- R3(E_ID, E_NAME, E_AGE) and R4(DEPT_ID, D_NAME, E_AGE), then this would result in the following tables 6.14(d) and 6.14(e)-

Table 6.14(d): Decomposition of Employee_Department relation to relation R3

E_ID	E_NAME	E_AGE
1	Xavier	42
2	Pallavi	34
3	Arun	42
4	Susane	28

Table 6.14(e): Decomposition of Employee_Department relation to relation R4

DEPT_ID	E_AGE	DEPT_NAME
1	42	Sales
1	34	Sales
2	42	Marketing
3	28	HR

The natural join of R3 and R4 (R3*R4) would result in table 6.14(f)-

Table 6.14(f): Natural join of R3 and R4

E_ID	E_NAME	E_AGE	DEPT_ID	DEPT_NAME
1	Xavier	42	1	Sales
1	Xavier	42	2	Marketing
2	Pallavi	34	1	Sales
3	Arun	42	2	Marketing
4	Susane	28	3	HR

As we can see that the natural join of R3 and R4 results in extra information that does not exist in

Check Your Progress

17. For multivalued dependency to occur in a relation, it must have at least ____ attributes.
18. A decomposition is loss-less if natural join of the relations in the decomposition does not produce any _____ tuple.
19. 4NF is based on _____ dependency.
20. _____ states that each attribute of the original relation must appear in at least one of the relation in its's decomposition.

the original relation R. Thus, the decomposition of R into R3 and R4 is a lossy join.

6.7 ALGORITHMS FOR RELATIONAL DATABASE SCHEMA

In this section, we present some algorithms related to the decomposition of a relational schema.

6.7.1 Relational Synthesis

Algorithm 6.2: Relational Synthesis into 3NF with Dependency Preservation
Input: A universal relation R with a set of a functional dependency F
Step 1: Find the minimal cover G of F.
Step 2: For each left-hand side of X of a functional dependency in G, construct a relational schema with attributes {X∪A1∪A2….∪Ak} with X as key, where X→A1, X→A2… X→Ak.
Step3: Place the remaining attributes (which could not be placed in any relation in step2) in single relation.
Claim: All the relational schemas created by algorithm 6.2 are in 3NF.

6.7.2 Testing lossless join property

Algorithm 6.3: Testing for lossless or non-additive join property
Input:A universal relation R, R's decomposition D = {R1, R2, . . ., Rm}, and a set of functional dependencies F.
Step1: Create an initial matrix *S* with dimension 'm' rows and 'n' columns, where 'm' is the number of relations in D and 'n' is the number of attributes in R.
Step2: Set each *S(i,j)* in the matrix to b_{ij}, where b_{ij} is a distinct symbol associated with S(i,j).
Step 3: For each row *i* in *S*:

For each column *j* in *S*:

If Ri contains attribute a_j then

set *S(i,j)*=a_j

Step 4: Repeat the following loop until S remains unchanged after a complete loop execution:

For each *A→B* in F:

For each row *i* in S, having the same symbol in the column corresponding to attribute for A:

Set the symbols in each column corresponding the attribute B to be the same. If there exists an 'a' symbol for any of these columns, set all other columns to symbols 'a', else chose any of the 'b' symbols that appear for any of these columns and update the symbols in the rest of the columns in all such rows to 'b'.

Step5: The decomposition has lossless join property only if there exists a row in S that contains only 'a' symbol. Otherwise, the decomposition is lossy.

Example: Lets consider a relation R = {E_ID, E_NAME, P_NO, P_NAME, P_LOC, HOURS} with the following set of functional dependency-

F = {E_ID → E_NAME, P_NO → {P_NAME, P_LOC}, {E_ID, P_NO} → HOURS}

Let D = {R1, R2, R3} be the decomposition the relation, where,

R1 = {E_ID, E_NAME}

R2 = {P_NO, P_NAME, P_LOC}

R3 = {E_ID, P_NO, HOURS}

Application of steps 1,2 and 3 results in the matrix in table 6.15(a)-

Table 6.15(a): Example for testing loss-less join property

	E_ID	E_NAME	P_NO	P_NAME	P_LOC	HOURS
R1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

Now, for the functional dependency E_ID → E_NAME, the E_ID attribute in R1 and R3 have the same symbol. So, the symbol for the E_NAME attribute in R3 will be updated to a_2 as R1 has a_2 for E_NAME. This results in the matrix in table 6.15(b).

Table 6.15(b): Example for testing loss-less join property

	E_ID	E_NAME	P_NO	P_NAME	P_LOC	HOURS
R1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R3	a_1	a_2	a_3	b_{34}	b_{35}	a_6

Similarly, for the functional dependency P_NO → {P_NAME, P_LOC}, in the values for attributes, P_NAME and P_LOC are updated to a_4 and a_5 respectively as R2 and R3 have the same symbol for P_NO. Thus the updated matrix will be as shown in table 6.15(c)-

Table 6.15(c): Example for testing loss-less join property

	E_ID	E_NAME	P_NO	P_NAME	P_LOC	HOURS
R1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R3	a_1	a_2	a_3	a_4	a_5	a_6

We can see observe from the above matrix that the row R3 has all 'a' symbols. Thus, the decomposition D has lossless join property.

6.7.3 TESTING LOSSLESS JOIN PROPERTY IN BINARY DECOMPOSITION (Property LJ1)

Breaking down a relation into two relations is called binary decomposition. The following property helps to test for lossless join property in binary decomposition-

- ***Property LJ1:*** The binary decomposition D = {R1, R2} of a relation R has the lossless join property with respect to a set of functional dependencies F on R *if and only if* either
  - $((R1 \cap R2) \rightarrow (R1 - R2))$ is in F^+, or
  - $((R1 \cap R2) \rightarrow (R2 - R1))$ is in F^+.

6.7.4 SuCCESSIVE LOSSLESS JOIN DECOMPOSITION (PROPERTY LJ2)

Let a decomposition D={R1, R2…. Rm} of a relation R, concerning a set of functional dependency F, has lossless join property. Now, let's divide a relation Ri in D to smaller relations Q={Q1, Q2…Qp} in such a way that Q also has lossless join property for F. If we now replace Ri by Q in D, then ***property LJ2*** states that a set of decomposition D1={R1, R2,…Ri-1,Q1,Q2,…Qp, …, Rm}) will also have lossless join property.

6.7.5 Non-additive Join Decomposition into BCNF Schemas

Algorithm 6.4: Relational decomposition into BCNF relations with lossless join property

Input: A universal relation R with F as set of specified functional dependencies .

Step1: Set D={R}

Step2: For each relation Q in D, which is not in BCNF:

Identify the functional dependency $A \rightarrow B$, that violates the BCNF.

Replace Q in D by two relations $(Q - B)$ and $(A \cup B)$

Step3: Stop

Explanation: Since $(Q - B) \cap (A \cup B) \rightarrow (A \cup B) - (Q - B)$ is equivalent to $A \rightarrow B \in F^+$. By virtue of property LJ1, the decomposition is lossless.

Example:R = {X, Y, Z} F = {XY → Z, Z → Y}

Let D= {{X, Y, Z}};
{X, Y, Z} in D is not in BCNF due to the functional dependency Z→ Y.
Thus, decompose {X, Y, Z} to $(\{X, Y, Z\} - Y)$ and $(X \cup Y)$, I.e. to {X, Z} and {X,Y}.
Replace {X, Y, Z} in D by {X, Z} and {X,Y}.
D={{X, Z},{X,Y}}
{X, Z} and {X,Y} both are now in BCNF.

6.7.6 Relational synthesis algorithm into 3NF with dependency preservation and lossless join property

Algorithm 6.5: Relational synthesis algorithm into 3NF with dependency preservation and lossless join property
Input: A universal relation R and a set of FDs F
Step 1: Compute a minimal cover G for F
Step 2: Construct a relational schema in D with attributes {X∪A1∪A2….∪Ak} with X as key, for each left-hand side of X of a functional dependency in G. The functional dependencies: X→A1, X→A2… X→Ak, should be the only dependencies in G with X on the left-hand side.
Step 3: If there is no relation in D that contains a key of R, then create one with the attributes of a key in R.
Example: R = {A, B, C, D, E, H} is a relation with functional dependencies F = {AE → BC, B → AD, CD → E, E → CD, A → E}.
Step 1: The minimal cover of F is G = {A → B, A → E, B → A, CD → E, E → CD} [Derived using the algorithm 6.1]
Step 2: Based on the functional dependencies in G the R will be decomposed into D={R1, R2, R3, R4, R5}, with the set of functional dependencies F1, F2, F3, F4 and F5 respectively, where

- R1 = {A, B, E} with F1 = {A → B, A → E}
- R2 = {B, A} with F2 = {B → A}
- R3 = {C, D, E} with F3 = {CD → E}
- R4 = {E, C, D} with F4 = {E → CD}

Combine R3 and R4 into one relation schema R5 = {C, D, E} and F5 = {CD → E, E → CD}. So, D={R1, R2, R5}
Step 3: In R, AH, and BH are candidate keys. As we can see that neither of these appears as key in any of the relations in D. So, we create another relational schema R6 = {A, H} and F = {}
Now, all the relations in D = {R1, R2, R5, R6} are in 3NF.

6.7.7 Finding a key *K* for relation schema *R* based on a set *F* of functional dependencies

Algorithm 6.6: Finding a Key for a rational schema based on a set of functional dependencies.
Input: A relational schema R(A1, A2,…, Am)

Step 1: Set the key K={A1, A2,…, Am}

Step 2:For each attribute Ai in K:

Determine $(K\text{-}Ai)^+$ with respect to F. If $(K\text{-}Ai)^+$ contains all the attributes in R, then set K=K-{Ai}

Example: Lets consider the relation R={A, B, C, D} with F={A → BCD, C → A}

- A^+=ABCD
- B^+=B
- C^+=ABCD
- D^+=D

So, the candidate keys are A and C as the closure of A and C contains all the attributes of R.

6.7.8 Relational decomposition into 4NF relations with lossless join property

Whenever a relational schema R is decomposed into D={R_1, R_2} based on multivalued dependency A→→B that holds in R, then property LJ1' presents the necessary and sufficient condition to check whether the decomposition is lossless or not.

- **PROPERTY LJ1'**
  - The decomposition D={R_1, R_2} of R, is a lossless (non-additive) join decomposition with respect to a set F of functional *and* multivalued dependencies if and only if
    - $(R_1 \cap R_2) \rightarrow\rightarrow (R_1 - R_2)$
  - or by symmetry, if and only if
    - $(R_1 \cap R_2) \rightarrow\rightarrow (R_2 - R_1)$).

Algorithm 6.7: Decomposition of a relation into 4NF relations with lossless join property

Input:A universal relation R and a set of functional and multivalued dependencies F.

Step 1: Set D := { R };

Step 2:While there exists a relation *Ri* in *D* that is not in 4NF do {

choose a relation schema *Ri* in *D* that is not in 4NF;

find a nontrivial MVD A →→ B in Ri that violates 4NF;

replace *Ri* in *D* by two relation schemas (*Ri* - *B*) and (*A* υ *B*);

SAQ

1. State the properties LJ1 and LJ2.
2. How we can test whether a binary decomposition is lossless or lossy?

};

6.8 SUMMING UP

In this module, we have discussed the goals of good relational schema and also some informal guidelines to achieve a good design. A relational schema having clear semantic can easily be interpreted by the users and thus is considered as a good design practice. Minimizing redundancy, null values, and generation of spurious tuples are some other criteria for a good design. Normalization based on primary keys and functional dependencies are used as tools to systematically evaluate the "goodness" of a relation and thus also help to convert a given relation to a better design i.e. to a

higher degree of normalization if possible. In this module we have discussed five normal forms- 1NF, 2NF, 3NF, BCNF, and 4 NF. Table 6.16 summarises the normal forms.

Table 6.16: Summary of normal forms

Normal form	Test
1NF	Disallows multiple values for the attributes in a tuple.
2NF	Disallows partial dependency of the non-key attributes on key attributes
3NF	Disallows transitive dependency of a non-key attribute on the key attribute via another non-key attribute.
BCNF	Whenever the dependency A→B holds in a relation, A must be the super key
4NF	Disallows multivalued dependency

Though normalization is a strong tool to analyze the goodness of a relation, it alone does not guarantee a good design. In this module, we have also presented two more criteria namely- attribute preservation and loss-less join property which must also be satisfied by the relations in a good design. We have also discussed some algorithms for ensuring lossless join property in a decomposition.

6.9 ANSWERS TO CHECK YOUR PROGRESS

1. Insertion, deletion and modification
2. Spurious
3.
   3.a.False
   3.b.True
4. Trivial
5. Transitivity
6. A→BC
7.
   7.a. False
   7.b.True
   7.c.True
8. Equivalent
9. G^+
10. {A, B, C, D}
11.
    11.a.True
    11.b.False
12. 1NF
13. Partially
14. Superkey

15. Transitively
16.
    16.a. True
    16.b.False
17. 3
18. Spurious
19. Multivalued
20. Attribute preservation propert

6.10 QUESTIONS AND ANSWERS

Multiple Choice Question

1. A relation R(Id, Name, Age) is decomposed into R1(Id, Name) and R2(Name, Age). The decomposition is-
    a. Loss-less
    b. Lossy
    c. Can't say anything
    d. None of the above
2. If the functional dependency A → B and B → CD hold in a relation then which of the following FD cab be inferred?
    a. A → CD
    b. CD → A
    c. C → A
    d. D →A
3. For transitive dependency to hold, a relation must have at least _____ attributes.
    a. 2
    b. 3
    c. 4
    d. 5
4. A relation is in 3NF, if for each functional dependency A → B that holds in a relation then which of the following is true?
    a. A is a super-key of the relation.
    b. B is a prime attribute
    c. Either (a) or (b)
    d. None of the above.
5. A relation, R(A, B, C) has the four tuples t1(1, 2, 3), t2(1, 2, 4), t3(1, 1, 3) and t4 (1, 1, 4). Which of the following is a valid functional dependency for the above relation?
    a. A → B
    b. B → C
    c. A → C
    d. A →→ C
6. The binary decomposition D = {R1, R2} of a relation R has the lossless join property with respect to a set of functional dependencies F on R *if and only if*-
    a. ((R1 ∩ R2) → (R1- R2)) is in F^+

b. $((R1 \cap R2) \rightarrow (R2 - R1))$ is in F^+
c. Either (a) or (b) is true
d. Both (a) and (b) is true

7. A relation is R is decomposed into two relations R1 and R2. This decomposition results in spurious tuples if-
   a. R1*R2 is the subset of R
   b. R1*R2 is equal to R
   c. R1*R2 is the superset of R
   d. None of the above.
8. The set of functional dependencies $F=\{A \rightarrow B,\ B \rightarrow C\}$ and $G=\{A \rightarrow B,\ A \rightarrow C\}$ specified for a relation R(A, B, C) are-
   a. Equivalent
   b. Trivial
   c. Limited
   d. None of the above

Answers: 1.b, 2.a, 3.b, 4.c, 5.d, 6.c, 7.c, 8.a

Match the Columns:

Column A	*Column B*
1. Functional dependency	1. Relationship among attributes
2. 1NF	2. Doesn't allow partial dependency of non-prime attributes on key.
3. 2NF	3. Based on multivalued attribute
4. 3NF	4. Disallows composite and multivalued attribute
5. 4NF	5. Based on transitive dependency.

Answers: 1.1, 2.4, 3.2, 4.5, 5.3

State true or false:

1. A set of functions dependency F covers G if F^+ is a subset of G^+.
2. An attribute is a prime attribute if it is part of a key.
3. Normalization alone is sufficient to guarantee a good relational schema design.
4. Composite attributes are allowed in 2NF but not in 1NF.
5. It is always possible to create a decompose a relation into smaller relation where each smaller relation is in 3NF and follows the dependency preservation rule.

Answers: 1.False, 2. True, 3. False, 4. False, 5. True

Short Answer type Questions:

1. Define closure of a set of functional dependencies.

2. Write down the properties LJ1 and LJ2.
3. State when two sets of functional dependencies are considered to be equivalent.
4. What are spurious tuples? Why are they considered as bad?
5. What are the problems with null values in a relation?
6. State the condition a relation must satisfy to be in 2NF.
7. List the conditions a binary decomposition must satisfy to be loss-less.
8. Write the condition for a relation to be in BCNF.

Long Answer type Questions:

1. Discuss the four informal guidelines for designing a good relation.
2. Discuss the problem of update anomalies in a relation with appropriate examples.
3. What is normalization? Why is it important? Discuss, with an example, how a relation which is not in 1NF can be converted to 1NF.
4. Define functional dependency. Briefly discuss the types of functional dependencies. Give one example each.
5. Write down the Armstrong's axioms for functional dependency. Why are these rules important?
6. Discuss the 1st, 2nd and 3rd normal forms with suitable examples.
7. What is multivalued functional dependency? Discuss the fourth normal forms with an example.
8. Discus attribute preservation and dependency preservation properties of a relational decomposition. Write the algorithm to decompose a relation to smaller relations where each smaller relation is in 3NF and dependency is also preserved.
9. What is loss-less join property of a relation? Write an algorithm to test the loss-less join property of a decomposition.

6.11 SUGGESTED READING

- Ramez, Elmasri. *Fundamentals of Database Systems*. Pearson Education India, 2020.
- Silberschatz, Abraham, Henry F. Korth, and Shashank Sudarshan. *Database system concepts*. McGraw-Hill, 1997.

UNIT 1 QUERY PROCESSING AND OPTIMIZATION

CONTENTS

1.0. INTRODUCTION

In this unit, you will get to learn in detail about query processing and query optimization and how their implementations are done in Database Management System(DBMS).You already know that databases are created to organize and store all the related data and information at a particular place so that the users can access and manipulate it as per requirement. It may so happen that different users may use their languages to access those data, which at times becomes difficult for the DBMS system to return the actual information, and also the data may not be accurate. So there comes the need for a common operation or you can say a language that can communicate between the system and the user.

Some of the terms you will learn here are

- **SCANNER** – Scanner helps in identifying the tokenized words present in the SQL query such as the keywords, relation names, attributes, etc.
- **PARSER** – Parser checks the syntax of the SQL query which is termed parsing. It also validates whether any semantic error is present or not.
- **PARSE / QUERY TREE** – The relational algebra is internally converted into a tree structure called the Parse tree.
- **QUERY BLOCKS** – SQL queries are first converted into some basic blocks containing algebraic operators known as query blocks after which the queries are optimized.
- **OPTIMIZATION** – Choosing the most effective minimal cost of execution is termed as optimization performed by an optimizer.

You will also learn about different execution plans which are cost-effective and generate a code optimizer.

1.1. OBJECTIVES

Studying this unit, you will be able to:

- *Understand* the concept of query processing
- *Understand* the concept of parsing, translation
- *Understand* the concept of optimization
- *Know* how to choose the evaluation plan

1.2. QUERY PROCESSING

To bridge the gap between the DBMS and the Users, a standard language, which is understandable by the DBMS is used between the two, so that correct data are retrieved on

processing. This standard language is known as the Structured Query Language (SQL) which is a High-Level Language (HLL). DBMS automatically converts this HLL to Low-Level language (LLL) which is a machine-understandable language using Relational Algebra whenever a query is encountered by the system [1].

A query is submitted to the database to retrieve relevant information from it. Whenever any Query is encountered by the system, verification of the query is done by the DBMS and that query is converted to some Low-Level Language after which path of the execution is selected and the queried data gets retrieved from the storage memory. It is done internally by the DBMS. This whole technique is performed by the Query Processor which is a feature present in the DBMS and the functions that are performed are termed as the Query Processing.

As discussed, DBMS consists of a Query Processor that checks and verifies the queries given by the Users which are in the form of SQL commands (HLL). It then translates these commands into Low-Level Language that is understandable to the DBMS system so that the system can work on it. It converts the HLL Step-by-Step into LLL and returns the value which the users want. In query processing, sorting is a very important step. For performing the operations like ORDER-BY, JOIN, PROJECT, SELECT, UNION, INTERSECTION, etc. sorting, or merge-sort algorithm is a very important phase. A term called 'external sorting' is applied when the memory is small and records of large files need to be stored. It consists of two phases – the sorting phase and the merging phase.

In the case of sorting phase, the files are sorted very precisely so that it easily fits into the available memory. The number of executions is specified to its initial and dedicated file blocks and also on the amount of buffer space.

In the case of the merging phase, the sorted executed query is merged at different passes. The amount of merge that can be occurred together is termed the degree of a merge. For each pass at least one buffer block is required for holding the executed query.

The processing of a query is performed in the following given steps.

a) Parsing and Translation of the Query.
b) Optimization of the parsed query.
c) Compilation or Interpretation in the Query Code Generator.
d) Execution into the Evaluation engine.

1.2.1. The Block Diagram to Show the Steps of Conversion in Query Processing

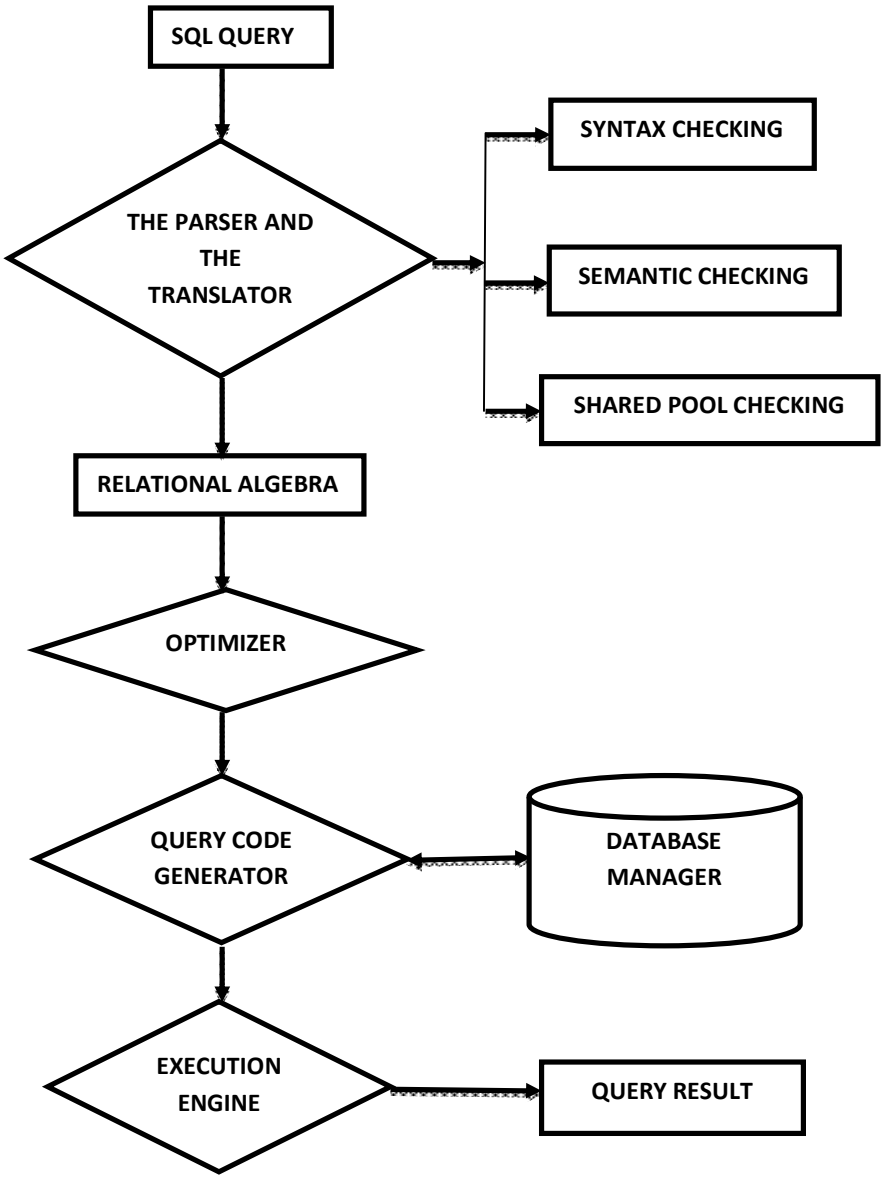


Figure 1: BLOCK DIAGRAM FOR QUERY PROCESSING

STEP 1.PARSING –

An SQL query is communicated to the DBMS system through an application program. The high-level language is translated into its low-level language. That is, the SQL query is converted into its relational algebra.

Query blocks are formed after the division of the original SQL query, and hence it becomes ready for its optimization. Query blocks can contain a single expression of SELECT, WHERE, FROM, HAVING, GROUP BY. When the nested queries are encountered having aggregate operators like COUNT, SUM, MIN, MAX, then a separate query block is formed.

For example, the following query is encountered by the system where the teacher wants to fetch the name of all the students where age must be less than 20 years.

select stud_name from Student where age<20;

This query is converted into its relational algebra as given below.

$$\pi_{age}\ (\sigma_{age}<20\ (Student))$$

On converting the query into its relational algebra internally, other steps of parsing are done. When this query is encountered with the system, scanning of the same is done to check whether any syntax error is present in the query or not.

- After proper validation, this relational algebra is then converted into tokenized form (from the example above 'select', 'stud_name', 'from', 'student', 'where', 'age<20' are different tokens) represented as a Parse tree.
- After that, the Parser performs some checks such as Syntax, Semantic, and Shared Poolchecks as shown in the diagram. In the Syntax checking the parser checks whether the syntax of the query is correct or not. An example is shown below.

select stud_name formStudent throughage<20;

It can be seen that '**form**' and '**through**' are the words respectively spelled incorrectly and wrongly written. These are the syntax errors.

- The semantic check helps in checking whether the table, keywords, columns present in the query are also present in the Database or not. If it is present, then it proceeds to the next step and if it is not present then it returns an error to the User.
- During its execution, every query is given a hash code by the parser and if this code is present then no extra performances are done in the rest of the steps. This hash code is checked by the Shared Pool [2].

STEP 2.OPTIMIZATION –

After the query goes through different processes in the Parsing phase, the parsed query is then shifted to the Optimization phase so that a minimal cost of execution is selected. Minimal evaluation cost is the time when a query is encountered in the system till it returns the result. For selecting the best minimal plan for executing the query, a Catalog Manager, present in the

Optimizer helps in the selection of the minimal cost. A hard parsing must be performed for at least a single DML statement and after that, the process of optimization is carried on. Access routines help in the implementation of query operations such as SELECT, JOIN, PROJECT, etc.

Typically, optimization takes any one of the following forms: The heuristic form of optimization or Cost based form of optimization.

(a) Heuristic optimization – With the help of heuristic rules, refinement of query execution is done so that individual operations are reordered.
(b) Cost-based optimization – with the help of cost-based rules, estimation of the cost of plans is done by the reduction of the overall cost of the query.

STEP 3. QUERY CODE GENERATION –

Executions plans are none other than the systematic way of ordering the access routines. Once the execution plans are selected and determined by the optimization process, it is the work of the code generator to determine the actual access routines needed to be executed. The query code is then compiled or interpreted and it is then transferred to the database. The database processor then helps the query for its execution. The execution plans are stored in the database.

STEP 4. EXECUTION –

After the query has gone through the processes of Parsing, Optimization, and Query Code Generation, it is passed into the execution phase. The results of the query are executed and then it is returned to the Users along with its runtime errors.

CHECK YOUR PROGRESS

1) Define the terms

(a) Optimization, (b) Query blocks

2) What is query processing?

3) Draw the block diagram to show the steps of conversion in Query Processing.

4) Fill in the Blanks:

(a) The SQL query is converted into its _____.

(b) For selecting the best minimal plan for executing the query, a _____, present in the Optimizer helps in the selection of the minimal cost

1.2.2. EQUIVALENCE RULES

Following are the equivalence rules used during the time of processing and optimizations.

1)SIGMA-CASCADE - Intersection of $\theta 1$ and $\theta 2$ makes the system very much expensive therefore inner selection of $\theta 2$ and outer selection of $\theta 1$ is done to make the system effective.

$$\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$$

2) COMMUTATIVE SELECTION – Since σ is commutative therefore the practical implementation of it is necessary.

$$\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$$

3) PI- CASCADE – Combining all the projections into a single projection is a very good option.

$$\pi_{L1}(\pi_{L2}(...(\pi_{Ln}(E)).) = \pi_{L1}(E)$$

4) CARTESIAN PRODUCTS AS THETA-JOINS –

(a) EQUIVALENCE 1 – Only using the cross product makes the system very expensive, so a theta join is also used to make it effective.

$$\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$$

(b)EQUIVALENCE 2 – If both the thetas are joined, then it will require little execution support.

$\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$

5) THETA JOINS ARE COMMUTATIVE – Since theta joins are commutative, therefore the query processing depends upon the inner and outer joins.

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

6) NATURAL JOIN – Since joins are both commutative and associative, therefore the tables having lesser entries must be joined.

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

7) DISTRIBUTION OF SELECTION OPERATION –

(a) EQUIVALENCE 1 – Theta join is performed, after selection is applied. Then E_1 is joined with E_2.

$$\sigma_{\theta 1 \wedge \theta 2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 1}(E_1)) \bowtie_{\theta} (\sigma_{\theta 2}(E_2))$$

(b) EQUIVALENCE 2 – Here $\theta 1$ and $\theta 2$ contains the attribute of E_1 and E_2 respectively.

$$\sigma_{\theta 0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_{\theta} E_2$$

8) THETA-JOIN PROJECTION –

(a) EQUIVALENCE 1 – Here L_1 is projected over E_1 and L_2 is projected over E_2. It is compulsory for doing projections before joining.

$$\pi_{L1 \cup L2}(E_1 \bowtie_{\theta} E_2) = (\pi_{L1}(E_1)) \bowtie_{\theta} (\pi_{L1}(E_2))$$

(b) EQUIVALENCE 2 - Here L_1 is projected over E_1 and L_2 is projected over E_2. It is compulsory for doing projections before joining. E_3 is an equivalence relation that joins both the relation using L_3.

$$\pi_{L1 \cup L2}(E_1 \bowtie_{\theta} E_2) = \pi_{L1 \cup L2}((\pi_{L1 \cup L3})) \bowtie_{\theta} (\pi_{L2 \cup L4}(E_2))$$

9) UNION AND INTERSECTION ARE COMMUTATIVE

$$E1 \cup E2 = E2 \cup E1$$
$$E1 \cap E2 = E2 \cap E1$$

10) UNION AND INTERSECTION ARE ASSOCIATIVE

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$
$$(E1 \cap E2) \cap E3 = E1 \cap (E2 \cap E3)$$

11) SELECTION OPERATION OVER DIFFERENCE

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - \sigma_P(E_2)$$

12) PROJECTION OVER UNION

$$\pi_L(E_1 \cup E_2) = (\pi_L(E_1)) \cup (\pi_L(E_2))$$

1.3. QUERY OPTIMIZATION

As discussed in the previous section, query optimization is performed by the DBMS for selecting the minimal cost of query execution, that is the time when a query is encountered in the system till it returns the result. It is an automated process. The optimizer selects for the query an efficient plan for its execution. This selection of the plan is performed by the catalog manager. Most of the structures of the query optimizers can be seen as the Left-deep Join orders. For pushing the selections and the projections through the query tree, a heuristic approach is used. This heuristic approach chose the best relation for joining the next relations. This left deep Join tree also helps in reducing the complexities caused by optimizations. Every input made on the right-hand side of the tree is a relation and not a join. In the figure given below, R1, R2, R3, R4 are the relations.

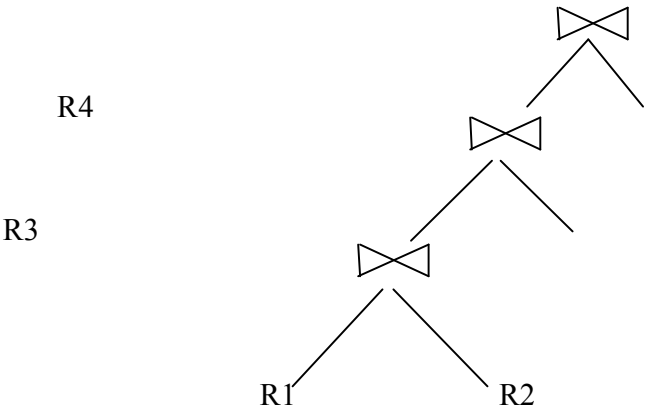


Figure 2: LEFT - DEEP JOIN ORDER

1.3.1. QUERY COST MEASUREMENT

Different DBMS structures balance the query plan and the choice of execution quality in many different possible ways. For these query plans and the choice of execution quality, the evaluation of the best Cost-based optimization is performed by the optimizer by choosing a minimal cost. There may be different types of cost efficiencies depending upon the need and situations, which may include

a) Minimization of the processing time
b) Minimization of the response time
c) Minimization of the input or output time
d) Minimization of the network time
e) Combination of the above situations etc.

Internally it can be seen that retrieving the desired data by the query, both from the primary and the secondary memories relatively takes an ample amount of time. This stipulated time is taken by different factors such as CPU time, disk I/O time, network access time, etc. It is seen that Disk I/O (finding the records by the processor from the secondary memory and returning the result) is the most efficient of all but it takes a much larger amount of time than the others. Mainly two factors are considered for the calculation of the Disk I/O access time. They are the Seek Time and the Transfer time. Let us understand this with the help of an example.

Suppose you need to find the **AGE** of the student whose name is '**Peter**' from the STUDENT table.

So the Disk I/O would work on the factors Seek and Transfer time as follows;

a) SEEK TIME – The amount of time required in finding a single record by the Processor in the disk memory is termed as the seek time. Seek time is usually represented by ***tS***. For the example given above, the time, from accessing the disk block in the memory till the search of the AGE of the student Peter is the Seek time of the Disk[1].

b) TRANSFER TIME –The amount of time required by the disk in returning the query result to the user or the processor is termed the transfer time. Transfer time is usually represented as ***tT***. For the example given above, the time of returning the value of the age of Peter to the User or the processor is known as the Transfer time of the Disk[1].

1.3.2. GENERATION OF EQUIVALENCE EXPRESSION

You are already familiar with the fact that relational algebras have equivalent expressions if they generate tuples of the same form, no matter what is the order of their tuples. In SQL statements,

input expression and output expression have tuples having multiset values and replace each other's form.

1.3.3. TRANSFORMATION EXAMPLES

- **PUSHING SELECTIONS**[3]

Pushing selection helps in the reduction of the Relation's size if it is performed in an early stage. An example of a query is given as "Fetch the Names of all the Employees in an IT department, along with the job profile of the Posts in which they work".

$\prod_{name,\ post}(\sigma_{dept\_name\ =\ "IT"}$ (employees $\bowtie$ (works $\bowtie$ $\prod_{job\_profile,\ post}$(profile))))

By transforming the above expression with the help of rule 7 (a) of the Equivalence Rule you will get the following expression

$\prod_{name,\ post}$ ($(\sigma_{dept\_name\ =\ "IT"}$ (employees)) $\bowtie$ (works $\bowtie$ $\prod_{job\_profile,\ post}$(profile)))

- **MULTIPLE TRANSFORMATION**[3]

An example of a query is given as "Fetch the Names of all the Employees in an IT department for the year 2020, along with the job profile of the Posts in which they worked".

$\prod_{name,\ post}$ ($\sigma_{dept\_name\ =\ "IT"\ \wedge\ year\ =\ 2020}$ (employees $\bowtie$
(works $\bowtie$ $\prod_{job\_profile,\ post}$(profile))))

By transforming the above expression with the help of rule 6 (a) of the Equivalence Rule you will get the following expression

$\prod_{name,\ post}$ ($\sigma_{dept\_name\ =\ "IT"\ \wedge\ year\ =\ 2020}$ ((employees $\bowtie$ works) $\bowtie$
$\prod_{job\_profile,\ post}$(profile)))

- **PUSHING PROJECTIONS**[3]

An example of a query is given as "Fetch the Names of all the Employees in an IT department, along with the job profile of the Posts in which they work".

$\prod_{name,\ post}$ ($\sigma_{dept\_name\ =\ "IT"}$ (employees) $\bowtie$ works) $\bowtie$ $\prod_{job\_profile,\ post}$(profile))))

By transforming the above expression with the help of rules 8 (a) and 8 (b) of the Equivalence Rule you will get the following expression

$$\prod_{name,\ post}(\prod_{name,\ job\_profile}(\sigma_{dept\_name\ =\ \text{“IT”}}(\text{employees}) \bowtie \text{works})) \bowtie \prod_{job\_profile,\ post}(\text{profile}))))$$

- **ORDERING OF JOIN**[3]

Join ordering minimizes the storage cost of the relation. For example, there are two conditions of joining attributes – one of them is large and the other is small. Suppose there are three relations R1, R2, R3.

R1 and R2 – Large join (R1 $\bowtie$ R2),

R1 and R3 – Smaller join (R1 $\bowtie$ R3)

If reordering of the Joins are done, you will get

(R1 $\bowtie$ R3) $\bowtie$ R2

CHECK YOUR PROGRESS

1) Describe all of the equivalence rules.

2) What is meant by query optimization?

3) How do you explain Query Cost Measurement?

4) What is Oordering of Join?

5) Fill in the blanks:

(a) _____ helps in the reduction of the Relation's size if it is performed in an early stage.

(b) The amount of time required in finding a single record by the Processor in the disk memory is termed as the _____.

1.3.4. CHOOSING THE EVALUATIONS PLANS

You should remember that only choosing the evaluation plan with its cheapest form will not always produce a good result. For solving that problem optimizers have in their systems implemented features such as Heuristic optimization and Cost - based optimizations.

HEURISTIC OPTIMIZATION

This type of optimization is very cost-effective and also choices of execution get reduced in this type of system. By using some rules as given below, query tree transformation is done by the heuristic optimizer for improving the performance of its execution.

- Selections must be performed early so that tuples get reduced.
- Projections must be performed early so that attributes get reduced.
- Join operations and restrictive selections are performed.
- Partial cost-based and only heuristic optimizations are used by different users.

1.3.4.1. COST-BASED OPTIMIZATION

This type of optimization is best for a bigger dataset but is very expensive. The process of conversion of a logical query into its physical query is performed by some rules known as the Equivalence rules. These rules determine what type of algorithms should be applied to the operations for determining the minimal cost of the operations.

Depending upon the equivalence rules, the cost-based optimizer is dependent on the following criteria.

- There must be some efficient techniques for deriving duplicate expressions.
- To avoid the formation of the copies of multiple sub-expressions, the representation of the expressions must be space-efficient.
- At the time of its first optimization, dynamic programming helps in storing the sub-expression based on Memoization and can reuse the sub-expression again and again.
- For avoiding all the plans to get generated, pruning techniques based on cost-based are applied.

1.3.5. ESTIMATION FOR THE STATISTICS OF COSTS

The cost is estimated as shown below.

1.3.5.1. COST ESTIMATION

Given below are some of the considerations for cost estimation.

- n_r – for a given relation 'r', 'n_r' is the number of tuples.
- b_r – for a given relation 'r', 'b_r' is the number of blocks.
- l_r - for a given relation 'r', 'l_r' is the size of the tuple.
- f_r - for a given relation 'r', 'f_r' is the tuples fitting into one block.
- V (A, r) - for a given relation 'r', V (A, r) is the distinct values for an attribute in relation r.
- If the above tuples are put inside a file, then you will have, $b_r = \lceil n_r / f_r \rceil$

1.3.5.2. SIZE ESTIMATION

- $\sigma_{A=v}$ **(r)**: For estimated size = 1, n_r / V (A, r) is the number of conditions that would satisfy the equality conditions.
- $\sigma_{A \leq V}$ **(r):** For the information having statistical values, it is assumed that the value of $\mathbf{c = n_r /2}$, where 'c' estimates the number of tuples.It satisfies the following minimum and maximum conditions.
  - If v < min (A, r), then c = 0;
  - $C = n_r (v - \min(A, r)) / (\max(A, r) - \min(A, r))$

1.3.6. MAINTAINING THE MATERIALIZED VIEW

Computing the contents of the costs and storing it is usually referred to as the materialized view and keeping them up-to-date is termed as materialized view maintenance. Maintenance can also be done by re-computing it. Incremental view maintenance helps in maintaining the relational database changes and updating it. Maintenance of the materialized view can be performed in the following ways:

- For each relation, insert, delete and update triggers are defined manually.
- For updating the database relations, views are updated by code which is manually written.

- Directly linked with the database.

1.3.6.1. OPERATIONS OVER THE MATERIALIZED VIEW

1) JOIN OPERATION

Let there be a relation 'r' where two states are present, r^o and r^n. Let another relation 's' be also present simultaneously. When we insert any value to the relation 'r' (i^r), we get the following $(r^n \bowtie s)$ and can be rewritten as $(r^o \cup i^r \bowtie s)$ or $(r^o \bowtie s) \cup (i^r \bowtie s)$.

2) SELECTION OPERATION

Let there be a relation 'r' where two views are present, v^o and v^n against a single view $v = \sigma_\theta(r)$. Then the selection of the view on relation r is depicted as $(v^n = v^o \cup \sigma_\theta(i^r))$.

3) PROJECTION OPERATION

This operation is complicated than the other two operations. It has a single projected tuple termed $\prod_A(r)$.

4) COUNT

Counting the number of tuples is helped by this example $(v = A^g count(B)^{(r)})$.

If tuples are already present in the view (v), only the count value is incremented, and if a new tuple needs to be added or subtracted, then the count value is given to be 1.

5) SUM

The Sum of the tuples is found out with this example $(v = A^g sum(B)^{(r)})$.

The concept of the sum is almost the same as that of the COUNT operation but instead of updating the count value for addition or subtraction of the tuples, it is needed to update the B value but the order of the count is maintained for each transaction.

1.4. SUMMING UP

- Databases are created to organize and store all the related data and information at a particular place so that the users can access and manipulate it as per requirement.

- To bridge the gap between the DBMS and the Users, a standard language known as the SQL, which is understandable by the DBMS is used between the DBMS and the Users, so that correct data are retrieved on processing.
- External Sorting is applied when the memory is small and records of large files need to be stored.
- Query blocks are formed after the division of the original SQL query, and it becomes ready for its optimization.
- When Nested queries are encountered having aggregate operators like COUNT, SUM, MIN, MAX, then a separate query block is formed.
- Executions plans are none other than the systematic way of ordering the Access Routines.
- The selection of the execution plan is done by the Catalog Manager.
- Most of the structures of the query optimizers can be seen as the Left-deep Join orders which help in reducing the complexities and optimizations.

1.5 KEY TERMS

- **SCANNER** – Scanner helps in identifying the tokenized words present in the SQL query such as the keywords, relation names, attributes, etc.
- **PARSER** – Parser checks the syntax of the SQL query which is termed parsing. It also validates whether any semantic error is present or not.
- **PARSE / QUERY TREE** – The relational algebra is internally converted into a tree structure called the Parse tree.
- **QUERY BLOCKS** – SQL queries are first converted into some basic blocks containing algebraic operators known as query blocks after which the queries are optimized.
- **OPTIMIZATION** – Choosing the most effective minimal cost of execution is termed as optimization performed by an optimizer.
- **SEEK TIME –** The amount of time required in finding a single record by the Processor in the disk memory is termed as the seek time.
- **TRANSFER TIME –** The amount of time required by the disk in returning the query result to the user or the processor is termed the transfer time.

1.6 QUESTIONS AND ANSWERS

Fill in the following blanks:

1. _____helps in identifying the tokens.

2. In query processing, _____ is a very important step.

3. The amount of merge that can be occurred together is termed as the _____.

4. Query_____ are formed after the division of the original SQL query.

5. The _____ check helps in checking whether the table, keywords, columns present in the query are also present in the Database or not.

6. With the help of heuristic rules, refinement of query execution is done so that individual operations are _____.

7. A _____ must be performed for at least a single DML statement and after that, the process of optimization is carried on.

8. Most of the structures of the query optimizers can be seen as the _____ orders.

9. Join ordering _____ the storage cost of the relation.

10. The amount of time required by the disk in returning the query result to the user or the processor is termed as the _____.

Answers: 1. Scanner, 2. Sorting, 3. Degree of merge, 4. Blocks, 5. Semantic, 6. Reordered, 7. Hard parsing, 8. Left Deep join, 9. Minimizes, 10. Transfer time

Short answer type questions:

1) Define the terms:

(a) Scanner, (b) Parser, (c) Parse tree

2) What is query processing?

3) What are the steps of query processing?

4) What happens during the execution phase?

5) Define heuristic optimization.

6) Define cost-based optimization.

7) Define seek time.

8) Define transfer time.

9) Define in brief the pushing projection.

10) How are the joins ordered?

11) What are some of the considerations for cost optimization?

12) How size is estimated for costs?

Long answer type questions:

1) Explain what is Query Processing.

2) Draw the Block Diagram of Query Processing and explain it.

3) Write the Equivalence rules for Query Processing.

4) Explain how Query Cost is measured.

5) Explain transformation with examples.

6) Explain Heuristic and Cost based optimizations.

7) What is the meaning of a materialized view? Explain the operations of the materialized view.

1.7 REFERENCES AND SUGGESTED READING

[1] https://www.tutorialcup.com/dbms/query-processing.htm
[2] https://www.geeksforgeeks.org/sql-query-processing/
[3] http://www.cbcb.umd.edu/confcour/Spring2014/CMSC424/query_optimization.pdf
[4] Database Systems Models, Languages, Design, and Application Programming by Ramez Elmasri and Shamkant B. Navathe, 5th edition by Pearson

Unit 2:
TRANSACTION PROCESSING

UNIT STRUCTURE

2.1 Learning Objectives

After going through this unit, you will be able to:

- Explain basic concept of a transaction and its different states during execution.
- Tell the desirable properties of a transaction and why are these properties significant.
- Explain the concept of a schedule and concurrent execution of multiple transaction.
- Explain the how recoverability deals with issues that arise due to transaction failure during concurrent execution of transaction.

2.2 Introduction

Consider the situation of a database user who is holding two accounts in a bank and wants to transfer some amount from his first account to the second account. The situation looks simple to implement however, it will involve several operations such as reading the balance amount available in the first account and if the balance is sufficient subtract the amount that needs to be transferred followed by updating the final balance in the first account. Then the balance in the second account is checked and the amount that needs to be transferred is added to the balance and the final balance is updated in the account. The operations involved in this example are reading, subtracting, updating, adding and writing. The set of all these operation is grouped into a single unit and is called as a transaction. So, a transaction processing system mustensure that all the operations in the transaction are executed in a proper order and either all operations are executed or none of them are executed. That is, in case of any failure, it should not be that the

first account is debited but the second account is not credited with the transferred amount. Also the transaction processing system allows multiple transaction to run concurrently such that the database remains consistent before and after the transaction. The unit introduces basic properties of a transaction, the sequence in which operations of concurrent transactions are executed and the concept of recoverability that is, acceptable schedules from the point of transaction failure.

2.3 Transaction Concepts

A transaction is a logical unit of job which accesses and possibly changes the contents of a database. Transactions include one or more database access operations such as insertion, deletion, modification and retrieval. A transaction can be embedded within an application program or can be specified using a high level query language such as SQL. There can be several transactions in an application program separated by *begin* transaction and *end* transaction.

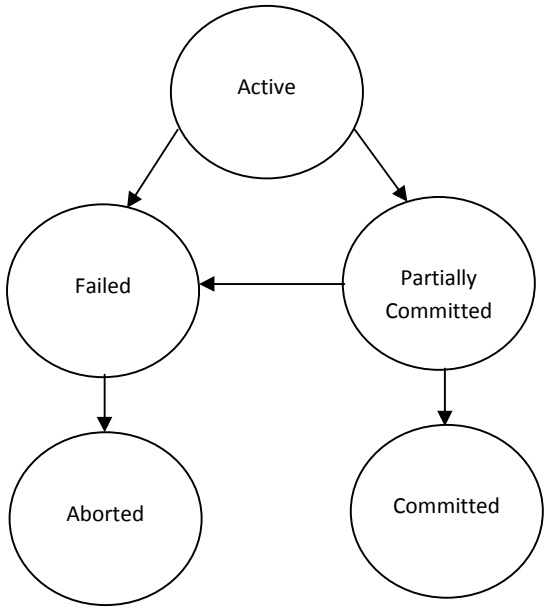


Figure 2.1: State Diagram of a transaction.

A transaction during the course of execution may go through several states as mentioned below. Figure 2.1 show the state diagram of a transaction during execution.

- Active – It is the initial state of any transaction during the period of execution.
- Partially Committed – It is the state when the transaction executes its last operation.
- Failed – It is a state when the transaction can no longer continue its normal execution.
- Aborted – It is the state when the transaction fails as a result of which it is rolled back that is the database is restored to the state earlier to the start of the transaction. At this point there are two possibilities either to restart the transaction or to kill the

transaction. In case of software or hardware error, the transaction is restarted whereas in case of logical error in the code the transaction is killed.

- Committed – It is the state when the transaction has executed all its operations successfully, that is all the changes are permanently stored on the database system.

2.4 Desirable properties

A database management system ensures the integrity of data before and after the execution of the transaction by having the following four properties, which are also called ACID properties.

- Atomicity
- Consistency
- Isolation
- Durability

Atomicity: The term atomicity means that either all the operations in a transaction are executed completely or none of the operations are executed. There should not be any partial execution i.e. a situation where only few operations of the transaction are executed and the remaining operations are not executed. For example, consider X and Y are having INR 1200 and INR 1300 respectively in their bank account. X wants to transfer INR 200 from his account to the account of Y. So, we can consider it to be a transaction having six operations as shown below:

Transaction: *fund_transfer*

Operation 1	Read balance from the account of X
Operation 2	Subtract 200 from the account of X
Operation 3	Write updated balance in the account of X
Operation 4	Read balance from the account of Y
Operation 5	Add 200 from the account of Y
Operation 6	Write updated balance in the account of Y

Suppose a failure occurs during the execution of the transaction, *fund_transfer*. The failure can be due to several reasons such as hardware failure like hard disk crash or software error or power failure. If the failure occurs between operation 3 and operation 4 then in such a situation the amount 200 will be debited from the account of X whereas the amount 200 will not be credited to the account of Y. Thus the balance in account of X and Y will respectively INR 1000 and INR 1300. The amount INR 200 will be lost due to failure and to avoid such situation atomicity or execution of all operations must be ensured in the transaction.

Consistency: A transaction is preserves the consistency of the database if, complete execution of the transaction changes the database from one consistent state to another consistent state. For example, consider the same transaction as discussed above in case of atomicity to transfer INR 200 from account of X to the account of Y.

It must be ensured that all operations in the transaction are executed completely to maintain the consistency before execution of the transaction and after the execution of the transaction. So, before the execution of the transaction sum of balance of account X and account Y is INR 2500 and after the completion of all the operations of the transaction the sum of balance of account X and account Y will still be INR 2500, which maintains the consistency of the database. However, it must be note that during the execution of the transaction, the database may be in inconsistent state. So, consistency of the database is always checked before the start of the transaction and after the completion of the transaction.

Isolation: The isolation property of a transaction is very important in context of concurrent execution of several transactions. If this property is not ensured it will result in inconsistency in database. Let us understand the property using an example.

Consider two transactions:
T1: Transfer of funds form account of X to account of Y
T2: Display total balance of account X and account Y

	T1	T2
Operation 1	Read balance of account of X	
Operation 2	Subtract 200 from the account of X	
Operation 3	Write updated balance in the account of X	
Operation 4		Read balance of account of X
Operation 5		Read balance of account of Y
Operation 6		Add both the Balance
Operation 7		Display total balance
Operation 8	Read balance of account of Y	
Operation 9	Add 200 from the account of Y	
Operation 10	Write updated balance in the account of Y	

Figure 2.2: Concurrent transaction.

Suppose transaction, T2 reads the balance in the account X and account Y sums it up and display's the total balance and all these operations are executed while transaction, T1 is in execution as shown in Figure 2.2. Even though both the transaction can complete their execution but the final result will be inconsistent in the case of transaction, T2. Considering X and Y are having INR 1200 and INR 1300 respectively in their bank account. Transaction, T2 will display the result as 1000 + 1300 = 2300 which is inconsistent and should have been 2500 instead. This inconsistency in result is due to the fact that T2 was executing before T1 could complete its execution.

One way of avoiding such inconsistency in case of concurrent execution of transaction is to execute them serially i.e. one after another. However, there are performance benefits of executing transaction concurrently. The isolation property of the transaction ensures that concurrent execution of transaction is equivalent to serial execution of the transactions in some order i.e. in concurrent execution of transaction one transaction is unaware that other transaction is executing. It looks as if both or multiple transaction are executed in isolation.

For a pair of transaction T1 & T2, it appears to T1 that T2 will start after T1 finishes its execution or T2 finished its execution before T1 started.

Durability: The durability property of a transaction ensures that after successful completion of a transaction, all the changes done to the database remains unchanged even if there is any hardware failure, software failure or power failure. Failure may result in data loss in main memory but data in disk are never lost. This is ensured by having the changes to data written to disk first before the transaction completes and also the information about the changes done to the database by the transaction that is the log file be written to disk, so that recovery from failure can be done when the system is restarts.

2.5 Schedules

Transactions are set of operations performed on the database. So, in case of concurrent execution of multiple transaction, there is a requirement of a sequence in which the operations are executed because at a time only one operation can be performed on the database. This sequence of operations is known as Schedule. For example, Figure 2.3 shows a serial schedule where transaction P completes its execution before transaction Q can start.

To access and process data, a transaction needs to perform read and write operation. A read operation, *read (X)* reads the item X from database and stores it in local buffer. A write operation, *write (X)*, writes the item X from the local buffer to the database.

For example, consider the two transaction

P:
```
read(X)
read(Y)
X = X + 100
write(X)
Y = Y + X
write(Y)
```

Q:
```
read(X)
temp=X * 0.5
X = X + temp
write(X)
```

Let us assume that the initial value of X and Y are 100 and 500 respectively. In the schedule 1 shown in Figure 2.3 the two transaction P & Q are executed serially that is one after another.So, the final value of X and Y will be 300 and 700 respectively. If, however the order of execution of the transaction is reversed with Q executing before P as shown in schedule 2 of Figure 2.4 then the final value of X and Y will be 250 and 750 respectively.

Transaction: P	Transaction: Q
read(X)	
read(Y)	
X=X+100	
write(X)	
Y = Y + X	
write(Y)	
	read(X)
	temp=X * 0.5
	X = X + temp
	write(X)

Figure 2.3: Schedule 1 - A serial schedule P followed by Q.

When several transactions are executed concurrently, then they need not be executed in serial order. The operating systems does a context switch between the transaction. That is the operating system will execute some operations of one transaction then switches to other transaction and executes some of its operations. The process of switching between the transactions cannot be decided on the number of operations that will be executed before switching. Figure 2.5 shows an example of concurrent schedule equivalent to schedule 1 that preserves the consistency of the database. It has to be noted that schedule 3 is one of the possible ways of executing transactions concurrently which preserves database consistency. The final value of X and Y will be 300 and 700 respectively for schedule 3. There are many ways of executing transaction concurrently, one such way is shown in Figure 2.6 of schedule 4. However, schedule 4 does not preserves database consistency as the final value of X and Y will be 150 and 700 respectively.

Transaction: P	Transaction: Q
	read(X)
	temp=X * 0.5
	X = X + temp
	write(X)
read(X)	
read(Y)	
X = X + 100	
write(X)	
Y = Y + X	
write(Y)	

Figure 2.4: Schedule 2 - A serial schedule Q followed by P.

Transaction: P	Transaction: Q
read(X)	
read(Y)	
X = X + 100	
write(X)	
	read(X)
	temp=X * 0.5
	X = X + temp
	write(X)
Y = Y + X	
write(Y)	

Figure 2.5: Schedule 3 - A concurrent schedule equivalent to schedule 1.

Transaction: P	Transaction: Q
read(X)	
read(Y)	
	read(X)
	temp=X * 0.5
X = X + 100	
write(X)	
	X = X + temp
	write(X)
Y = Y + X	
write(Y)	

Figure 2.6: Schedule 4 - A concurrent schedule.

2.6 Recoverability

The concept of recoverability deals with issues that arise due to transaction failure during concurrent execution of transaction. Suppose that two transactions T1 and T2 are executed concurrently. If there is some kind of dependency exists between the two transactions like the data produced by T1 using the write operation is consumed by T2 using read operation. In such case if the transaction T1 is aborted due to some failure, the dependency of T2 on T1 will result in aborting T2. In this context there are two types of schedules that are accepted from the point of recovery from transaction failure.

2.6.1 Recoverable Schedules

Recoverable schedule deals with the pair of transaction T1 and T2 such that if the data item produced by T1 is consumed by T2, then the commit operation of T1 must be executed before the commit operation of T2.

For example, in schedule 5 of Figure 2.7, the commit operation of transaction Q is executed before the transaction P.Now, if the transaction P fails before the commit operation then the data item X read by transaction Q is invalid and must be aborted. However, as the transaction Q has alreadycommitted, it cannot be aborted. Schedule 5 having commit operation by transaction Q

before execution of transaction P makes it non recoverable schedule. Therefore, it must be ensured that all schedules are be recoverable.

Transaction: P	Transaction: Q
read(X)	
write(X)	
	read(X)
read(Y)	

Figure 2.7: Schedule 5.

2.6.2 Cascadeless Schedule

Consider the situation of schedule 6 in Figure 2.8. The data item X produced by transaction P is consumed by transaction Q. Similarly, the data item X produced by transaction Q is consumed by transaction R. Clearly, we could see that for data item X dependency of transaction R on transaction Q and the dependency of transaction Q on transaction P. Now, if there is a situation where transaction P fails then transaction Q and transaction R also required to be rolled back. This is called cascading rollback and is undesirable due to significant number of rollbacks required for recovery. So, it is preferred to avoid such schedules which may have cascading effect. Such schedules which avoids cascading rollback are called Cascadeless schedule.

Transaction: P	Transaction: Q	Transaction: R
read(X)		
write(X)		
	read(X)	
	write(X)	
		read(X)
		write(X)

Figure 2.8: Schedule 6.

2.7 Check Your Progress

i. Which of the following represents the ACID properties of a transaction?
   a) Atomicity, Consistency, Integrity, Durability
   b) Atomicity, Concurrency, Isolation, Durability
   c) Atomicity, Concurrency, Integrity, Durability
   d) Atomicity, Consistency, Isolation, Durability

ii. Which of the statement is true about transaction?
   a) A program in execution
   b) A logical unit of work
   c) A set of operations
   d) A set of operations to carry out a work.

iii. Execution of a transaction in _____ preserves_____.
   a) Atomicity, Consistency
   b) Isolation, Atomicity
   c) Isolation, Consistency
   d) Consistency, Durability

iv. Which of the following is not a transaction state?
   a) Committed
   b) Failed
   c) Rollback
   d) Aborted

v. What happens to a transaction in abort state when it is rolledback?
   a) Kill or Restart
   b) Kill
   c) Restart
   d) Allow new transaction

vi. The _____ property of a transaction is preserves the consistency of the database if complete execution of the transaction changes the database from one consistent state to another consistent state.
   a) Atomicity
   b) Isolation
   c) Consistency
   d) Durability

vii. The _____property of a transaction means either all the operations in a transaction are executed completely or none of the operations are executed.

a) Atomicity
b) Isolation
c) Consistency
d) Durability

viii. _____schedule deals with the pair of transaction such that if the data item produced by first transaction is consumed by second transaction then the commit operation of first transaction must be executed before the commit operation of second transaction.

a) Recoverable
b) Cascadeless
c) Cascading
d) Durable

ix. To avoid cascading rollback, it is desirable to have _____ schedule.

a) Recoverable
b) Cascadeless
c) Cascading
d) Durable

x. The _____ property of a transaction ensures that after successful completion of a transaction, all the changes done to the database remains unchanged even if there is any failure.

a) Atomicity
b) Isolation
c) Consistency
d) Durability

2.8 Answers to Check Your Progress

i, d	ii, d	iii, c	iv, c	v, a
vi, c	vii,a	viii,a	ix,b	x,d

2.9 Let us Sum Up

- A transaction is a logical unit of job which accesses and possibly changes the contents of a database.

- A transaction during the course of execution may go through several states such as active, partially committed, Failed, Aborted and Committed.
- A transaction is said to be in Committed state when the transaction has executed all its operations successfully, and all the changes are permanently stored on the database system.
- A transaction is in Aborted state when the transaction fails as a result of which it is rolled back that is the database is restored to the state earlier to the start of the transaction.
- A database management system ensures the integrity of data before and after the execution of the transaction by having the following four properties, which are Atomicity, Consistency, Isolation and Durability.
- The term atomicity means that either all the operations in a transaction are executed completely or none of the operations are executed.
- A transaction is preserves the consistency of the database if complete execution of the transaction changes the database from one consistent state to another consistent state.
- The durability property of a transaction ensures that after successful completion of a transaction, all the changes done to the database remains unchanged even if there is any failure.
- The sequence of operations in case of concurrent execution of multiple transaction is known as Schedule.
- Recoverability deals with issues that arise due to transaction failure during concurrent execution of transaction.

2.10 Further Reading

- Database System Concepts 6th Edition by Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw – Hill International Edition.
- Fundamentals of Database System Seventh Edition, by Elmasri Ramez and Navathe Shamkant, Pearson.
- Database Management Systems by Raghu Ramakrishnan, Johannes Gehrke,Irwin Computer Science
- Database Systems: The Complete Book by Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom

2.11 Model Question

Q1 List the ACID properties of a transaction. What is the significance of these properties?

Q2 A transaction goes through several states before it commits or aborts. Explain all these states and its importance.

Q3 What is a transaction? Does serial execution of two transaction equivalent to concurrent execution of the same two transactions.

Q4 What are the benefits of concurrent execution of transactions?

Q5 Define a schedule with a proper example.

Q6 Explain the isolation property of a transaction.

Q7 Write a schedule that suffers from cascading rollback and a schedule that do not suffer from cascading rollback.

Q8 When a transaction enters abort state, what are the actions taken by the database system.

Q9 What is a recoverable schedule?

Q10 What is a Cascadeless schedule? Explain using an example.

Q11 Consider two transaction T1 and T2. Does the order of execution affect the final result of the common data items involved in the execution?

Q12 Consider the transaction given below with initial values of X and Y as 100 and 200. What will be the final value of X and Y if the transactions are executed in the order:

i. T1 followed by T2

ii. T2 followed by T1

T1:

```
read(X)
X = X * 10
write(X)
read(Y)
temp=X+X*0.2
Y = Y + temp
write(Y)
```

T2:

```
read(X)
X = X + 20
write(X)
```

Unit 3: Concurrency Control and Recovery Techniques

CONTENTS

3.0 Introduction

Banking industry, online shopping, stock markets, airline reservation, IRCTC reservation required transaction processing system which allows large databases to access and thousands of concurrent users perform operations on database transactions concurrently. The system that is employed for such tasks should be of high availability and faster response time to cope with hundreds and thousands of concurrent users. A transaction is a logical unit of database processing in which it includes commands such as retrievals, insertion, update and deletion.

3.1 Objectives

The unit is describing the concurrency control and recovery techniques in database transactions. After completing the unit students′ will able to:

- Understand the requirement of Concurrency Controls
- Describe the States of Transaction
- Explanation of Transactional Properties
- Detailed discussion on Schedules of Transaction
- Concurrency Controls
- Learn about Locks and Locking
- Lastly why Recovery is necessary

3.2 Concurrency

A database which is used by multi-users, access the data concurrently. However, a single users database management system is limited to personal computers only; most database management systems are multi-users. The concept of multiprogramming allows the operating system (OS) of the computer to execute multiple processes at the same time such that multiple users can access the database simultaneously. In a single Central Processing Unit (CPU) execute only a single process at a particular time. While, in a multiprogramming OS executes a process then halt the process and execute the next process, so on and so forth. A process which is halted earlier is resumed at an instance where it was suspended whenever CPU processing time is given to it. This concurrent execution of processes is interleaved which means when a process is in the CPU and waiting for Input or Output (I/O) operation, the CPU time is shifted to another process that way the CPU is always kept busy. Suppose, there are two processes P_1 and P_2executing concurrently in an interleaved manner. Process P_1 is waiting for I/O operation , process P_2 will be executed which was waiting for CPU time such that interleaved method doesn't allow the CPU to be idle while P_1 is waiting for I/O time. Benefit of interleave is that it doesn't allow a long process to delay other processes.

In DBMS, the primary resources are the stored data in the database that can be accessed concurrently by multiuser which allows the user to retrieve and modify the database concurrently.

STOP TO CONSIDER

- A multiprogramming operating system executes multiple processes concurrently similarly using a multiprogramming operating system multiple transactions of DBMS are achieved concurrently.

3.2.1 Requirement of Concurrency Control

Concurrency control and recovery mechanisms are deployed on database operations in a transaction. Various users may submit transactions which are executed concurrently to access and update the database items. If uncontrolled concurrent transactions are executed it may lead to many issues such as an inconsistent database. Some of these problems are discussed by taking a Railway ticket reservation system.

READ (A): Reads a database named A into a variable in a program.
WRITE (A): Writes the value of variable A into the database item named A.

T_1
READ (A); A = A - X; WRITE (A); READ (B); B = B + N;

Figure 3.0 a) Transaction T_1

T_2
READ (A); A = A + R; WRITE (A);

Figure 3.0 b) Transaction T_2

Figure 3.0 (a) Transfer of X reserved seats from one train which is stored in database item named A to another train whose reserved seats are stored in database item named B in transaction T_1. Figure 3.0 (b) T_2 transaction implies to reserved R seats which refers to transaction in T_1 that is A.

The same program can be used for multiple transactions as each of the transactions will have a different date, number of reserved seats and train number. In figure 3.0 a) and b) there are T_1 and T_2 transaction have specific date, seat and train number which are stored in A and B database that is the purpose of concurrency control. If these two transactions are executed concurrently we may encounter different types of problems which are discussed below

- **Temporary Update or Dirty Read Problem.** This issue occurs when a transaction makes changes in the database item, then the transaction fails due to some error. At the same time, the updated item from the database is read by another transaction before it could be changed back to its original value in the database.

T_1	T_2
READ (A); A = A - X; WRITE (A); READ (Y);	 READ (A); A = A + R; WRITE (A);

Figure 3.1 Temporary Update

For example in Figure 3.1 shows an item A is updated by T_1 transaction and suppose T_1 fails before the transaction completion. As the transaction fails to complete the system should change A to the original value in the database . Transaction T_2 access the value of A which becomes a temporary value before the system could update it in the database and the value of A is not recorded permanently in the database. As this temporary value of A is accessed by T_2 transaction that is why this problem is called dirty read.

- **Incorrect Summary Problem.** When a transaction is processing a number of database items by calculating an aggregate summary function and another transaction is updating these items. While the aggregate function may calculate few values before and after updation of the database items.

T_1	T_2

SUM = 0; READ (X); SUM = SUM +X; • • • READ (A); SUM = SUM + A; READ (B); SUM = SUM + B;	 READ (A); A = A - R; WRITE (A); READ (B); B = B + R; WRITE (B);

Figure 3.2 Incorrect Summary Problem

For example, T_1 transaction is executing and calculating number of reserved seats in a train as shown in Figure 3.2. The transactions are running in an interleaved manner. The resultant value of T_1 will be incorrect by an amount X as T_1 access the value of A once X seats are subtracted from A and the value of B is accessed before the reserved seats X are added to it.

- **Loss Update Problem.** This kind of issue occurs when two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent.

T_1	T_2
READ (A); A = A - X; WRITE (A); WRITE (B); B = B + X; WRITE (B);	 READ (A); A = A + R; WRITE (A);

Figure 3.3 Loss Update Problem

We presume that the transactions T_1 and T_2 are executed at the same time and they are processed in a interleaved manner. In Figure 3.3 the final result of A is incorrect as T_2 reads the value of X, update it before T_1 changes it in the database. Value of A is overwritten in T_2 and hence the updated value of T_1 is lost.

- **Unrepeatable Read Problem.** This problem occurs when two or more reading variables of the same transaction tries to access different values of the same variable. As transaction T_1 read some value twice during a single transaction and the value of the item is changed by T_2 transaction in between the read commands. That is why it is called unrepeatable read as it gets different values of the same item.

CHECK YOUR PROGRESS

1. What is concurrency in DBMS?
2. What is a multiprogramming operating system?
3. How are processes executed in an interleaved manner?
4. Why are concurrency controls required?
5. What kind of problem occurs when two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent?

3.3 Transaction and Transactional Properties

The operations on a database in a form of transaction can be done by a user interface program or through a query language such as SQL. To form a transaction it is to specify transaction boundaries which are BEGIN TRANSACTION and END TRANSACTION statements in an application program. The access operation of all databases between these two boundaries are considered as one transaction. To contain more than one transaction in a single program, it has to contain several transaction boundaries.

Read-Only Transaction. The program to retrieve only data and not to update the database in a transaction.

Read-Write Transaction. The program to retrieve and update the database in a transaction.

3.3.1 States of Transaction

A transaction is an atomic unit which means if a transaction is executed it should be completed entirely or not at all. For the purpose of recovery the system needs to monitor each transaction starts, terminate and abort. The recovery system should monitor the following

Begin_Transaction. Keeps the track of execution of the beginning of a transaction.

READ or WRITE. Read and write operation which is executed on a database as a part transaction.
End_Transaction. This monitors the operation of READ and WRITE have ended which is the end of transaction.
Commit_Transaction. It means a successful execution of a transaction, any changes executed by the transaction can be committed to the database and will not be changed.
RollBack or Abort. It means the transaction is unsuccessfully ended and the changes made by the transaction to the database must be ROLLBACK or undone.
The execution states of a transaction as follows:

A transaction starts, it moves from inactive to active state and it executes READ and WRITE operation. It enters a partially committed state once the transaction is completed. This activates some recovery protocols at this time, if the system fails it will not result in loss of data permanently. After the recovery protocol is executed successfully the transaction enters the committed state. In the committed state it is concluded that the execution is completed and the data are permanently recorded in the database, even if the system encounters a failure.
While a transaction can enter a fail state, if any failure of the checks is noticed or during the active state the transaction is aborted. Then the transaction may have to roll back the WRITE operation which is executed on the database. The transaction is halted due to the error corresponding to the terminated state**.**

3.3.2 Transactional Properties

Transactions should have four properties which are popularly known as ACID properties; these properties should be employed by the concurrency control and recovery mechanism of the database. The ACID is an abbreviation of Actomicity, Consistency, Isolation and Durability. These are as follows:
Atomicity: A transaction should be atomic in nature; it should either be completely executed or not at all.
Consistency: A transaction is completely executed from beginning to end without any interference of other transactions; it means a transaction is consistency preserving. A consistent transaction will take the database from one state of consistency to another.
Isolation: A transaction should look like it is being run in isolation without interference from other transactions, even if many transactions are running concurrently.
Durability: The updates or changes executed on the database items by a committed transaction must be persistent. The changes (update) should not be lost due to any failure in the system.

3.4 Schedules of Transactions

A schedule of transactions is an order list of transaction such as *S* schedule of *m* transactions will be T_1 , T_2 , T_3 , ……, T_m . In a schedule *S* operations of different transactions can be interleaved. The transactions in the schedule *S* should be in the same order as they are in T_i. Schedule *S* operations are considered to be in total order which means one operation may execute before another from any two operations in the schedule.

In this topic we are interested in recovery and concurrency controls so our objectives are READ, WRITE, commit and abort operation. Here we used a few shorthand notation for begin_transaction is b, READ is r, WRITE is w, end_transaction is e, commit is c and abort is a such that

S_1: $r_1(A)$; $r_2(A)$; $w_1(A)$; $r_1(B)$; $w_2(A)$; $w_1(B)$;

S_2: $r_1(A)$; $w_1(A)$; $r_2(A)$; $w_2(A)$; $r_1(B)$; a_1;

To be conflict the two operation in a schedule have to satisfy the three following conditions

1. Operations belong to different transactions.
2. Operations access the same item A.
3. Minimum one of the operations is WRITE.

In schedule S_1, operations $r_1(A)$ and $w_1(A)$ conflict. Similarly operations $r_2(A)$ and $w_1(A)$ and operation $r_1(A)$ and $r_2(A)$ do not conflict as they are read operations. Operations $w_2(A)$ and $w_1(B)$ also do not conflict as w_2 is on A and w_1 operation is on B. Also, the operations $r_1(A)$ and $w_1(A)$ do not conflict as both the operations belongs to the same transaction.

READ-WRITE Conflict. The order of two operation are changed such as from $r_1(A)$; $w_2(A)$ to $w_2(A)$; $r_1(A)$, this changes in the transaction T_1 as it read the value of A as in the second order, the value of A is updated by $w_2(A)$ before it can be read by $r_1(A)$.

WRITE-WRITE Conflict. If the order of two operation is changed from $w_1(A)$; $w_2(A)$ to $w_2(A)$; $w_1(A)$. In write-write conflict the final value of A will differ as in first case it is changed by T_2 and later by T_1. It is to be noticed that two read operations do not conflict by changing their order which makes no difference in the final result.

Schedules depend on serializability. The schedules which are always observed to be correct when executed concurrently are known as serializable schedules. Suppose a DBMS transaction is submitted with T_1 and T_2 simultaneously. If no interleaved operation is allowed; it will lead to two possible results.

A. In transaction T_1 all the operation will be executed in sequence followed by transaction T_2 in sequence.
B. All operations of transaction T_2 will be executed followed transaction T_1 in sequence such transactions are called serial schedule.

3.5 Serial, Nonserial and Conflict-Serializable Schedule

T_1	T_2

READ (A); A = A - R; WRITE (A); READ (B); B = B + R; WRITE (B);	READ (A); A = A + O; WRITE (A);

Figure 3.5 (a) Schedule 1

T_1	T_2
READ (A); A = A - R; WRITE (A); READ (B); B = B + R; WRITE (B);	READ (A); A = A + O; WRITE (A);

Figure 3.5 (b) Schedule 2

T_1	T_2
READ (A); A = A - R; WRITE (A); READ (B); B = B + R; WRITE (B);	 READ (A); A = A + O; WRITE (A);

Figure 3.5 (c) Schedule 3

T_1	T_2

READ (A); A = A - R; WRITE (A); READ (B); B = B + R; WRITE (B);	 READ (A); A = A + O; WRITE (A);

Figure 3.5 (d) Schedule 4

Schedule 1 in Figure 3.5 (a) and Schedule 2 in Figure 3.5 (b) are serial as the operations of each transaction are executed without interleaved operation from other transactions. In a serial schedule, transaction are executed serially such that in the order of T_1 and then T_2 and T_2 and then T_1.
Schedule 3 in Figure 3.5 (c) and Schedule 4 in Figure 3.5 (d) each from the two transactions each sequence interleaves operation so they are called nonserial schedules. Initializing database items with the value A = 80, B = 80 and R = 3 and O = 2. After transaction T_1 and T_2 are executed. We expected the database values to be A = 79 and B = 83 according to the layout of the transaction. Schedule 1 or Schedule 2 gives a correct outcome as they are executed in a serial schedule. Considering nonserial Schedule 3 and Schedule 4 for execution. Schedule 3 gives outcome A = 82 and B = 83, where the value of A is wrong and Schedule 4 gives a correct outcome.
The outcome of Schedule 3 is wrong due to the lost update problem. While some non-serial schedules are calculated correctly such as Schedule 4. We would like to know which gives the wrong result and which gives the correct result. This is used to characterize schedules in the fashion of serializability of a schedule.
By the definition of serializable schedule: A schedule S of m transactions is serializability when it is equivalent to a few serial schedules of the same m transaction. By implying a non-serial schedule is serializable it means the result is correct that is the same as in a serial schedule which is regarded as correct. Two schedules that produce the same final outcome of the database are called result equivalent. The schedule to be equivalent to the execution done on each item in one schedule should also be applied to database items in both the schedules in similar order.
If the order of any two conflicting operations is the same it is said to be the conflicting equivalent, if conflict equivalent some schedule S` and it can be reorder the non conflicting operation in S till it forms the equivalent series of schedule S` is said to be conflict serializable.

STOP TO CONSIDER

- Transactions should have four properties which are popularly known as ACID properties; these properties should be employed by the concurrency control and recovery mechanism

of the database. The ACID is an abbreviation of Actomicity, Consistency, Isolation and Durability.

- The schedules which are always observed to be correct when executed concurrently are known as serializable schedules

CHECK YOUR PROGRESS

6. What do you mean by transaction in DBMS?
7. Define Read- only transaction?
8. Define Write-only transaction?
9. Why is a transaction required to be atomic?
10. What are the states of a transaction?

3.6 Concurrency Controls

3.6.1 Lock and Modes of Locking

A variable associated with each data that implies whether a read operation or a write operation is to be implemented to the data items is called lock. The speciality of lock is that it gives synchronized access to the data items by concurrent transactions.

The concurrency control technique which allows the locked data items to be accessed and manipulated is called locking, to maintain the consistency and integrity of the database. DBMS implements two modes of locking namely Exclusive and Shared.

Exclusive Lock. It provides exclusive controls on the data item to a transaction. The transaction must acquire the exclusive lock to read and to write a data item. Hence, an exclusive lock is also known as update lock or write lock. For example, a transaction T_1 acquires exclusive lock on data item P. No other transaction is allowed to access P until transaction T_1 releases its lock on P.

Shared Lock. When a transaction wants to read a data item only, not to modify it in such instances, shared lock can be implemented on that data item. It is also known as read lock. For example, a transaction T_1 has acquired a shared lock on data item P, T_1 can read P but cannot write on P. Furthermore, multiple transactions can acquire shared locks on P at the same time. However, any transaction can acquire an exclusive lock on P.

3.6.2 Lock Compatibility

When a transaction requires to perform some operation on a data item, it requests for an appropriate lock mode on the data item. The lock manager grants lock immediately if the requested data item is not locked by any other transaction. Otherwise, the lock request may or may not grant depending on the compatibility of locks. Lock compatibility regulates whether locks can be

acquired on a data item by any number of transactions simultaneously. For example, a transaction T_1 request a lock of mode m_1 on a data item P on which another T_2 transaction currently holding a lock of mode m_2. If mode m_1 and m_2 are compatible the request will be grant immediately; otherwise rejected. The lock compatibility matrix:

Requested Mode	Shared	Exclusive
Shared	YES	NO
Exclusive	NO	NO

Figure 3.6 Compatibility matrix

In figure 3.6 the term 'YES' indicates the request can be granted and the term 'NO' indicates request cannot be granted.

The lock request of transaction T_1 is granted immediately if m_1 is shared and, if and only if m_2 is also shared. Otherwise the lock request is not granted and transaction T_1 has to wait. Furthermore, if mode $m_1$is exclusive, then the lock request by transaction T_1 will not be granted and T_1 has to wait.

3.6.3 Two- Phase Locking Techniques for Concurrency Control

Two phase locking 2PL is a lock-based concurrency control technique that divides each transaction into two phases. At the first phase, the transaction acquires all the locks and during the second phase it releases all the locks.

- **Grow Phase.** In this phase the number of locks held by a transaction increases from zero to maximum.
- **Shrinking Phase.** During this phase the locks are released, due to which it is called the shrinking phase. The number of locks held by the transaction decreases from maximum to zero.

Whenever a transaction releases a lock data item it enters the shrinking phase. Once a data item in the shrinking phase, it is not allowed to acquire any locks further. Therefore, the release of locks must be delayed until all the required locks on the data items are acquired. Considering two transaction T_1 and T_2 along with their lock request

T_1
LOCK-X (A); READ (A); A = A - 200; WRITE (A); UNLOCK (A); LOCK-X (P); READ (P);

P = P + 200; WRITE (P); UNLOCK (A);

T_2
LOCK-X (SUM); SUM = 0; LOCK = 0; LOCK -S (P); READ (P); SUM = SUM + P; UNLOCK (P); LOCK-S (A); READ (A); SUM = SUM + A; WRITE (SUM); UNLOCK (A); UNLOCK (SUM);

Figure 3.7 Transaction T_1 and T_2 with their lock request

T_1	T_2
LOCK-X (A); READ (A); A = A - 200; WRITE (A); UNLOCK (A); LOCK-X (P); READ (P); P = P + 200; WRITE (P); UNLOCK (A);	LOCK-X (SUM); SUM = 0; LOCK = 0; LOCK -S (P); READ (P); SUM = SUM + P; UNLOCK (P); LOCK-S (A); READ (A); SUM = SUM + A; WRITE (SUM);

Figure 3.8 Transaction T_1 and T_2 in two-phase locking

In figure 3.8 the statement for releasing the lock is written at the end of the transaction. But, such statement do not needs to appear at the end of the transaction to retain two-phase locking properties such as the UNLOCK (A) statement of T_1 may appear just after the LOCK- X(P) statement and still maintain the two phase locking property.

1.6.4 Deadlock

Deadlock occurs when all transactions in a set of two or more transactions are waiting for some data item that is locked by some other transaction. Hence, each transaction in the set is waiting for the other waiting transaction in the set. None of the transactions can proceed until and unless one of the waiting transactions releases lock on the data item. Deadlock can be prevented by applying deadlock prevention protocols.

3.7 Recovery of DBMS

When a transaction of DBMS is executed, it is the responsibility of the system to make sure all the operations in a transaction are completed successfully and their outcomes are recorded in the database permanently or the transaction has no effect on the database. If the transaction fails while executing it shouldn't have any effect on the database.
Types of failure may encounter which executing DBMS in the middle of a transaction are:

1. **A System Crash.** During the execution of the transaction there could be a hardware, software or network error. Hardware crash mainly consists of main memory failure.
2. **System error.** The transaction may cause failure due to the programming error which may be caused by various reasons such as division by zero or integer overflow. Furthermore, the user may also interrupt the execution of the transaction in between.
3. **Concurrency Control Enforcement.** A transaction can be aborted by concurrency control method due to the violation of serializability or it may also abort transaction or more transactions to avoid deadlock among several transactions. Transactions aborted due to these kinds of reasons are started automatically later.
4. **Exception Conditions or local detected by the transaction.** During the execution of the transaction, many necessary conditions need to be fulfilled such as data for a transaction may not be found.
5. **Disk Failure.** There can be a disk head crash during read or write operation or some disk block may lose their data due to a read or write malfunction. Such events can happen during a read or write operation of a transaction.
6. **Physical Damage and Catastrophes.** This is a never ending list of problems which includes power failure, fire, theft, over writing disk by mistake, damaging the hardware by physical means etc..

3.8 Transaction Failure

A transaction aborts when it is failed to execute or a transaction reaches an instance where it can't go any further. The atomicity property of transaction, which suggested that all operations in a transaction have to execute completely or not at all. There cannot be a case where only half of the

operation will be executed or else this will lead to a transaction failure. Reason for transactions failure:

Logical error: This kind of error occurs when a transaction cannot complete due to some code errors or any internal condition error.

System error: In this kind of error, when the database system terminates an active execution of a transaction as the system is not able to execute it. This kind of error occurs due to deadlock or unavailability of resources, the system aborts an active transaction.

3.9 Recovery System in DBMS for transaction Failure

There are two techniques to recover if the system encounters a transaction failure. They are

1. Log-based recovery
2. Shadow paging

3.9.1 Log-based recovery

A log is a sequence of records that maintains the history of all the updates which are implemented on the database. It used to hold the records of modification done on the database and it is also known as system log. In an ongoing transaction, if the system crashes, by using log files it can be returned back to its previous state as if nothing has happened to the database.

Suppose,

$< T_1, X_1, D_1, D_2 >$: Updates log records where, T_1 is the transaction, X_1 is the data, D_1 is the previous data and D_2 is the new data.

$< T_1$, starts $>$: Transaction T_1 start executing

$< T_1$, Commit $>$: Transaction T_1 to commit

$< T_1$, Abort $>$: Transaction T_1 is aborted

Deferred Database Modification

In deferred database modification the updation is delayed or deferred in the database until the last operation of a transaction is executed and reaches to completion.

Recovery system:

Redo (T_1) : All data items updated by the transaction T_1 are set to a new value.

Immediate Database Modification

In this type of modification, the database is modified after a WRITE operation, it immediately modifies the database whenever a transaction performs a WRITE or update operation. Update log records contain both previous and new values of data items.

Recovery system:

Undo (T_1) : All the data items changed by T_1 transaction are set to the previous values.

Redo (T_1) : All the data items changed by T_1 transaction are set to new values.

3.9.2 Shadow Paging

Shadow paging is an alternative to log-based recovery. In shadow paging it maintains two tables during the execution period of a transaction; a current table and a shadow table. The shadow page table is stored in non-volatile memory, such that the state of the database prior to transaction execution can be removed and the shadow page table is never modified during execution.
Both the page tables are identical where the current page table is accessing used data items during the execution of the transactions. Wherever any page is written for the first time, a copy of this page is made on an inside page. The current page table is then made to point to the copy and the updation is performed on the copy

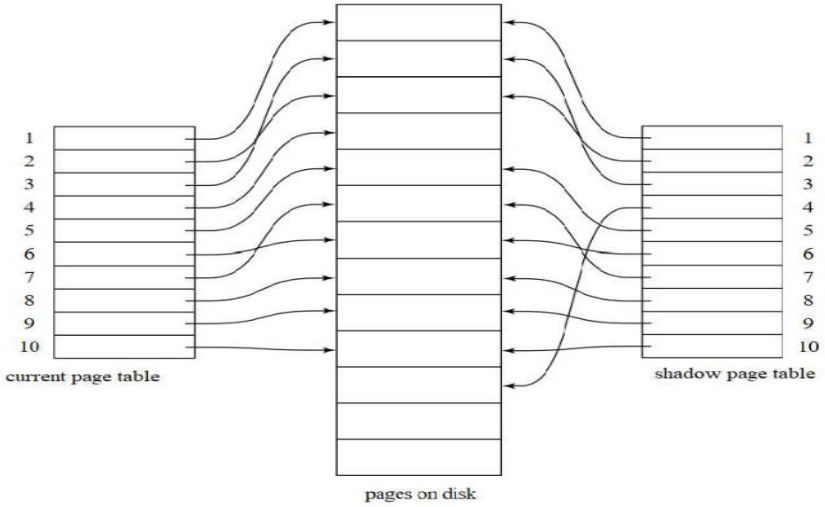


Figure 3.9 :Shadow and current page table

To commit a transaction

1. Put all modified pages in main memory to disk
2. Put all output current page table to disk.
3. Make the current page, the new shadow page. Keeps a pointer to the shadow page table at a fixed known location on disk.

Once the pointer to the shadow page table has been written, a transaction is committed. If there is a crash encountered no recovery is required. After a crash a new transaction can start immediately using the shadow page table.

3.10 SUMMARY

You have learned that the database management system is a very complex system. To understand the concept of transactions which is a logical unit of DBMS processing it is essential to understand transactional properties, concurrency controls and recovery which are explained and discussed extensively in this unit.
This unit concentrates on how concurrency in transactions is achieved with the help of multiprogramming OS and it is implemented on the database management system. It also explained how multiple processes are executed in an interleaved manner.

The requirements of concurrency controls are explained. What are the problems that a system will face if uncontrolled transactions are implemented and detailed discussion on temporary update, incorrect summary, loss update and unrepeatable read problems are done. With great details the transactions and its properties are discussed such as READ-Only transaction and READ-WRITE transactions which leads us to the ACID properties of the transaction. And if a transaction doesn't follow the ACID properties there will be an inconsistent database.
Transaction has its history which is known as Schedule of transactions that formalized the discussion on the different serial and non serial transactions with sufficient examples. Serializability schedules are illustrated with examples such as non-conflicting operations and how conflict serializable operations are formed.
The concurrency controls such as the locks and locking which are exclusive lock and shared lock that holds the access of a data item are held and the grant of access decision of a data item are made by these locks to achieve concurrency. The compliance of modes of locking makes access decisions on multiple data items to be accessed or not simultaneously. Two phase locking is a lock which is implemented in two different phases such as grow and shrinking phase.
Lastly in this unit the discussion on the deadlock situation in DBMS is also discussed and recovery of the database is necessary to save it from failures which may occur are also discussed.

3.11 Key terms

- **Transaction:** A transaction is a logical unit of database processing in which it includes commands such as retrievals, insertion, update and deletion.
- **READ Operation:** Reads a database item into a variable in a program.
- **WRITE Operation:** Writes the value of a variable in a program into the database item .
- **Dirty Read Problem:**This issue occurs when a transaction makes changes in the database item, then the transaction fails due to some error. At the same time, the updated item from the database is read by another transaction before it could be changed back to its original value in the database.
- **Incorrect Summary Problem:** When a transaction is processing a number of database items by calculating an aggregate summary function and another transaction is updating these items. While the aggregate function may calculate few values before and after updation of the database items
- **Loss Update Problem:** This kind of issue occurs when two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent.
- **Unrepeatable Read Problem:** This problem occurs when two or more reading variables of the same transaction tries to access different values of the same variable.
- **Read-Only Transaction:** The program to retrieve only data and not to update the database in a transaction.

- **Read-Write Transaction:** The program to retrieve and update the database in a transaction.
- **ACID :** ACID is the abbreviation of Actomicity, Consistency, Isolation and Durability.
- **Serializable schedules:**The schedules which are always observed to be correct when executed concurrently are known as serializable schedules.
- **Serial schedule:** In a serial schedule, transaction are executed serially
- **Lock:** A variable associated with each data that implies whether a read operation or a write operation is to be implemented to the data items is called lock.
- **Locking:** The concurrency control technique which allows the locked data items to be accessed and manipulated is called locking, to maintain the consistency and integrity of the database.
- **Deadlock**: Deadlock occurs when all transactions in a set of two or more transactions are waiting for some data item that is locked by some other transaction. Hence, each transaction in the set is waiting in a never ending process.

3.12 Answers to check your progress

1. Concurrency in DBMS is the process of storing data in the database that can be accessed concurrently by multiuser which allows the user to retrieve and modify the database concurrently.
2. OS executes a process then halt the process and execute the next process, so on and so forth. A process which is halted earlier is resumed at an instance where it was suspended whenever CPU processing time is given to it; this is called Multiprogramming OS.
3. The concurrent execution of processes is interleaved which means when a process is in the CPU and waiting for Input or Output (I/O) operation, the CPU time is shifted to another process that way the CPU is always kept busy.
4. Concurrency control and recovery mechanisms are deployed on database operations in a transaction. Various users may submit transactions which are executed concurrently to access and update the database items. If uncontrolled concurrent transactions are executed it may lead to many issues such as an inconsistent database.
5. Loss Update is the kind of problem that occurs when two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent.
6. A transaction is a logical unit of database processing in which it includes commands such as retrievals, insertion, update and deletion.
7. Read-Only Transaction is a program to retrieve only data and not to update the database in a transaction.
8. Read-Write Transaction is a program to retrieve and update the database in a transaction.
9. A transaction needs to be an atomic unit which means if a transaction is executed it should be completed entirely or not at all to make the database consistent

10. The states of transactions are Begin Transaction, READ or WRITE. End Transaction, Commit Transaction and RollBack or Abort.

3.13 Questions and Answers

Multiple Choice Questions

1. When multiple transactions are processed in a controlled manner using some protocols is known as
   (a) Data Control
   (b) Entity Control
   (c) Concurrency Control
   (d) DBMS Control

2. The concurrent execution of processing is interleaved transaction will be executed in
   (a) T_1 then T_2 and T_2 then T_1
   (b) T_1 then T_4 and T_2 then T_1
   (c) T_9 then T_5 and T_2 then T_1
   (d) None of these

3. When two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent this kind of problem is known as
   (a) Loss Update Problem
   (b) Dirty Read Problem
   (c) Unrepeatable Read Problem
   (d) Incorrect Summary

4. During the process of transaction the recovery system should monitor
   (a) Begin Transaction
   (b) Read Write
   (c) Commit Transaction
   (d) All of these above

5. When a transaction is unsuccessfully ended the changes made by the transaction to the database must be
   (a) Read
   (b) Write
   (c) Rollback or undone
   (d) All of the above

6. As transaction T_1 read some value twice during a single transaction and the value of the item is changed by T_2 transaction in between the read commands. That is called
   (a) Temporary Update Problem
   (b) Unrepeatable Read Problem
   (c) Loss Update Problem
   (d) Incorrect Summary Problem

7. Transactional properties should have four properties popularly known as
   (a) Read - write property
   (b) Write - read property
   (c) ACID property
   (d) BASE property

8. The concurrency control technique allows to lock data items to be
   (a) Accessed and manipulate
   (b) Extract information
   (c) Enter and Access
   (d) Process Information

9. How many locking modes are there
   (a) One
   (b) Two
   (c) Three
   (d) Four

10. Which incident may cause physical damage to the system
    (a) Earthquake
    (b) Hacking
    (c) Virus
    (d) None of these

Answers: 1. (c) , 2.(a), 3.(a), 4.(d), 5.(a), 6. (b), 7.(c) 8.(a), 9.(b), 10.(a)

Fill in the following blanks

1. A transaction is a _____ of database processing in which it includes commands such as retrievals, insertion, update and deletion.
2. To form a _____ it is to specify transaction boundaries which are BEGIN TRANSACTION and END TRANSACTION

3. A transaction is _____ which means if a transaction is executed it should be completed entirely or not at all.
4. A transaction starts executing, it moves from inactive to active state and it executes _____ and_____operation.
5. _____ occurs when two or more reading variables of the same transaction tries to access different values of the same variable.
6. The program to retrieve only data and not to update the database is _____ transaction.
7. A _____ transaction will take the database from one state of consistency to another.
8. The schedules which are always observed to be correct when executed concurrently are known as schedules.
9. In a _____ schedule, transaction are executed serially such that in the order of T_1 and then T_2 and T_2 and then T_1.
10. At the first phase of two-phase locking, the transaction _____ all the locks and during the second phase it releases all the locks.

Answers: 1. logical unit, 2. transaction, 3. an atomic unit, 4. READ and WRITE, 5.Unrepeatable Read Problem, 6. Read-Only, 7. consistent, 8. serializable, 9. serial, 10. acquires

Match column A with column B

	column A		**column B**
1.	Multiple transactions executing concurrently	A	Shrinking Phase
2.	In this phase the number of locks held by a transaction increases from zero to maximum	B	Abort
3	Reads a database item into a variable in a program	C	Exclusive Lock
4	Temporary value of a variable accessed by a transaction	D	Theft
5	In this phase the locks are released, due to which it is called the shrinking phase. The	E	Concurrency

	number of locks held by the transaction decreases from maximum to zero.		
6	It means the transaction is unsuccessfully ended and the changes made by the transaction to the database must be ROLLBACK or undone	F	Loss Update Problem
7	A transaction is completely executed from beginning to end without any interference of other transactions; it means a transaction is consistency preserving. A consistent transaction will take the database from one state of consistency to another.	G	Grow Phase
8	It provides exclusive controls on the data item to a transaction. The transaction must acquire the exclusive lock to read and to write a data item. Hence, an exclusive lock is also known as update lock or write lock.	H	Consistency
9	Catastrophic Failure	I	Shared Lock
10	When a transaction wants to read a data item only, not to modify it in such instances, shared lock can be implemented on that data item.	J	Read Operation

Answers: 1. (E), 2. (G), 3. (J), 4. (A), 5. (F), 6. (B), 7. (H), 8. (C), 9. (D), 10. (I)

State whether True or False

1. The multiprogramming allows the operating system of the computer to execute multiple processes at the same time.
2. If an uncontrolled concurrent transaction is executed it may lead to a consistent database.
3. When a value of variable is stored temporarily and change is called dirty read
4. The access operation of all databases between begin transaction and end transaction boundaries is one considered as one transaction.
5. In serial schedule transaction are executed randomly such as T_1, T_3 and T_2 or T_2, T_1 and T_3 .

6. The recovery protocols are executed after the transaction enters the committed state.
7. Transactions should have four properties which are known as BASE properties.
8. Insolation is a property of a transaction that should look like it is being run in isolation without interference from other transactions.
9. Schedule depends on serializability.
10. Lock means locking a data variable which dined access variable for any operation.

Answer 1. True, 2. False, 3. True, 4. True, 5. False, 6. True, 7. False, 8. True, 9. True, 10. False.

Short answers type questions

1. Why are concurrency controls required?
2. Explains the phases of transactional states and its execution process.
3. Discuss the desirable properties of transactions.
4. What are the necessary conditions to be fulfilled by a transaction to be in conflict ?
5. What will be the possible outcome for two transactions submitted simultaneously for execution and if no interleaved operation is allowed?
6. What is serial schedule?
7. What is non-serial schedule?
8. What is conflict schedule?
9. What is locking?
10. Short note on Deadlock.

Long answers type questions

1. List a few database applications where transaction processing is used. List a few different transaction types for each application.
2. How concurrency is achieved using multiprocessing OS? What is the advantage of interleaved processing?
3. What are the problems occur when two transactions are executed in an uncontrolled manner? Give examples and explain.
4. Discuss the actions taken by READ and WRITE operations on a database.
5. Discuss the typical state that a transaction goes through during execution.
6. Discuss the atomicity, durability, isolation and consistency prevention.
7. What is serializable schedule? Why is a serial schedule considered correct? Why is a serializable schedule considered correct?
8. What is lock? Explained the modes of locking.
9. What do you understand by lock compatibility? Discuss the details with examples.
10. Discuss the different types of failures. What is meant by catastrophic failure?

3.14 Suggested Readings

1. Elmasri, Ramez, Navathe, Shamkant B., *Fundamentals of Database Systems,* Pearson Education.
2. Gehrke, Johannes, Ramakrishnan, Raghu, *Database Management Systems,* McGraw-Hill Education.

UNIT 4: Database Security

4.1 Introduction

The use of various tools to protect huge virtual data storage units is known as database security. The field consists of several components, but it is primarily concerned with how to protect user databases from external attacks. Protecting the data itself (data level security), the applications used to process and store data, the physical servers, and even the network connections that allow users to access databases are all areas of database security (system level security). Database security procedures protect the data within the database and the database management system, and all applications that access it from intrusion, data misuse, and damage. It's a large term that refers to various processes, tools, and methodologies used to ensure database security.

Here in this unit, we will discuss the various aspect of database security. You'll be able to understand the fundamental problems that compromise database security. As you'll see, database-driven systems canhave problems at any stage, including during database creation, deployment, and even later. This unit aims to provide a brief overview of database security threats and various challenges. When developing a security system, security models are the most fundamental theoretical tool to use. Security policies, which are governing regulations adopted by any organization, are enforced by these models. Access control models are security models that restrict the activities of authorized users. Discretionary, mandatory, and role-based access controls are the three main types of access control. Every technique has its own set of advantages and disadvantages. The choice of an appropriate access control model is based on the requirements as well as the types of attacks that the system is vulnerable. A strong authentication and authorization strategy helps protect the users and their data from attackers. You will get a brief overview of the data encryption process. Different types of encryption algorithms will discuss at the last of this unit.

4.2 Objectives

After going through this unit, you will be able to:

- understand the needs of Implicit parallelism techniques.
- understand the database security issues
- know the principle of database security
- describe the security model
- describe the various control method of database security
- know different threats and challenges in database security
- know the idea of Multilevel security
- describe different types of access control mechanisms
- distinguish between authentication and authorization
- understand the encryption process and its various types of algorithm
- know what a digital signature is

4.3 Security issues

The technique which helps to give protection and security to the database against some threats (like accidental or intentional threats) is known as database security. Security problems will extend beyond the data in an organization's database: a breach of security may impact other system components, affecting the database structure in the end.Consequently, database security includes software parts, hardware parts, human resources, and data.

Database security refers to the set of tools, procedures, and mechanisms used to ensure the confidentiality, integrity, and availability of a database.

Appropriate controls, which are distinct in a particular mission and purpose for the system, are required to use security efficiently.
Database security can be considered regarding the following situations:

- Loss of data privacy.
- Loss of data integrity.
- Loss of availability of data.
- Loss of confidentiality or secrecy.
- Theft and fraudulent.

The organization should focus on reducing the threat which can be incurring loss or damage to data inside a database from the listed circumstances above. Because all of the data inside an organizationare interrelated, an activity that results in a loss in one area can often result in a loss in another. These scenarios primarily represent areas where the company should concentrate on lowering the risk of data loss or destruction in a database.

4.4 Principles of database security:

We need a security model to organize our ideas on security, which may take many different shapes depending on responsibilities, level of detail, and goal. The most important categories are areas of interest (threats, impact, and loss) as well as the actions involved in dealing with them.

The loss of assets is one example of a security risk.Among these are:

- Hardware
- Software
- Data quality

- Data Credibility
- Data
- Availability
- Business benefit

4.5 Security models

The formal description of security policies is called a security model. A security model offers the environment for database considerations, such as implementation and operation, by establishing external criteria for analyzing security issues in general.Security models for specific DBMSs are significantly essential in system design and operation. Security models describe the elements of a database management system (DBMS) that must be deployed to set up and run actual security solutions. Concepts are embodied, regulations are implemented, and servers are provided for such functions. Any deficiencies in the security model will translate either into insecure operations or inefficient systems. For evaluating and comparing security policies, security models are useful tools. We can use security models to check for completeness and consistency in security policies.

The following elements are used to describe security models:

- **Subjects**: Entities that request access to objects.
- **Objects**: Entities for which subjects are making the access request.
- **Access Modes**: various operations performed by the subject on the object (read, write, create, etc.).
- **Policies**: Enterprise-wide accepted security rules.
- **Authorizations:** Specification of access modes for each subject on each object.
- **Administrative Rights**:Who has rights in system administration, and what responsibilities do administrators have.
- **Axioms**: Basic working assumptions.

4.6 Some common threats in database security

a. **Privilege Elevation**: There are some software flaws that attackers can exploit to elevate their access privileges from a regular user to that of an administrator, resulting in misunderstandings of typical analytical data and funds transfers to fake accounts for specific analytical data.

b. **SQL or nonSQL Injection:**

The insertion of arbitrary SQL or nonSQL attack strings into database queries served by web applications or HTTP headers is a database-specific threat. These attacks are vulnerable to organizations that do not follow secure web application coding practices or conduct regular vulnerability testing.

c. **Excessive Privilege Abuse**: When database users are given various privileges and allowances that go beyond what is required of them, they may be abused for nefarious purposes. For example, if a company user has the ability to change employee residence information, that user could abuse their database update privileges and change someone's salary information.

d. **Legitimate Privilege Abuse**: This occurs when a legitimate database user abuses their rights to access the database for illegal purposes. This is triggered when a system manager or database administrator abuses their authority and engages in any illegal or unethical behavior.

e. **Platform Vulnerabilities**: Platform information describes the operating system in use. Vulnerabilities in operating systems such as Windows 2007, Linux, and Windows XP, as well as the additional services installed on a database server, can result in data corruption, illegal access, or a denial of service. A database system's security measures and protection can be overridden by an operating system's flaws.

f. **Database Communication Protocol Vulnerabilities**:Almost all database retailers' database communication protocols have a significant amount of security flaws. False activities directed at such vulnerabilities can range from unauthorized data access to denial of service and data exploitation, among other things.

g. **Denial of service (DoS/DDoS) attacks**: In a denial of service (DoS) attack, the attacker floods the target server—in this case, the database server—with so many requests that it can no longer fulfill legitimate requests from real users, and the server becomes unstable or crashes in many cases. The deluge comes from multiple servers in a distributed denial of service (DDoS) attack, making it more difficult to stop the attack.

h.**Malware Malware:** Malware Malware is software written specifically to exploit vulnerabilities or otherwise cause damage to the database. Malware may arrive via any endpoint device connecting to the database's network.

4.7 Challenges of database security in DBMS

Given the huge growing number and speed of threats to databases and many other types of information assets, research efforts should focus on data quality, intellectual property rights, and database survival.

1. Data quality –

- To analyze and confirm the quality of data, the database community needs approaches and certain organizational solutions. Straightforward mechanisms such as quality stamps that are displayed on various websites are examples of these approaches. We also require approaches that will enable us to develop more effective integrity semantics verification tools for data quality evaluation, based on a variety of approaches such as record linkage.
- Application-level recovery methods are also required to rectify the erroneous data automatically.
- These challenges are now being addressed by ETL (extract, transform, and load) technologies, which are extensively used for putting data into data warehouses.

2. Intellectual property rights –

As Internet and intranet usage grows, legal and informational issues over data are becoming important problems for many organizations. To address these issues, watermarking is employed, which helps to secure content against unlawful copying and dissemination by granting the owner of the verifiable content power.

They are traditionally reliant on the availability of a broad domain within which the objects can be changed while maintaining their fundamental or vital features.

However, further research is desirable to assess the robustness of many of these strategies, as well as to analyze and explore a wide range of approaches or methodologies targeted at preventing intellectual property rights violations.

3. Database survivability –

Despite disruptive events such as information warfare assaults, database systems must continue to operate and perform their tasks despite decreased capabilities.

A database management system should be able to accomplish the following in addition to making every attempt to avoid and detect attacks:

- **Confident:**
  We must act quickly to remove the attacker's access to the system and isolate or confine the problem to prevent it from spreading further.
- **Damage assessment:**
  Determine the scope of the issue, including any failed functions or data corruption.

- **Recover:**
  To re-establish a normal level of functioning, recover corrupted or loss of data and repair, as well as reinstall, failed functions.
- **Reconfiguration:**
  Reconfigure the operation to allow it to continue in a degraded condition while recovery occurs.
- **Fault treatment:**
  Determine the weakness exploited in the attack to the degree feasible and take actions to avoid a recurrence.

4.8 Control methods of database security

Database security refers to the protection of sensitive data and the prevention of data loss. The Database Administrator is responsible for the database's security (DBA).The following are the primary control mechanisms used to ensure database data security:

1. Authentication

2. Access control

3. Inference control

4. Flow control

5. Database Security applying Statistical Method

6. Encryption

7. RAID Tools

8. Backup and Recovery

These are explained as following below.

1. **Authentication :**

   Authentication is the practice of determining if a user logs in solely with the permissions granted to him to execute database transactions. A user can only login to the extent of his power, but he cannot access any additional sensitive data. Authentication limits the privilege of accessing sensitive information. These biometric authentication technologies, such as retina and figure prints, can protect the database from unauthorized or fraudulent users.

2. **Access Control :**
   Unauthorized users' access to the database must be limited by the security mechanism of the database management system. The DBMS manages access control by generating user accounts and controlling the login procedure. As a result, database access to sensitive data is limited to those people (database users) who are permitted to do so, and unauthorized access is prohibited.During the whole login time, the database system must also maintain track of all actions conducted by a certain user.

3. **Inference Control :**

   This approach is referred to as the statistical database security countermeasures. Its purpose is to prohibit the user from finishing any inference channel. This approach prevents sensitive information from being disclosed inadvertently. There are two sorts of inferences: identity disclosure and attribute disclosure.

4. **Flow Control :**
   This prohibits information from reaching unauthorized users. Covert channels are paths for information to pass implicitly in ways that violate a company's privacy policy.

5. **Database Security applying Statistical Method :**
   The Statistical database security is concerned with protecting personal individual values kept in and utilized for statistical reasons, as well as retrieving value summaries based on categories. They do not allow for the retrieval of personal information.
   This permits access to the database in order to obtain statistical information on the number of employees employed by the company but not to obtain detailed confidential/personal data on a particular individual employee.

6. **Encryption :**
   This technique is mostly used to secure sensitive information (such as credit card numbers and OTP numbers) and other numbers. Some encoding algorithms are used to encode the data.Unauthorized users will have a pretty hard time decoding this encoded data, whereas authorized users are given decoding keys to decrypt data.

7. RAID Tools:
   RAID (Redundant Array of Independent Disks) is a huge disk array that consists of numerous independent disks that are structured to promote reliability while also increasing performance.Data striping (the data is divided into equal-size partitions) boosts performance by distributing it across many disks in a transparent manner. Using a parity scheme or an error-correcting technique, reliability is improved by storing redundant information across disks.

8. **Backup and Recovery:**

Backup is the process of copying the database and log files to offline storage media on a regular basis. A database management system should provide backup capabilities to aid in the recovery of a database in the event of a breakdown. It's usually a good idea to make backup copies of the database and log files on a regular basis and keep them in a secure location. The backup copy and the details saved in the log file are used to restore the database to the most recent feasible, consistent state in the event of a failure that renders it useless.

Journaling is the process of storing and maintaining a log file (or journal) of all changes made to the database in order to successfully recover in the case of a failure. A database management system should include logging capabilities, also known as journaling, that keep track of the current state of transactions and database modifications to aid recovery procedures. The benefit of journaling is that in the event of a failure, a backup copy of the database plus the information in the log file can be used to restore the database to its last known consistent state. If no journaling is configured on a failed system, the only way to recover the database is to restore it from the most up-to-date backup version.

STOP TO CONSIDER

A covert channel is any communication channel that can be used by a process to transfer data in a way that goes against the security policy of the system. In a summary, covert channels transmit data using non-standard methods that are incompatible with the system's design.

Check Your Progress I

1. Why is database security essential?
2. What is a database threat?
3. What are the control measures for database security?
4. What is RAID in DBMS?
5. What role does backup perform in database security?

4.9 Multilevel security

The security policy that allows you to classify objects and users using hierarchical security system levels and a non-hierarchical security system is known as Multilevel security.

Multilevel security prevents unauthorized users fromaccessing information that is classified higher than their authorization level and users from declassifying information.

The following are the various benefits of multilevel security:

1. Multilevel security enforcement is mandatory and automatic.

2. Methods that are difficult to express through traditional SQL views or queries can be used by multilevel security.

3. To provide row-level security control, multilevel security does not rely on special views or database variables.

4. Multilevel security controls are consistent and integrated throughout the system, allowing you to avoid defining users and authorizations multiple times.

5. Users are unable to declassify information due to multilevel security.

Using multilevel security, you can describe security for Db2 objects and perform other checks, such as row-level security checks. Row-level security checks let you control which users have permission to view, modify, or perform other actions on specific rows of data.

Row access control and multilevel security are mutually exclusive. You can enable column access control on a table with a security label column and enforce it on that column, but you can't do the same with row access control. You can't use row access control to enable a security label column in a table. Vice versa is proper; if a table is activated through row access control, you won't be able to add a security label column to it.

4.10 Types of access control

Organizations must decide which access control model to use based on the nature and sensitivity of the data they're processing. Older access approaches include discretionary access control (DAC) and mandatory access control (MAC), but today's most used model is role-based access control (RBAC). Access Control is a permission that allows a user to create or access (that is, read, write, or update) a database object (such as a relation, view, or index), as well as run some DBMS utilities. Offering too many unneeded privileges can compromise

security. A privilege should only be granted to a user if that user would be unable to perform his or her activities without it. Following that, the DBMS keeps track of how these privileges are provided to other users and possibly withdrawn, ensuring that only users with the proper privileges have access to an object at all times.

4.10.1 Mandatory access control (MAC)

It is based on system-wide policies that individual users cannot modify. In this technique, each database object is designated a security class, and each user is given clearance for a security class, and users are subjected to restrictions about database object reading and writing. The DBMS decides whether a user may read or write an object based on a set of rules that include the object's security level and the user's clearance. These rules aim to ensure that sensitive information is never passed on to another user without permission. Support for MAC is not included in the SQL standard.

It's worth noting that most commercial DBMSs only provide mechanisms for discretionary access control at the moment. However, government, military, and intelligence applications, as well as numerous industrial and corporate applications, all require multilevel security. Some DBMS providers, such as Oracle, have issued customized versions of their RDBMSs for government usage that include required access control.

Top secret (TS), secret (S), confidential (C), and unclassified (U) are the most common security levels, with TS being the highest and U being the lowest. Other, more sophisticated security classification methods exist, such as those that organize security classes into a lattice. To keep it simple, we will use the system with four security classification levels, where TS ≥ S ≥ C ≥ U, to illustrate our discussion. The Bell-La Padula model, which is widely used for multilevel security, divides each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of four security classifications: TS, S, C, or U. The clearing (classification) of a subject S will be referred to as class(S), and the classification of an object O will be referred to as class(O) .

Based on the subject/object categories, there are two restrictions on data access that has been enforced:

1. Simple security property:A subject *S* is not allowed read access to an object *O* unless class(*S*) ≥ class(*O*). This restriction is self-explanatory, and it enforces the obvious rule that no subject can read an object with a higher security classification than the subject's security clearance.
2. Star property (or *-property): A subject *S* is not allowed to write an object *O* unless class(*S*) ≤ class(*O*). This restriction is less intuitive in this instance. It prevents a subject from writing an object with a lower security classification than their own. If this rule is broken, information can flow from higher to lower classifications, which is against multilevel security's basic tenet.

A user (subject) with TS clearance can make a copy of a TS-classified object and then write it back as a new U-classified object, making it visible throughout the system.

STOP TO CONSIDER

The major security method for relational database systems has typically been the discretionary access control approach of giving and revoking privileges on relations. This is an all-or-nothing approach: A privilege is either granted or denied to a user. Many applications require an extra security policy that categorizes data and users according to security classes. This method is referred to as mandatory access control (MAC).

4.10.2 Discretionary access control (DAC)

The granting and revoking privileges is a common method of enforcing discretionary access control in a database system. Consider privileges in the context of a relational database management system. In particular, we'll discuss a privileges system similar to the one created for the SQL language. Many current relational DBMSs use this technique in some form or another. The main concept is to include statements in the query language that allow the DBA and certain users to grant and revoke privileges.

Types of Discretionary Privileges

The concept of an authorization identifier is used in SQL2 and later versions to refer to, roughly speaking, a user account (or group of user accounts). In place of an authorization identifier, we'll use the terms user and account interchangeably. The database management system (DBMS) must allow selective access to each database relation based on specific accounts. Because operations can be controlled, having an account does not necessarily entitle the account holder to all of the DBMS's functionality. Informally, there are two levels for assigning database system privileges:

a. **The account level.** The DBA specifies the specific privileges that each account has at this level, which are independent of the database relations.

b. The relation (or table) level. The DBA can manage the privileges to access any particular relation or view in the database at this level.

On R, you have access to references. When establishing integrity constraints, the account now has the ability to reference (or refer to) a relation R. This permission can also be restricted to R's specific attributes.

Note that in order to specify the query that corresponds to the view, the account must have the SELECT permission on all relations included in the view definition.

Specifying Privileges through the Use of Views

The views mechanism is a significant discretionary authorization method in its own. If the owner A of a relation R wants another account B to be able to retrieve only particular fields from R, A can construct a view V of R that only includes those characteristics and then grant SELECT on V to B. The same can be said for limiting B to just retrieving particular tuples from R; a view V can be defined by a query that picks only those tuples from R that A wishes B to have access to.

Revoking of Privileges

It may be necessary to temporarily grant a user a privilege in some instances. For instance, the owner of a relation could want to give a user the SELECT privilege for a specific task and then withdraw it once the work is accomplished. As a result, revocation of privileges requires a mechanism. REVOKE is a SQL command that can be used to revoke privileges.

Propagation of Privileges Using the GRANT OPTION

When the owner A of a relation R gives another account B a privilege on R, the privilege might be given to B with or without the GRANT OPTION. If the GRANT OPTION is specified, B has the ability to grant that privilege to other accounts on R. Assume A grants the GRANT OPTION to B, and B subsequently grants the privilege on R to a third account C, which has the GRANT OPTION as well. Privileges on R can be propagated to other accounts in this manner without the knowledge of R's owner. If the owner account A now revokes the privilege assigned to B, the system should automatically revoke all privileges that B spread based on that privilege.

A user's privileges can come from two or more places. For instance, both A2 and A3 may grant A4 a specific UPDATE R privilege. In this situation, even if A2 revokes A4's privilege, A4 will retain it because it was granted by A3. If A3 later revokes A4's privilege, A4's privilege is completely lost. As a result, a database management system that permits permission propagation must keep note of how each privilege was provided so that privilege revocation may be done accurately and completely.

An Example to Illustrate Granting and Revoking of Privileges:

Assume the DBA creates four accounts: A1, A2, A3, and A4, but only A1 is allowed to create base relations. The DBA must use the GRANT command in SQL to accomplish this:

GRANT CREATETAB TO A1;

The CREATETAB privilege is an **account privilege** that allows account A1 to create new database tables (base relations).

To achieved this DBA issue a CREATE SCHEMA command, as follows:

CREATE SCHEMA EXAMPLE AUTHORIZATION A1;

The user account A1 can now create tables in the EXAMPLE schema. Assume that A1 builds the two base relations EMPLOYEE and DEPARTMENT; A1 becomes the owner of these two relations and has ***all relation privileges*** on each of them.

Assume that account A1 wants account A2 to have the privilege to insert and delete tuples in both of these relations. A1, on the other hand, does not want A2 to be able to extend these rights to other accounts. A1 can issue the following command:

GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;

The owner account A1 of a relation has the GRANT OPTION enabled by default, allowing it to grant privileges on the relation to other accounts. However, because account A2 was not provided the GRANT OPTION in the preceding command, it cannot grant INSERT and DELETE privileges on the EMPLOYEE and DEPARTMENT tables.

Let's say A1 wants account A3 to obtain data from either of the two tables and distribute the SELECT privilege to other accounts. A1 can issue the subsequent command:

GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;

The section WITH GRANT OPTION indicates that A3 can now use GRANT to extend the privilege to other accounts. By performing the following command, A3 can provide A4 the SELECT privilege on the EMPLOYEE relation:

GRANT SELECT ON EMPLOYEE TO A4;

Assume A1 decides to revoke A3's SELECT privilege on the EMPLOYEE relation; A1 can then run the following command:

REVOKE SELECT ON EMPLOYEE FROM A3;

The SELECT privilege on EMPLOYEE from A3 and the SELECT privilege on EMPLOYEE from A4 must now be automatically revoked by the DBMS.This is due to the fact that A3 granted that privilege to A4, yet A3 no longer has it.

Check Your Progress II

6. What are the main differences between DAC and MAC?
7. What are the drawbacks of discretionary access control?

4.10.3 Role-Based access control (RBAC)

RBAC (role-based access control) became popular in the 1990s as a tested method of administering and enforcing security in large-scale enterprise-wide systems. Privileges and other permits are associated with organizational roles rather than individual users, according to the core premise. After that, users are assigned roles that are appropriate for them.

The CREATE ROLE and DESTROY ROLE commands can be used to create and destroy roles. When needed, the GRANT and REVOKE commands can be used to assign and revoke rights from roles as well as individual users. Roles such as sales account manager, purchasing agent, mailroom clerk, department manager, and so on may exist in a company. A number of people can be assigned to each role. The role name receives security privileges that are common to all roles, and anyone assigned to this role acquires those privileges automatically.

RBAC can be used in conjunction with standard discretionary and mandatory access controls to ensure that only approved users who are assigned to specific roles have access to specific data or resources. Users can create sessions during which they can activate a portion of their roles. Each session can have several roles assigned to it, but it only maps to one user or one subject. Many database management systems provide the concept of roles, which can be assigned roles.

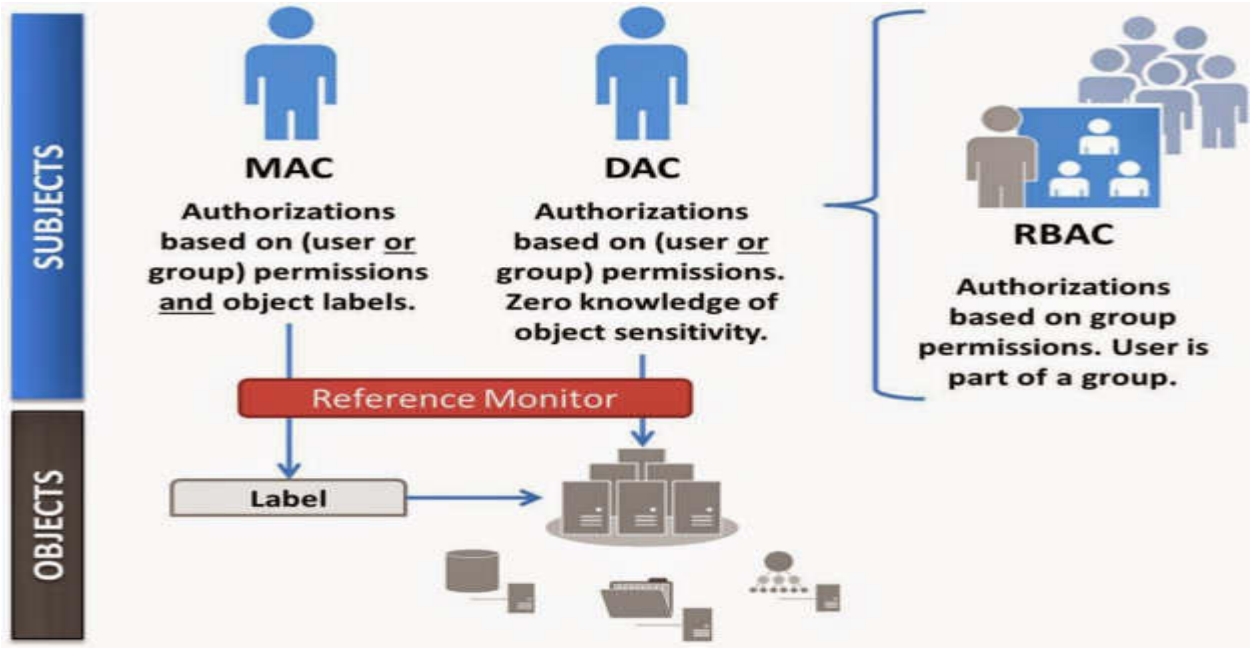


Fig 4.1: MAC vs. DAC vs... RBAC (**Adapted from:[2]**)

4.11 Authentication and Authorization

As users, we are all aware of most systems' login requirements. In most cases, gaining access to IT resources necessitates a secure login process. This subject is about database management system access, and it provides an overview of the procedure from the standpoint of a DBA. The majority of what follows is on Relational Client-Server Systems.

Other system models change to different extents, but the simple principles remain the same.

Authentication

The client must establish the server's identification, and the server must establish the client's identity. To achieve this, shared secrets (i.e., a password/user-id pair or shared biographic and/or biometrics) are often utilized. It can also be accomplished through the use of a system of higher authority that has already established authentication earlier.

Authentication from a peer system may be appropriate in client-server systems when data (not necessarily the database) is disseminated. It's interesting to note that authentication can be passed from one system to the next.As far as the DBMS is concerned, the result is an authorizationidentifier. Authentication does not grant any special permissions for certain operations. It only proves that the DBMS believes the user is who he or she claims to be and that the user believes the DBMS is indeed the intended system.

Authentication is a prerequisite for authorization. Authentication is required before authorization may be granted.

Authorization

Authorization refers to a user's ability to carry out specific transactions, such as changing the database's state (write-item transactions) and/or receiving data from the database (read-item transactions).Authorization, which must be done on a transactional basis, is a vector: Authorization (item, auth-id, operation). A vector is a set of data values that are stored at a certain location in the system.

The DBMS functionality determines how this is implemented. On a logical level, the system structure necessitates the use of an authorization server that works in tandem with an auditing server. As the authorization is transmitted from system to system, there is a problem with amplification and server-to-server security. Amplification here refers to the fact that as the number of DBMS servers involved in the transaction grows, so do the security concerns.

Generally, Audit requirements are routinely applied in an ineffective manner. You must log all accesses and all authorization details with transaction identifiers to be safe. There is a requirement to audit on a regular basis and keep an audit trail, which is typically for a long period.

4.12 Encryption

In certain situations where the DBMS's normal security mechanisms are insufficient, a DBMS can use encryption to protect data. For example, An intruder couldsteal data tapes or tap a communication line. The DBMS ensures that stolen data is not intelligible to the intruder by storing and transmitting it in encrypted form. As a result, encryption is a technique to provide privacy of data.

4.12.1 Data Encryption

Encryption is the process of transforming plaintext to ciphertext, and decryption is the reverse procedure.The plaintext is the unencrypted message in cryptography. A function with a key parameter transforms the plaintext. The ciphertext is the final result of the encryption procedure. The network is then used to send the ciphertext.

The transmitting end performs encryption, while the receiving end performs decryption. The encryption key is required for the encryption process, and the decryption key is required for the decryption process, as indicated in the diagram. Intruders who do not know the decryption

key are unable to convert ciphertext to plaintext. Cryptography is another name for this technique.

The primary idea behind encryption is to utilize a user-specified or DBA-specified encryption key and an encryption algorithm that may be accessible to the intruder to encrypt the original data. The encrypted form of the data is the algorithm's output. A decryption algorithm is also available, which takes the encrypted data and the decryption key as input and returns the original data. The decryption algorithm creates garbage without the proper decryption key. The encryption and decryption keys may or may not be the same., but they must have a secret relationship.

There are the following techniques used for the encryption process:

• **Substitution Ciphers**: To mask each letter or group of letters, a substitution cipher replaces them with another letter or group of letters. For instance, A is replaced by D, B by E, C by F, and Z by C. As a result, the attack becomes. Because an intruder can readily predict the substitution characters, substitution ciphers are not very safe.

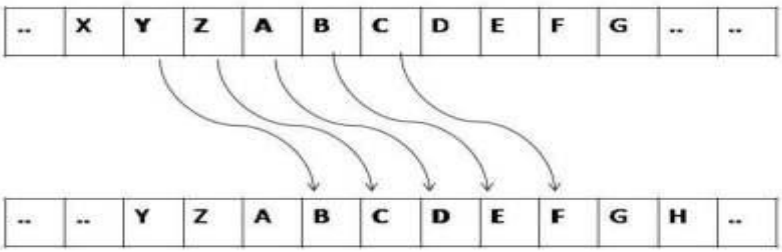


Fig 4.2: Example of Substitution Ciphers

• **Transposition Ciphers**: Substitution ciphers keep the plaintext symbols' order but hide them. The transposition cipher, on the other hand, rearranges the characters without masking them. A key is used in this process. For instance, A may be coded as B. When compared to substitution ciphers, transposition ciphers are more secure.

Plaintext (M) : WELCOME TO IDOL GAUHATI UNIVERSITY

KEY: 632415

Ciphertext (C) : ODHIT LOAUS ETGIR CIUNI MOAVY WELTE

Column out	6	3	2	4	1	5
	W	E	L	C	O	M
	E	T	O	I	D	O
	L	G	A	U	H	A
	T	I	U	N	I	V
	E	R	S	I	T	Y

Fig 4.3:Example of Transposition Ciphers

4.12.2 Algorithms for Encryption Process

For the encryption process, there are several widely used algorithms. The following are examples of these

a. **Data Encryption Standard (DES):**On the basis of an encryption key, it performs both character substitution and order rearrangement. The main flaw in this approach is that the encryption key must be communicated to authorized users, and the mechanism for doing so is vulnerable to clever intruders.

b. **Public Key Encryption:**

In recent years, a method of encryption known as public-key encryption has grown in popularity. Rivest, Shamir, and Adheman proposed the RSA encryption scheme, which is a well-known example of public-key encryption. Every authorized user has a public encryption key that is known to everyone, as well as a private decryption key (which is used by the decryption algorithm) that is chosen by the user and is only known to him or her. The encryption and decryption algorithms are assumed to be known to the general public

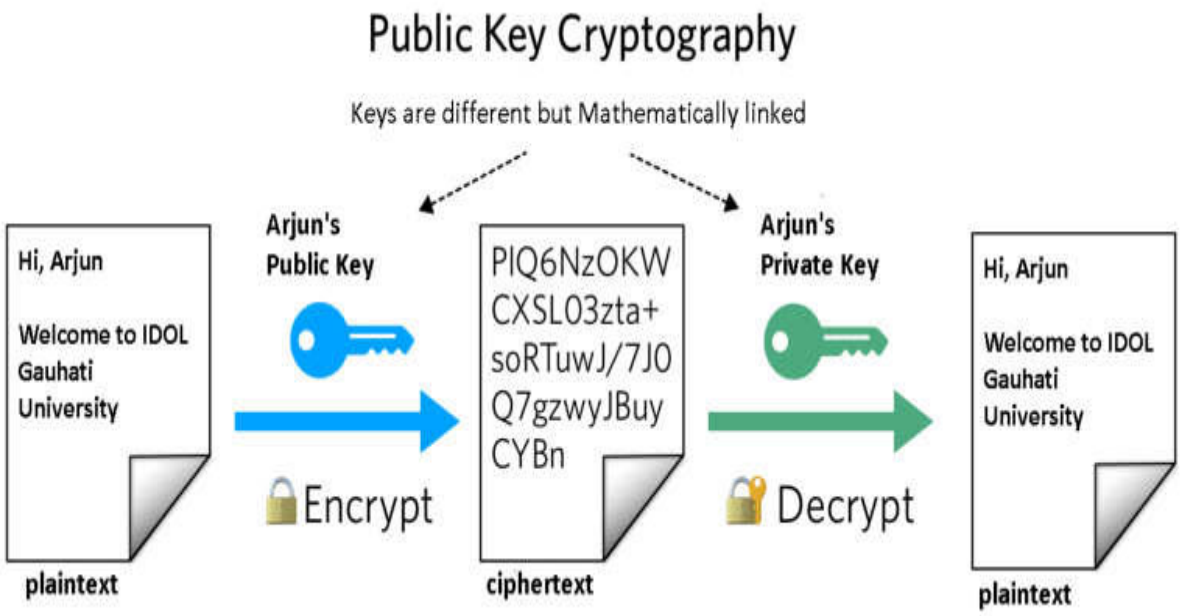


Fig 4.4: Public Key Encryption process

The following are the most important characteristics of a public key encryption scheme: −

- For encryption and decryption, different keys are used. This is a feature that distinguishes this scheme from symmetric encryption schemes.
- Each receiver has his own decryption key, which is commonly referred to as his private key.
- The receiver must publish his public key, which is an encryption key.
- To avoid spoofing by an adversary as the receiver, some assurance of the authenticity of a public key is required in this scheme. This type of cryptosystem typically involves a trusted third party that certifies that a specific public key belongs to a single person or entity.
- The encryption algorithm is complicated enough that an attacker will be unable to deduce the plaintext from the ciphertext and the encryption (public) key.
- Despite the fact that private and public keys are mathematically related, calculating the private key from the public key is not possible. In fact, designing a relationship between two keys is an intelligent part of any public-key cryptosystem.

4.12.3 Disadvantages of encryption

There are the following challenges of Encryption:

1. The management of keys (i.e., keeping them hidden) is a problem. The decryption key must be kept secret even in public-key encryption.

2.Even in an encrypted system, data must be processed in plaintext regularly. As a result, transaction programs may still have access to sensitive data.

3.At the level of physical storage organization, encrypting data causes severe technical issues. Indexing data that is stored in encrypted form, for example, can be extremely difficult.

STOP TO CONSIDER

The sender and receiver of a message in a **symmetric cryptography** system share a single, common key that is used to encrypt and decrypt the message. This is a simple system to set up, and both the sender and the receiver can encrypt and decrypt messages.

Asymmetric cryptography, also known as public-key cryptography, is a system in which the sender and receiver of a message use a pair of cryptographic keys to encrypt and decrypt the message – a public key and a private key. This is a complicated system in which the sender can encrypt the message with his key but cannot decrypt it. The receiver, on the other hand, can decrypt but not encrypt the message using his key.

4.13 Digital Signature

In e-commerce applications, a Digital Signature (DS) is an authentication technique based on public-key cryptography. It assigns an individual a distinct mark within the body of his message. This allows others to authenticate that message senders are genuine.

To protect against fraud and theft, a user's digital signature is typically different from message to message. The method is as follows −

1. The sender takes a message and calculates the message digest before signing it with a private key.
2. After that, the sender appends the signed digest to the plaintext message.
3. The communication channel is used to send the message.
4. The receiver eliminates the appended signed digest and verifies it with the public key associated with it.
5. The receiver then applies the same message digest algorithm to the plaintext message.
6. If the results of steps 4 and 5 matches, the receiver can be confident that the message is genuine.

CHECK YOUR PROGRESS III

8. Is authentication required for authorization?

9.What is plaintext and ciphertext?

10. What do you mean by public key encryption?

4.14 SUMMING UP

In this unit, we learned about database security, which is protecting a database against unauthorized access, malicious destruction, and even accidental loss or misuse. Any condition or incident, whether intentional or unintentional that will have an impact on a system or organization is considered a danger. This unit has been focusing on several forms of threats. A variety of DBMS security issues have been discussed. Authorization, access controls, views, backup and recovery, encryption, and RAID technology are all computer-based security controls for multi-user environments.A DBMS's backup is essential for recovering the database in the event of failure or damage. Unauthorized users' access to the database must be restricted by the security mechanism of the database management system. The DBMS manages access control by creating user accounts and controlling the login procedure. As a result, database access to sensitive data is limited to those (database users) permitted to do so, and unauthorized access is prohibited. The three primary types of access controls are discretionary, mandatory, and role-based access controls. Each method has its own set of benefits and drawbacks. Proxy servers, firewalls, digital signatures, and digital certificates are some of the security features related to DBMS on the web. A digital signature could be used to confirm that the intended recipient sent the data. Encryption is the process of encoding data with a particular method that makes it unreadable by any computer that does not have the decryption key.The Data Encryption Standard is a single-key encryption method that requires the confidentiality of the encryption key.A public key, as well as a private key, are used in the public key encryption technique.The public key is openly accessible, whereas the authorized user selects and keeps the private key private. In this unit, we discussed how an authentication and authorization strategy might help protect users and their data from attacks.

4.15 KEY TERMS

Database Security: Database security refers to the set of tools, procedures, and mechanisms used to ensure the confidentiality, integrity, and availability of a database.
Authentication: The process of ascertaining whether someone or something is who or what it claims to be is known as authentication.
Authorization: The practice of granting someone permission to do or have something is known as authorization.

Encryption: This is the process of encoding data with a special algorithm that makes it unreadable by any program that doesn't have the decryption key.

Access control: In a computing environment, access control is a security technique that regulates who or what can view or use resources. It is a basic security concept that reduces the risk to the company or organization

Discretionary access control: Discretionary access control (DAC) is a type of security access control that enables or restricts object access based on an access policy set by the owner group and/or subjects of the object.

Mandatory access control: Mandatory access control (MAC) is a system-enforced access control method that enforces security policy through clearances and labels.

Digital signature: A digital signature could be used to confirm that the data was sent by the intended recipient. It is made up of two pieces of data: a string of bits computed from the data being signed using signature algorithms and the private key or password of the person wishing to sign the document.

Ciphertext: Ciphertext is plaintext that has been encrypted using an encryption algorithm. Ciphertext cannot be read until it has been decrypted (converted to plaintext).

4.16 Answers to Check Your Progress

1. It's important to protect the data that any company collects and manages. Database security can protect your database from being hacked, which can result in financial loss, reputational damage, consumer distrust, brand erosion, and non-compliance with government and industry regulations.
2. A database threat is an object, person, or other entity that poses a risk of sensitive data loss or corruption to an asset.

3. These various security controls aid in the management of security protocol circumvention.

i. System hardening and monitoring.
ii. DBMS configuration
iii. Access
iv. Database auditing
v. Authentication
vi. Encryption
vii. Backups

4. Redundant Array of Independent Disks (RAID) is a technology that combines multiple small, low-cost disc drives into an array of disk drives that outperforms a single large, expensive drive (SLED). Redundant Array of Inexpensive Disks is another name for RAID.

5.In data management, making backups of collected data is critical. Human error, hardware failure, virus attacks, power outages, and natural disasters are all protected by backups. If these failures occur, backups can help save time and money.

6.The main difference between DAC and MAC is that DAC is an access control method in which the resource owner determines access, whereas MAC is an access control method in which access is granted based on the user's clearance level.

7. DAC is simple to use and understand, but it does have some drawbacks, such as:

- Inherent vulnerabilities (Trojan horse)
- Grant and revoke permissions maintenance.
- ACL maintenance or capability.
- Limited negative authorization power.

8. Authentication is a prerequisite to authorization because it allows for the secure validation of the subject's identity. After the authentication process is completed, authorization policies begin. What data you can access is determined by the authorization process.

9. The input to an encryption algorithm is plaintext. The unreadable output of an encryption algorithm is known as ciphertext.

10.The term "public-key encryption" refers to a type of encryption that implements two keys. There are two types of keys: a public key that everyone knows and a private key that only you know.

4.17 POSSIBLE QUESTIONS

Multiple Choice Questions:

1. What is true about data security?
   a. Data security is the protection of programs and data in computers and communication systems against unauthorized access
   b. It refers to the right of individuals or organizations to deny or restrict the collection and use of information
   c. Data security requires system managers to reduce unauthorized access to the systems by building physical arrangements and software checks.
   d. All of the above

2. which statement is used to revoke an authorization,
   a. Alter
   b. Modify
   c. Revoke
   d. All of these
3. In _____ attacks, the attacker manages to get an application to execute an SQL query created by the attacker.
   a. SQL injection
   b. SQL
   c. Direct
   d. Application
4. By data integrity,we mean
   a. maintaining consistent data values
   b. integrated data values
   c. banning improper access to data
   d. not leaking data values
5. Data integrity is ensured by
   a.good data editing
   b. propagating data changes to all data items
   c. preventing unauthorized access
   d. preventing data duplication
6. By data security in DBMS, we mean
   a. preventing access to data
   b. allowing access to data only to authorized users
   c. preventing changing data
   d. introducing integrity constraints
7. DBMS must implement management controls to
   (i) control access rights to users

(ii) implement audit trail when changes are made
(iii) allow data to be used extensively in the organization
(iv) duplicate databases

a. i, ii
b. ii, iii
c. iii, iv
d. i, iv

8. A database administrator
   a. administers data in an organization
   b. controls all inputs and all outputs of programs
   c. is controller of data resources of an organization
   d. controls all data entry operators
9. Access right to a database is controlled by
   a. top management
   b. system designer
   c. system analyst
   d. database administrator
10. Encrypted data is commonly referred to as
   a. ciphertext
   b. plaintext
   c. information
   d. raw data

Answers:- (1. a, 2. c, 3. a, 4. a, 5.b, 6. b, 7.a , 8.c, 9.d, 10.a)

Fill in the blanks:

1. _____ is the act of granting someone permission to do or have something.

2. _____is a system-enforced access control method that uses clearances and labels to enforce security policies.

3. A private key is a variable in _____ that is used with an algorithm to encrypt and decrypt data.

4. The set of tools, procedures, and mechanisms used to ensure the confidentiality, integrity, and availability of a database is referred to as_____.

5. The Data Encryption Standard (DES) is a symmetric-key algorithm for digital_____.

6. A _____is an individual or person responsible for controlling, maintenance, coordinating, and operation of the database management system.

7. _____channels are paths for information to pass implicitly in ways that violate a company's privacy policy.

Answers:-
1. Authorization 2. Mandatory access control (MAC) 3. cryptography 4. database security 5. data encryption6. Database Administrator (DBA) 7. Covert

Short answer type questions:

1. What do you mean by database security?
2. What is the need fordatabase security?
3. What is the principle of database security?
4. Mention the threat of database security in DBMS.
5. What are various control methods used to secure DBMS?
6. Define RAID technology.
7. Define flow control.
8. What is SQL injection?
9. What is the access control method?
10. Describe Authentication and Authorization.
11. Define ciphertext and plaintext.
12. Why is backup essential in DBMS?
13. What is the Mandatory access control method?
14. Differentiate between MAC and DAC.
15. Write a short note on Role-Based Access Control?
16. What is Data Encryption Standard(DES)?
17. What is a digital signature?
18. What do Database Encryption and Decryption mean?

Long answer type questions:

1. Explain the needs of database security
2. Describe the different threats to database security.
3. Discuss various challenges in database security.
4. Describe the access control method.
5. Explain multilevel security in DBMS.
6. Explain various types of access control methods.
7. Describe the Mandatory access control methods in DBMS.
8. Describe the Discretionary access control method with an example.
9. Explain the Role-Based Access Control (RBAC) method.
10. Describe the data encryption process.
11. Explain how digital signature works.

4.18 REFERENCES&FURTHER READINGS

[1]Chapter 12. Database Security. ttps://www.cs.uct.ac.za/mit_notes/database/pdfs/chp12.pdf

[2]Cloud Audit Controls: MAC vs. DAC vs. RBAC.
http://www.cloudauditcontrols.com/2014/09/mac-vs-dac-vs-rbac.html.

[3] Discretionary Access Control Based on Granting and
https://www.brainkart.com/article/Discretionary-Access-Control-Based-on-Granting-and-Revoking-Privileges_11580/

[4]ElamasriR . and Navathe, S., Fundamentals of Database Systems (3 rd Edition), Pearson Education, 2000.

[5] Database Management Systems, Raghurama Krishnan, Johannes Gehrke, TATA McGraw Hill 3rd Edition.

***

UNIT-I: OBJECT ORIENTED DATABASE SYSTEM

Unit Structure

1.1Introduction

In the earlier chapters, learners have been acquainted with some advanced form of traditional database concepts. In this chapter, the learners are going to be acquainted with a new dimension or extension area of the existing traditional database theory. All the data are imagined or visualized as some objects in this new area of discussion in addition to the relational form of existing database management system. The data here are stored, manipulated and accessed as objects, which is done in object oriented programming paradigms. The concept of object can be realized by defining one class with underlying characteristics like data abstraction, information hiding, encapsulation and imposing on it other object oriented features like inheritance, polymorphism, early binding and late binding. The idea of object oriented database approach comes into existence because of the acceptance of object oriented

programming approach among wide range of users worldwide. Some object databases, accepted widely and appreciated by the database community are mentioned in this unit. The required standards in the design of the object oriented database systems and the associating query languages needs to be discussed in order to have a detailed insight into it. The relational database system is the basis on which the OORDBMS approach is evolving. The history of object relational database system is covered, which is followed up by its detailed description. The object oriented relational database management approach is compared with the classic relational database management approach as the conclusive topic.

1.2 Unit Objectives

After going through this unit, you will be able to:

- Learn the object-oriented database system.
- Lean the need of object-oriented approach in databases.
- Learn the advantages and disadvantages of object-oriented databases.
- Accustom with some popular object databases.
- Learn the standards, languages and design issues of object based databases.
- Understand the history of object-oriented relational database system.
- Understand how object-oriented relational database system works.
- Compare RDBMS and OORDBMS.

1.3 Concepts of Object-Oriented Databases

A database is generally considered as data storage. This storage is further used for the purpose of searching, editing or updating, generating reports etc. Data storages can further be classified in four widely spelled categories viz., Traditional File System, Relational DBMS, Object Oriented DBMS and Object-Oriented Relational DBMS. These categories are classified on looking into the pattern of data.

The object-oriented paradigm is the basis upon which the object-oriented database is designed. The data or the information in the object-oriented database is represented and stored in the form of certain objects. The object-oriented database is also known as object database and is handled though object-oriented database management systems (OODBMS).

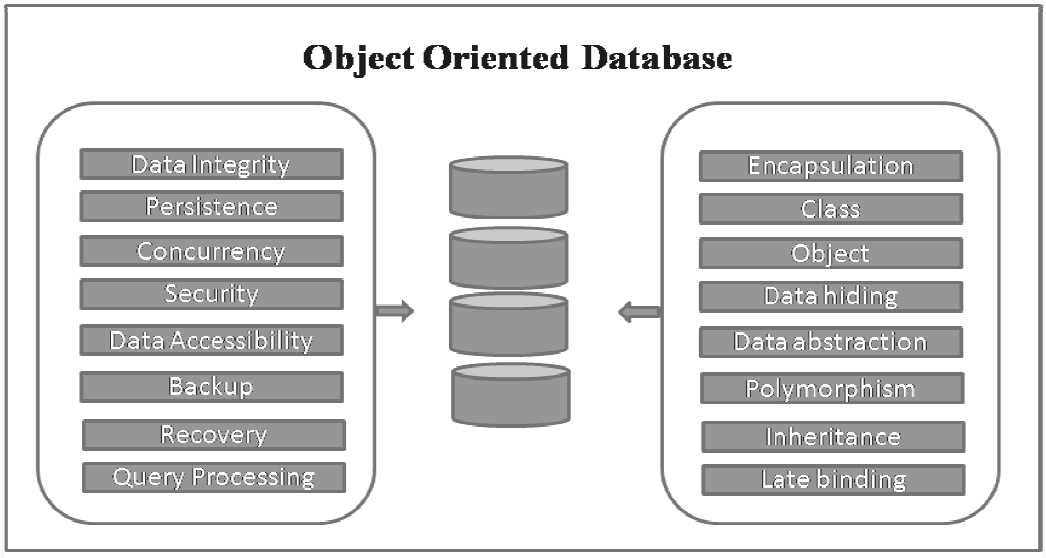


Fig-1.1: Conceptual bock diagram for OODBMS

The OODBMS encompasses the conventional DBMS features as well as the object oriented features together. The conventional DBMS features are like data integrity, persistence, concurrency, security, backup, recovery query processing etc., while the object oriented features are encapsulation, class, object, overloading, overriding, inheritance, early binding, late binding etc. Some of the popularly known object oriented programming (OOP) languages are C++, Java, Perl, Ruby, Python and Java-script. Object-oriented databases are administered through the object database management systems (ODBMS).The preparatory idea of object oriented databases immerged in the late nineties of the nineteenth century and currently it has become common for various OOP based languages, such as C++, Java, Smalltalk and LISP. For example, Smalltalk is used in GemStone, LISP is used in Gbase, and COP is used in Vbase and so on.

Objects are composed of some data members and member functions or methods, which are encapsulated within a single unit with individual values and certain properties. Objects come into existence by instantiation of certain user defined classes. Objects generally go through a cycle that includes the creation or allocation of objects, use of the objects and the deletion or de-allocation of objects. Object databases are common among many modern high performances applications with high speed data access and manipulative facilities. Some of the significant areas where object databases are taking a pivotal role are the real-time systems, architectural engineering for 3-D modeling, telecommunications, robotics, molecular science, astronomy and many more.

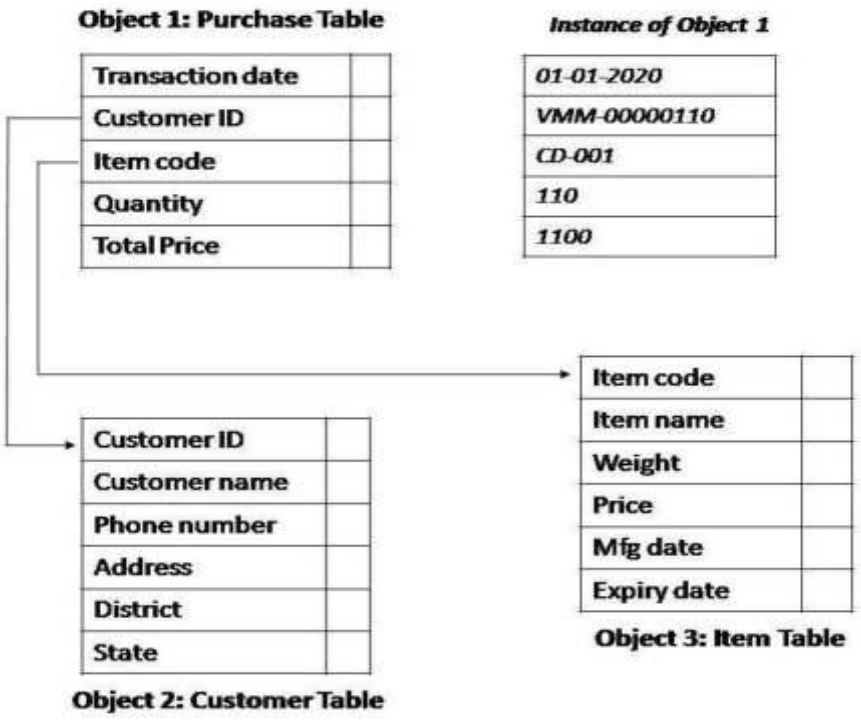


Fig-1.2: Object Oriented Model in OODBMS

1.3.1 Key Features of Object Databases

- Object oriented databases or Object-oriented DBMS systems provide persistent storage to objects.
- It is capable of storing and reading data and converting the same into program objects for further storing of reading data & loading object based data in memory.
- Object databases bring permanent persistence to objects.
- The reading and mapping of the data of an object database to the objects is direct without any API like tool.

Object databases facilitate quick access of data or information and better performance inevitably. There are some object based databases with multi-lingual supports too. For example, Gemstone is such an object database that supports C++, Smalltalk and Java programming languages.

1.3.2 Drawbacks of Object Databases

- Object databases are still not popular among vast community of database users as compared to RDBMS.
- Developers are less in numbers in handling of object databases.

- Not many programming language support object databases.
- RDBMS have SQL as a standard query language. Object databases do not have such a standard.
- Object databases are difficult to learn for non-programmers

1.3.3 Popular Object Databases

Following are some of the popular object databases. These databases are accepted by most database users because of the highly flexible features that conform to the needs of current users. The descriptions of few such databases are mentioned below.

Cache

Cache is developed by Inter Systems and it is a high-performing object database. This object based database facilitates a set of services that include data storage, concurrency management and handles diverse transactions issues and process management activities. Cache engine can be treated as full-fledged powerful database toolkit with extensive relational database features. This database can be used for diverse queries and modification purposes using standard SQL via ODBC, JDBC or object based methods. The computational efficiency of Cache is enormous and it is a most reliable relational database with high scalability parameters. Some of the important features of Cache database are mentioned below.

- Able to model data as objects, while eliminating mismatch between databases and object-oriented applications.
- Supports user-defined data types.
- The ability to take the advantage of methods and inheritance like functions.
- Object-extensions for SQL to handle object identity and relationships.
- The ability to avail SQL and object-based access through a single application.
- Clustering is used to store data ensuring maximum performance.

Concept Base

Concept Base is another database system with multi-user and object-oriented support which is deductive in nature. It is a powerful tool for meta-modeling and is very useful for customizing modeling languages. Concept Base comes with an associating graphical user interface (GUI) facilitating the users with some common routines. Concept Base is developed by the Concept Base Team at University of Skövde (HIS) and the University of Aachen (RWTH). Commonly available operating systems like Linux, Windows and Mac support Concept Base. There is also a pre-configured virtual application within Concep Base, which contains associating executable files and source files along with the tools for compiling. The system is distributed under a FreeBSD-style license.

Object DB

Object DB is a powerful object-oriented database management system (ODBMS) based on Java language. It is a compact but reliable system, which is easy to use and extremely fast in terms of object database access. It supports both the client-server mode and the embedded mode. Object DB provides all the standard database management services. This is the reason, why the development process gets easier and the applications behave faster. It is capable of handling advanced level queries and providing enhanced indexing facilities. It is very much effective in multi-user environments, where there is always a rush of users. Object DB can easily be embedded in any applications irrespective of its sizes and types.This is such a database, which has been tested with Tomcat, Jetty, GlassFish, JBoss and Spring.

Several other popular object based databases are Object Database++, GemStone/S, Perst, ZODB, Wakanda, ODABA, Objectivity/DB. The discussions on these object databases is beyond the scope of this syllabus. The learners can use various internet sources to gather a detailed knowledge on these object based databases.

1.4Standards, Languages and Design

There should always be a standard agreed upon by all vendors of a particular type of database system. A standard can be resembled with an agreed roadmap maintaining uniformity among all stakeholders to proceed through a common model.

1.4.1 Standards and Languages

Some of the sound reasons for the need of standards are as follows.

- Standard provides support in maintaining the portability of database applications. Portability is defined as the capability to execute particular software or application on different platforms with minimal modifications.
- Standard shelpin achieving interoperability. Interoperability refers to the ability of an application to access multiple systems. Here, the same application program may access some data stored under one ODBMS package, and another data stored under another source or package.
- Standard allows customers to compare commercial products of various vendors more easily by determining which parts of the standards are applied in their purchased product.

ODMG (Object Data Management Group) is an association for monitoring the object oriented database management activities. This association proposed a standard for ODBMS in the year 1993 and it was named as ODMG 1.0 followed by ODMG 2.0 in 1995 and ODMG 3.0 in 2000.

The ODMG 3.0 standard has the following major specifications:

- Object Model
- Object Definition Language (ODL)
- Object Query Language (OQL)
- C++ Language Binding
- Smalltalk Language Binding
- Java Language Binding

1.4.1.1Object Model

The object model specifies the ODMS based constructs. The basic building blocks of the object model are –*objects* and *literals*.

An object is referred to as the instance of its class type. The *state of an object* is composed of the values that the object carries for a certain set of properties. On the other hand, *the behavior of an object* is defined by the set of operations executed by the objects.

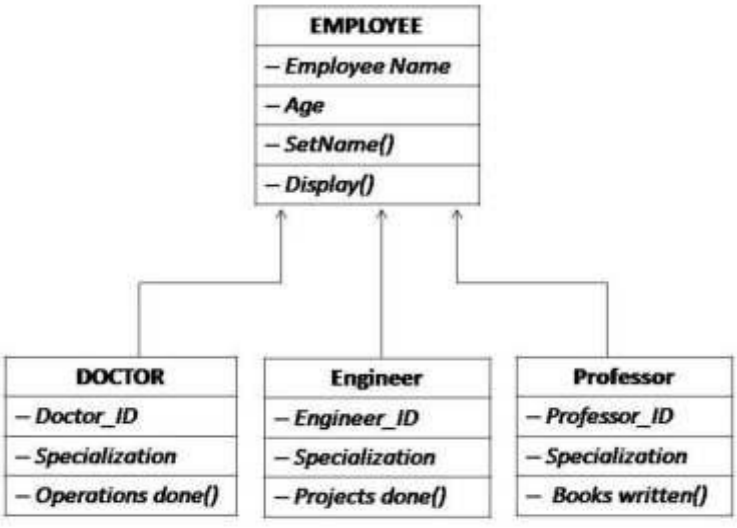


Fig-1.3: Hierarchy of classes/objects in OODBMS

An object is described with some associating parameters i.e. identifier, name, lifetime and structure. The details of these parameters are mentioned below.

Object Identifier: An object can be differentiated from all other nearby objects within its storage domain by using the object identifier. The objects always preserve the same object identifier in its lifetime during execution of a computer program. Thus, the value of an object's identifier never changes. The object remains the same, even if its attributes or the relationship values change. Object identifiers are generated by the OODBMS, but not by the other applications.

Object Name: In addition to the object identifier, the OODBMS may assign one or more names to the objects that are meaningful for the programmers or the end users. The system can refer to an object by its object name. It applies certain mapping functions to determine the object identifiers and locate the desired object.

Object Lifetime: The lifetime of an object is another crucial issue to be addressed. Object lifetime determines the extant of memory or the storage time allowed to the object. Two variants of lifetime for the object are supported in the object based models. They are *transient* and *persistent*. An object, whose lifetime is transient, is allocated a memory space to be managed by the program's runtime system. When the process terminates, the memory is de-allocated. On the other extreme, an object, whose lifetime is persistent, is allocated memory space to be managed by the OODBMS runtime system. This kind of objects exists in memory after the termination of the process initiated by the application program. So, it has a long lifetime as compared to transient form.

Object Structure: The structure of an object can be either atomic or non-atomic (if the object is composed of other objects). The atomic object referred here is user-defined in nature. There is no built-in atomic object type included in OODBMS object models.

Some other important definitions useful for the demonstration of an object model are stated in the following section. The terms used here are *class*, *interface*, *struct*, *literals* and *various literal types*.

- *A class* defines both the abstract state and the abstract behaviour of the object.
- The *interface* defines only the abstract behavior of some objects.
- The *struct* defines the abstract state of some *literals*.
- *A literal* has no identifier and cannot act alone. The *literals* are embedded in objects and cannot be individually referenced. Objects and literals can be

classified by their types. Although, all the elements of a given type have a common range of states and behaviors, a literal defines only the abstract state of a literal type. The value of literals does not change. Few examples of literal values are 67, 17.161576, 'P', 'Q', "GUIDOL" and "August-15, 2021".These examples are some constant numbers, characters and strings.

- In addition to the *struct* definition and the *primitive literal data types* (*boolean*, *char, short, long, float, double, string*), object definition languages define declarations for user-defined *collection, union and enumeration literal types*.

- Three literal types are supported by the object models. They are *atomic*, *collection* and *structured* literals.

    - **Atomic literals** correspond to the values of basic data types. Various numeric numbers, characters, Boolean values etc. are the examples of atomic literal types.
    - **Collection literals** are typically found in the ODMG object models that support literals of the following types: set<t>, bag<t>, list<t>, array<t>, dictionary<t, v> where t is a type of objects or values in the collection.
    - **Structured literals** correspond to the values constructed by tuple constructor. They include the date, time, interval and timestamp as built-in structures and any other user defined structures.

1.4.1.2 Object Definition Language (ODL)

ODL is a specific kind of a language that specifies the structure of databases in object-oriented terms. ODL is an extension of Interface Description Language (IDL), which is again a component of CORBA (Common Object Request Broker Architecture). CORBA is a standard for distributed, object-oriented computing which will be discussed in the later chapters. The ODL is basically a specification language or a design language, which is used to define the specifications of object types that obey the rules of ODMG object model. This can be used like the E/R diagram used in the case of RDBMS platform. ODL is independent of any programming language and it is not used for database manipulation activities.

1.4.1.3 Object Query Language (OQL)

OQL is a query language preferred by object data management group (ODMG) for object oriented database management purpose. OQL works closely with programming languages like C++. The embedded OQL statements within a host language return compatible objects useful for further processing. OQL's syntax is similar to SQL with additional features for object handling. This query language is designed to operate on

databases described through ODL. Unlike SQL, which produces collection, OQL produces collections (sets, bags, lists) of objects. OQL fits naturally in object oriented host languages. Returned objects are assigned in the variables present in the host program and these variables are then used for further programming based manipulative works.

1.4.1.4 C++ Language Binding

Binding of ODMG implementations to C++intends at the writing of portable C++ codes that manipulates persistent objects. This object manipulative language of C++ is abbreviated as OML. The C++ language binding includes a version of the ODL that uses C++ syntax, OQL invoking interface and some procedures for operations on OODBMS prescribed transactions.

1.4.1.5 Smalltalk Language Binding

Binding of ODMG implementations to Smalltalk focuses on the binding in terms of the mapping between ODL and Smalltalk. The Smalltalk bindings also include a mechanism to invoke OQL and required procedures for operations on databases and other transactions.

1.4.1.6 Java Language Binding

The binding between the ODMG Object Model (ODLs and OMLs) and the Java programming language is defined here. The Java language binding includes some mechanism allowing the invoking of the desired OQL and procedures for operations on ODMSs and transactions.

1.4.2 Object Database and Relational Database Design

Whenever we discuss the differences between object database designs (ODB) and relational database designs (RDB), the handling of the relationships issue takes a major role.

In relational database designs, the relationships among the tuples or records are specified by the attributes with matching values. These can be termed as value references and is specified through the *foreign key* concept. Foreign keys are the values of primary key attributes in tuples of the referencing relation or table. The primary keys are limited to being atomic in nature in each record.

In object database design, the relationship issue is handled by reference attributes that include object identifiers (OIDs) of the related objects. In object database design, both single references as well as collection of references are allowed. Another notable and influencing difference between ODB and RDB design is how the inheritance is handled. These mentioned structures are built into the model, so that the mapping is achieved by using the inheritance constructs. Inheritance can be achieved through derived (:) and extends constructs. In relational design, there are several options to choose, because there is no built-in constructs for inheritance in the classic version of relational design. It is necessary to specify the operations early on in the design since they are part of the class specifications. It is an important matter to specify the operations needed during the design phase for all types of databases. But, it may be delayed in RDB design, because it is not mandatory until the implementation phase comes in force. One can easily observe one realistic difference between the relational model and the object model of data in terms of behavioral specifications. Although relational data models do not compel or encourage the database designers to set some valid behaviors or operations, this is an implicit requirement in the case of object models.

1.4.2.1 Mapping of an Enhanced Entity Relationship (EER) Schema into an Object Database (ODB) Schema

The correlation of EER schemas and ODB schemas is simple, because the ODB schemas provide support for inheritance. Once the mapping has been completed, the operations need to be added to ODB schemas. It is because the EER schemas do not include any operations like ODB. The mapping of EER into ODB schemas can be exhibited using the following steps.

Step -1

- Creation of an ODL class for each EER type.
- Multi-valued attributes are declared by sets, bags or lists.
- Composite attributes are mapped into tuple constructors.

Step – 2

- Add reference attributes for each binary relationship into the ODL classes that participate in the relationship.
- Relationship cardinality is set as single-valued for 1:1 and N:1 types and set-valued for 1:N and M:N types.
- Relationship attributes are created through the use of tuple constructors.

Step - 3

- Include the operations corresponding to each class.
- EER schema does not provide these operations and it must be added to the database design by choosing it from the original requirements.
- The associating constructor and destructor operations must also be included.

Step - 4

- Inheritance relationships can be specified via *extends* clause.
- An ODL class that corresponds to a sub-class in the EER schemas inherits the types and methods of its baser-class in the ODL schemas.
- Its non-inherited attributes, relationship references and operations are specified as mentioned in the earlier steps.

Step - 5

- Weak entities can also be mapped in the same way as the regular entity types.
- Non-participating weak entities in any relationships may alternatively be presented as composite multi-valued attribute of the owner entity.
- The attributes of the weak entity are included in the struct<... > construct.

Step - 6

- Map categories (union types) to object definition language.
- May follow the same mapping used for EER-to-relational mapping.
- Declare a class to represent the category.
- Define the 1:1 relationships between the category and each of its base-classes.

Step – 7

- Map multi dimensional cardinality relationships whose degree is greater than 2.
- Each relationship is mapped into a separate class with appropriate reference to each participating class.

1.5 Object Relational Database Systems

Object–relational database systems are commonly termed as Object–relational database management systems (OORDBMS). OORDBMS is an object oriented version of the traditional relational database management systems (RDBMS). This is a kind of a hybrid approach capable of handling the object oriented as well as relational aspects of DBMS, which well fits with the current industry requirements.

1.5.1 Relational Database Management System (RDBMS)

RDBMS is a simply the relational version of traditional DBMS, which incorporates the terms relations, tables, attribute, columns, integrity, security etc. into its operational procedures. RDBMS deals with a number of tables together to store, edit, update and delete data considering the normalization aspects like 1NF, 2NF, 3NF and BCNF forms. Standard SQL statements are used to operate on RDBMS. Various commonly used RDBMSs are Oracle, Microsoft's SQL server, MySQL etc. Although, most of the needs of a common database user are addressed by these softwares, the object oriented aspects could not be incorporated here. This is the reason why the object oriented relational database management system is becoming the need of the hour. The later section is going to elaborate these aspects followed up by sating the differences between OODBMS and RDBMS.

1.5.2 History of Object Relational Database System

The Object–relational database system or OORDBMS came into light in the early 1990s. This trend comes into existence by extending the relational database concepts with the addition of the concept of *object*. The industry experts aimed to get hold on a declarative query-language based upon predicate calculus as a vital component of OORDBMS. Two most notable research projects viz., Illustra and PostgreSQL was brought into reality by Postgres (UC Berkeley) during this time. In the mid of 1990s, early commercial available products were released. These releases include various products like *Illustra* (IBM), *Omniscience* (Oracle)and *UniSQL* (KCOMS). The Ukrainian developer Ruslan Zasukhin, who is the founder of Paradigma Software, Inc. developed and released the first version of Valentina database in the mid of 1990s, which was used as C++ SDK. After less than a decade of time, PostgreSQL had become a commercially available database and has become the basis for several currently available products incorporating OORDBMS features. The experts in the domain started referring these products as *object oriented relational database management systems* or OORDBMS. Many of the ideas of early object relational database efforts have largely been incorporated into SQL:1999 via specific structured types. For example, IBM's DB2, Oracle database, and Microsoft's SQL Server are claiming to support most OORDBMS requirements and do so with a varying degree of success.

SQL statements are written in RDBMS like this-

```
CREATE TABLE Customers (
Id   CHAR(10)  NOT NULL PRIMARY KEY,
Surname  VARCHAR(30) NOT NULL,
FirstName  VARCHAR(30) NOT NULL,
DOB  DATE   NOT NULL                    [# DOB : Date of Birth]
);

SELECT InitCap(Surname) || ', ' || InitCap(FirstName)
FROM Customers
WHERE Month(DOB) = Month(getdate())
AND Day(DOB) = Day(getdate());
```

Standard SQL databases allow customized functions also, which allowthe following type of query-

```
SELECT Formal(Id)
FROM Customers
WHERE Birthday(DOB) = Today();
```

In OORDBMS, queries containing user-defined data-types and expressions like BirthDay() are seen as mentioned below-

```
CREATE TABLE Customers(
Id  Cust_Id  NOT NULL  PRIMARY KEY,
Name  PersonName  NOT NULL,
DOB  DATE  NOT NULL
);

SELECT Formal( C.Id )
FROM Customers C
WHERE BirthDay ( C.DOB ) = TODAY;
```

The object relational models can offer another interesting capability. Here, the database can make use of the relationships between the data to easily fetch the related records. For example, in an address book software application, an additional table is added to the

existing onesto hold the addresses of customers. Using a traditional RDBMS, collecting information for both the user and their address requires a "join" as mentioned below-

```
SELECT InitCap(C.Surname) || ', ' || InitCap(C.FirstName), A.city
FROM Customers C join Addresses A ON A.Cust_Id=C.Id
WHERE A.city="New York";
```

The above query when applied in an object–relational database appears in a simpler way as mentioned below-

```
SELECT Formal( C.Name )
FROM Customers C
WHERE C.address.city = "New York";
```

1.5.3Object-Oriented Relational Database Management System (OORDBMS)

An object-relational database is maintained by a relational database management system with an associating object-oriented database model, where all data and data models are created treating them as objects. Data abstraction, data hiding, early binding, late binding, polymorphism and inheritance like properties are directly supported in the database schemas and the associating query languages support the object based data access. *Oracle* is one of the popular RDBMSs, which meets the industry standards. The object-relational database systems are an attempt to merge the two dissimilar trends together. It can be visualized as an object database expansion of a relational model resulting in a hybrid design. One of the most visible aspects that we might observe is in the addition of object database features in the SQL revision. But, the tough part of a relational model immerges when someone tries to describe complex objects.

The object-oriented relational database mechanism gains its importance with the introduction of the type constructors describing row types, array type being replaced by collections, sets and lists. The creation of derived mechanisms for specifying object identity, encapsulation and inheritance is also helping OORDBMS to gain its importance. It is to be noted that the core technology used in OORDBMS is based on relational models. The commercial products (e.g. Microsoft SQL Server)have simply added a layer of some object-oriented principles on top of the relational database management system. The translation of object-oriented mechanism into relational mechanism is one of the challenging tasks for typical OORDBMS. This problem is typically addressed by an object-oriented application that does the communication between the object-oriented applications with the underlying relational databases.

Both relational and object-oriented mechanisms are having a lot of differences in terms of their underlying principles. This is the reason why this model tries to negotiate among these two techniques to adopt some intermediate measures for the sake of developer's convenience. One of the very important reasons is to permit the storage and retrieval of objects in a way how RDBMS functions. This act provides an extensive liberty to query languages to work on the object-oriented principle. Some of the common implementations in this regard are the Oracle Database, PostgreSQL, and Microsoft's SQL Server. IBM DB2 also supports objects and can be considered as OORDBMS.

In OORDBMS, the approach is essentially that of relational databases, where the data resides in the database and is manipulated collectively with queries through a query language. But, in OODBMS, where the database is essentially a persistent object store for software written in an object-oriented programming language, a programming API is solely responsible for storing and retrieving of objects. In this case, a very little or no specific support presents for query languages.

The basic need of object–relational database arises from the fact that both relational and object databases have their individual advantages and drawbacks. Although, the object oriented databases allow sets, lists, arbitrary user-defined data types and nested objects, they do not provide any mathematical base for in-depth analysis. The basic goal for the object–relational database is to bridge the gap between relational databases and the object-oriented modeling techniques. The commonly used programming languages such as C++, C#, Java and Visual Basic.NET are seen implementing these extensive features of object-relational databases. Further, the object–relational DBMS or OORDBMS allows software developers to integrate user-defined data types and methods that apply to them into the DBMS. Some of the leading features or characteristics of OORDBMS are *Complex data, Type inheritance* and *Object behavior.*

Complex data creation is based on basic schema definition through the user-defined types. Structured complex data are when stored in a hierarchy; it offers an additional property termed as *type inheritance*. That is, a structured type can have subtypes that reuse all of its attributes and contain additional attributes specific to the subtype. Finally, the *object behavior* is related with the access to the program objects. Such program objects must be storable and transportable for database processing. This is the reason why they are usually named as persistent objects. Inside a database, all the relations with a persistent program object are relations with its object identifiers. The mentioned points above can be addressed in a proper relational system, although the SQL standard and its implementations enforce arbitrary restrictions and some amount of additional intricacy. Extension of the data model with custom data types and methods is possible in a properly arranged relational system.

1.5.4Comparative Analysis of RDBMS and OORDBMS

The comparative analysis of a typical RDBMS with the OORDBMS helps understanding the changes made in the OORDBMS.

Table 1.1: Comparative Analysis of RDBMS and OORDBMS

RDBMS	OORDBMS
Ensures only the data independence part	Ensures data independence as well as data encapsulation and abstraction of data
Data can only be recognized without affecting the mode of using it	Data as well as class/object can be recognized without affecting the mode of using it
Stores only the data	Stores not only the data but also the methods imposed on that data
Data can be partitioned depending upon the user's requirements and specific user's applications	The data can be used in direct access mode and also through the class/object methods and sometimes the entire data can be made public using specific access controls
Users can perceive data as columns, rows or tuples (records) and tables	Apart from handling complex structure of data this system can handle relational data

Thus, after all these discussions, an OORDBMS can be understood as a DBMS, but with the extended relational and object oriented capabilities. It is because of the functional differences among these two extending approaches, they are to proceed in a hand-in-hand strategic and somewhat compromising approach.

1.6 Summing Up

This unit begins with the basic concept of object based databases and its management aspects. The conceptual block diagram of the object oriented database management systemis followed up by the introduction of few OODBMS applications. The standards, languages and design issues related to OODBMS are then explained. The need for the standards is cited with the focus on most relevant and important standard terminologies.

The RDBMS with object oriented features or the object oriented relational database management system is explained. The difference of RDBMS and OORDBMS is the final section of this unit to wind up.

1.7 Key Terms

DBMS It is a software platform, where data are stored, managed and retrieved as per user's requirement through some standard language queries like SQL.

RDBMS It is a similar system like DBMS which also store, manage and retrieve data when needed by the users, but is has the additional capability of maintaining relational tables or data.

ODMG ODMG (Object Data Management Group) is a consortium responsible for the monitoring of object oriented database management activities.

OODBMS It is a DBMS system, where the data are stored, managed and retrieved considering most of the data as objects is called the OODBMS. Object oriented features like inheritance, polymorphism are applicable here.

OORDBMS It is a RDBMS system with the object oriented extension which is capable of implementing object oriented features like class and object, inheritance, polymorphism etc.in addition to the classic RDBMS features.

ODL ODL is a specific kind of a language that specifies the structure of databases in object-oriented terms.

OQL OQL is a query language preferred by object data management group (ODMG) for object oriented database management purpose.

1.8 Check Your Progress

Multiple choice questions:

1. Object databases are based on
   - *i)* Relational approach
   - *ii)* Object based approach
   - *iii)* Both *(i)* and *(ii)*
   - *iv)* None of these
2. The term attribute refers to a
   - *i)* Record
   - *ii)* Row
   - *iii)* Column
   - *iv)* Key

3. Which of the following can be defined using ODL?
   i) Structure
   ii) Attribute
   iii) Operation
   iv) All of above
4. Which of the following belongs to an atomic literal?
   i) String
   ii) Boolean
   iii) Long
   iv) All of above
5. Which among the following is/are not Object Based Database(s)?
   i) Cache
   ii) Foxpro
   iii) Wakanda
   iv) Both (i) and (iii)

State whether *True* or *False:*

1. A single programming paradigm acts behind a single programming language.
2. A class is an instance of an object in OOP.
3. ODMG looks after the object models in an OODBMS.
4. MS SQL Server does not support OOP principles in any of its versions.
5. OORDBMS works in the principles of OOPs as well as the relational models.

Fill in the blanks:

1. _____ is obtained after the instantiation of a class.
2. _____ property of OOP paradigm allows multiple classes together.
3. *Identifier* and *Lifetime* are two associating properties of _____.
4. Atomic literals correspond to values of _____ data types.
5. OORDBMS stores not only the data but also _____imposed on that data.

Match the following:

1. Object	i) Inheritance
2. ODMG	ii) BCNF
3. Literals	iii) “GUIDOL”
4. Persistent	iv) RDBMS
5. Concurrency	v) ODL
	vi) Lifetime
	vii) Instantiation

1.9 Answers to Check Your Progress

Multiple choice questions:

1.(ii) 2. (iii) 3. (iv) 4.(iv) 5. (ii)

State whether *True* or *False:*

1. False 2. False 3. True 4. False 5. True

Fill in the blanks:

1. Object 2. Inheritance 3. An object 4. Basic 5. Methods

Match the following:

1.(vii) 2. (v) 3. (iii) 4.(vi) 5. (iv)

1.10 Possible Questions

Short answer type questions:

1) What is the difference between an object and a literal in the object oriented data model (OODM)?
2) What is an object? What is an object model with reference to ODMG standards?
3) What are the main difference between designing a relational database and an object database?
4) Differentiate between:
   i) Interface and Class
   ii) Atomic object and Collection object
   iii) Object identifier and Object lifetime
   iv) Persistent object and Transient object
5) What is the significance of ODL in OODBMS?

Long answer type questions:

1) Explain the major specifications mentioned in ODMG 3.0 standard.
2) Describe the differences and similarities between objects and literals in the ODMG object model?
3) Describe the steps involved in mapping the EER schema into ODB schema.
4) Explain in detail the OORDBMS concept with the introduction to all its organizing components.
5) Describe in detail the differences between RDBMS and OORDBMS.

1.11 References and Further Readings

i) M. Stonebraker, "Inclusion of New Types in Relational Database Systems", In Proc. of the International Conf. on Data Engineering (1986), pages 262–269.

ii) M. Stonebraker and L. Rowe, "The Design of POSTGRES", In Proc. of the ACM SIGMOD Conf. on Management of Data (1986), pages 340–355.

iii) M. Atkinson, et.al., "The Object-Oriented Database System Manifesto", In Proceedings of the "First International Conference on Deductive and Object-Oriented Databases", pages 223-40, Kyoto, Japan, December 1989.

iv) W. Kim And Lochovsky (Eds),Object-Oriented Concepts, Databases, and Applications, Addison-Wesley (Reading MA), 1989

v) https://en.wikipedia.org/wiki/Comparison_of_object_database_managementsystems

vi) https://en.wikipedia.org/wiki/Object_database

vii) http://www.odmg.org

UNIT 2: DISTRIBUTED DATABASE

Unit Structure:

2.1 INTRODUCTION

The database is a collection of structured information. Among other database systems, a distributed database is one where files are stored in different sites or systems. This unit will give an overview of the distributed database Management System (DDBMS). The unit shows the uses of distributed databases. Data fragmentation, replication, and allocation are very much important in a database. These are also explained in detail in this unit by considering the example of distributed database. Types of the distributed database are also explained in this unit by considering the examples. Query processing and data recovery of the distributed database are shown by taking the database example.

2.2 UNIT OBJECTIVES

After going through this unit, you will be able to know

i) About the distributed database

ii) About the types of the distributed database

iii) About the data fragmentation, replication, and allocation in a distributed system.
iv) About the query processing database distributed system.
v) About the concurrency and recovery in a distributed database.

2.3DISTRIBUTED DATABASE (DDB)

The database is a collection of structured information. Among all database systems, a distributed database is one where files are stored in different computer systems or sites. These sites are connected through a communication network. The application service layer of the distributed system provides services to the user. The users are unknown about the distributed storage structure of the system. They think that one single database is present in the system to provide services to the user. Data is distributed among the system through the communication network and it is controlled by the Distributed Database Management System (DDBMS).

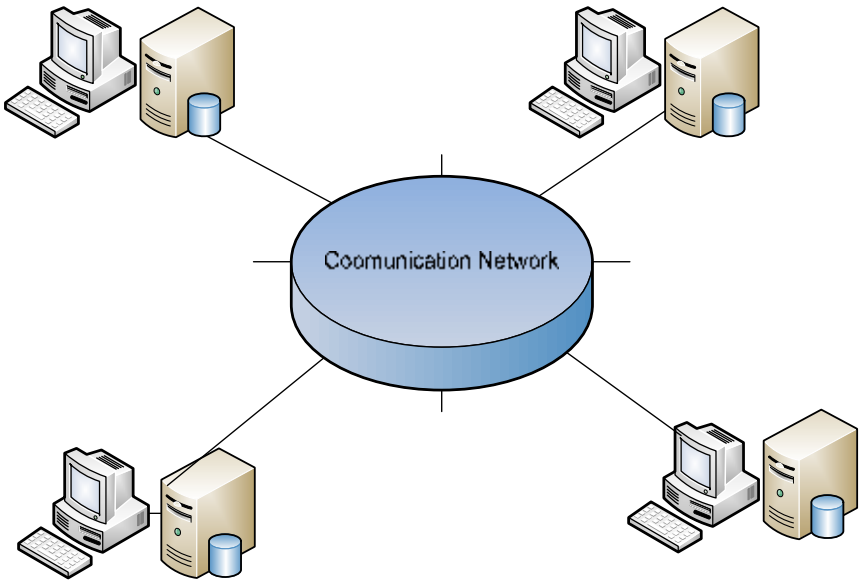


Fig. 2.1. Architecture of DDBMS

In Fig. 2.1, four different systems are interconnected through the communication network. This type of system is known as a distributed system which is also known as a loosely coupled

system. In this type of system, the data is distributed among the system. That is the reason the database of this type of system is known as a distributed database. Every DDBMS has some features.

I. Databases of the DDBMS are interlinked logically and they are connected through a communication network. Often,the DDBMS act as a single database for the user.

II. Data is physically stored in multiple sites and the data is managed by a local DBMS in the site which is independent of the other sites.

III. Adistributed database is not a loosely connected system.

IV. A distributed database integrates transaction processing.

V. DDBMS synchronizes the distributed database periodically for which it is transparent to the users.

Every distributed database has to build with some goals and these are as follows.

i) Reliability: In DDBMS, if one of the systems fails, then other systems will provide the service to the user. The other system can complete the task of the failure system.

ii) Availability: In DDBMS, sites or systems are available to provide reliability to the system. If one distributed system fails, other sites can give service, and it maintains the availability of the systems.

iii) Performance: Performance of the DDBMS can be achieved by distributing data or information over different sites which are located in different locations. So, the databases are available to every location which is maintained through the communication channel.

Check Your Progress-1

1. What do you mean by distributed database?
2. What is reliability in DDB?
3. What are the goals of a distributed system?

2.4DATA FRAGMENTATION IN DDB

Fragmentation is a normal process of diving the database into different tables in DBMS. In a distributed database, the entire database is divided into different subtables or sub relations so that each subtable or sub relation can be saved in different sites of the distributed system. These subtables or sub relations are the logical units of the DDBMS. The fragmentation is done in such a way that the subunits give the actual distributed database after combining it. Let's, you have a table T in your distributed system and it is fragmented into different sub tables t1, t2, t3, ----, tn. These fragments should have sufficient information, so that it will restore the original table after combining the t1, t2, t3, ---, tn using the UNION or JOIN operation. These subtables are known as the fragments and the process is known as data fragmentation in DDBMS. The fragments are independent of each other's and user are concerned about the data fragmentation. This is known as fragmentation transparency.

The distributed data fragmentation process has some advantages:

I. As the data is fragmented and can be stored locally, the performance of the DDBMS will increase.
II. Due to the local data store in the local sites, local query optimization is possible in DDBMS.
III. Fragmentation helps to main the security and privacy of the local system which will help to main the overall security of the DDBMS.

You have 3 methods for data fragmenting of a table and they are.

i) Horizontal Fragmentation.
ii) Vertical Fragmentation.
iii) Hybrid Fragmentation.

2.4.1 HORIZONTAL FRAGMENTATION

Horizontal fragmentation allows dividing a table horizontally into subsets of tables. It means that it will divide the table row-wise(tuple). Let's you have a table IDOL as follows.

S_Roll_No	S_Name	Branch
2020001	A	MSc. IT
2019001	D	BSc. IT
2020002	B	MSc. IT
2020003	C	MSc. IT
2019002	E	BSc. IT

Now, you can divide this table into different fragments based on different conditions such as branches. For example,

i) you may Fragment 1 where Branch is BSc. IT.
ii) you may Fragment 2 where Branch is MSc. IT.

Now, the fragment outputs are presented below.
Fragment 1 =

S_Roll_No	S_Name	Branch
2019001	D	BSc. IT
2019002	E	BSc. IT

Fragment 2 =

S_Roll_No	S_Name	Branch
2020001	A	MSc. IT
2020002	B	MSc. IT
2020003	C	MSc. IT

Now if you combine these two fragments (fragment 1 and fragment 2), you will get the original table after performing the union operation between fragment 1 and fragment 2 as follows.

IDOL = fragment 1 U fragment 2.

In DDBMS, the fragments are saved in different sites as follows. In Fig. 2.2, fragment 1 is saved in site A where fragment 2 is saved in site B.

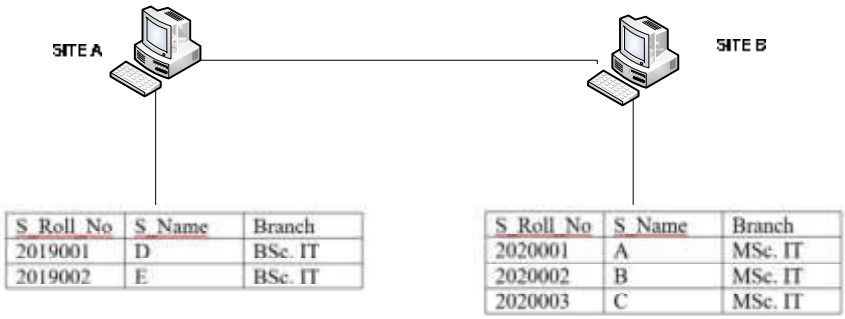


Fig. 2.2: Horizontal Fragmentation in DDBMS.

2.4.2 VERTICAL FRAGMENTATION

Vertical fragmentation divides the table column-wise (attribute). It is more complex than horizontal fragmentation. For the reconstruction of the original table from the fragment, the primary key should be available in all the fragments. The reconstruction is doing using join. For the above table IDOL database,you can create the follows vertical fragmentation.

Vertical Fragmentation 1 =

S_Roll_No	S_Name
2020001	A
2019001	D
2020002	B
2020003	C
2019002	E

Vertical Fragmentation 2 =

S_Roll_No	Branch
2020001	MSc. IT
2019001	BSc. IT
2020002	MSc. IT
2020003	MSc. IT
2019002	BSc. IT

In vertical fragmentation 1 and vertical fragmentation 2, one filed is common i.e the primary key of the IDOL table. It is required to perform the join operation between the fragments. You can join the two fragments to get back the original table IDOL as follows.
Π_{IDOL} (T1 ⋈T2).
In DDBMS, the vertical fragments are saved in different sites as follows. In Fig. 2.3, fragment 1 is saved in site A where fragment 2 is saved in site B.

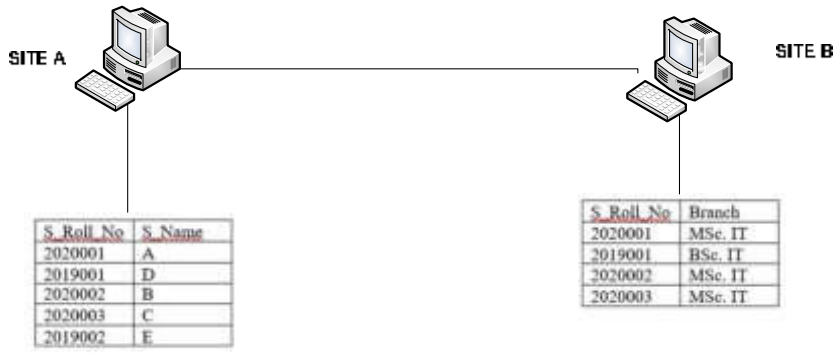


Fig. 2.2: Vertical Fragmentation in DDBMS.

2.4.3 HYBRID FRAGMENTATION

It is a combination of horizontal and vertical fragmentation. This fragmentation can be done in two ways.

I. The first method is to first create horizontal fragments and then create vertical fragments.
II. The second method is to first create vertical fragments and then create horizontal fragments

For the above IDOL table, the hybrid fragmentation can be found as follows.

S_Name	Branch
A	MSc. IT
B	MSc. IT
C	MSc. IT

In this fragmentation, the first horizontal fragmentation followed by the vertical fragmentation has been done.

Check Your Progress-II

4. True or False

   i) Vertical fragmentation divides the table column-wise (attribute).

   ii) Horizontal fragmentation allows dividing a table horizontally into subsets of tables

5. Let's you have a table COURSE as follows.

S_Roll_No	S_Name	Course
2020001	A	MSc. IT
2019001	D	BSc. IT
2020002	B	MSc. IT
2020003	C	MSc. IT
2019002	E	BSc. IT

   i) Create one horizontal fragmentation based on MSc. IT Course

   ii) Create one vertical fragmentation based on Roll No and Course = Bsc. IT

2.5DATA REPLICATION AND ALLOCATION

The process of storing data or information in more than one site or system in a distributed system is known as data replication. It is useful in improving the availability of data.Data replication

copying the data from the database of one system to another system. Due to this process, users can send the same data without any inconsistency. The main goal of data replication is to increase the availability of the data and also to increase the query processing technique. Two types of data replication are present.

i) **Synchronous Data Replication**: In this type of replication, once the changes are made in a table of the database, the data replication is done immediately.

ii) **Asynchronous Data Replication**: In asynchronous replication, the data replication is done after the commit operation of the database.

Apart from the above data replication, there are another few data replications in a distributed database.

i) **Transactional Replication**:Transactional replication is generally used in server-to-server communication. In this replication, a full copy of the database is present with one system and that system gets the update notification from the other system once the data changes. Data replication is done in real-time, so it gives a consistency guarantee. For example, Azure SQL.

ii) **Snapshot Replication**: In this replication, a snapshot of the database is sent to one database from another database. Data is not updated continuously. Data is updated infrequently at a specific time. It is more complex than transactional replication. For example, SQL Server replication.

iii) **Merge Replication**: In this replication, data of one database is combined with another database. In this type of replication, the data is updated from both databases, so hard to main consistency and concurrency. For example, Server and Client Communication (SQL server).

Data replication in DDBMS happens in different modes.They are as follows.

i) **Full Replication**: In this mode, a full copy of the database is present at every site of the distributed system. This mode increases the availability of the data in the system, and the user getsthe highest experience from it. It is hard to main the concurrency.

ii) **No Replication:** Here, the data is divided into different fragments and each fragment is present at only one site which is located in different locations. Data availability is less than the full replication but concurrency can be controlled.

iii) **Partial Replication:** Here some of the data fragments of the database are replicated but some are not. Data replication is depending on the demands of the respective data fragments.

Data allocation is a process to decide where exactly you want to store the data. It involves at per which data has to be stored at what location.The data allocation technique allocates data fragments to a site in a distributed database. Each data fragment or its replication can be stored in the particular site of a system. The process of storing data in a siteis known as data allocation. The sites and numbers of data replication depend on the demand of the data fragments. The choice of sites and the degree of replication depend on the system performance and availability and also depend on the number of transactions submitted on the site. If the user demands high availability of data, then full replication is a good choice for this allocation. Otherwise, if a fragment of data is required then partial replication can be used to allocate the data.

Three main data allocation methods are there and they are as follows.

i) Centralized: Here entire database is stored in a single site. No such data distribution or replication occurs in this process.

ii) Partitioned: In this technique, data is divided into different fragments and those fragments are stored in different sites of the distributed systems.

iii) Replicated: In this technique, a copy of the database is present in a different location and it is accessed from those locations.

2.6 TYPES OFDDB SYSTEM

There are two types of distributed databases are found and they are homogenous database and heterogeneous database.

i) Homogeneous Database

In a homogeneous database, the physical and logical structures of the database are identical for all the systems. It means that OS, DBMS, and software are the same for that system. Hence, it is easy to manage the homogenous distributed database. For example, Oracle Database server.

ii) Heterogeneous Database

In a heterogeneous distributed database, the physical and logical structures of the database are not the same. The sites use different schema and software. It is hard to manage concurrency and transactions in a distributed system. Here, one site of the distributed system may be completely unaware of the other sites of the systems. For example,Oracle8

i.

2.7 QUERY PROCESSING IN DDB

In a distributed database system, query processing is doneat the end of the user siteand server site. A query comes from the user site, so it is checked and optimized at the user site i.e. it is atthe local level. The query comes to the server, so it is processed and optimized at the server site i.e., it is at the global level.

Ina homogeneous distributed database, when a query comes from a user site, it will be able to manage the query easily as the sites have the same physical and logical structure. But in heterogeneous systems, it will not able to manage the work easily. So, there should be some techniquesto handle queries in heterogeneous system databases. There are two types of mechanisms in the heterogeneous system to manage such situations.

i) Multi-Database

In this method,a dynamic schema is created for the respective databases. If a user site uses a database, then a dynamic schema is created to connect the database D. Due to this schema, the user query is flexible with the database.

ii) Federated mechanism

In this method, a global schema is used to access the database. It means that a centralized schema is used to access all the databases of the distributed system. This global schema will work properly even though the data is fragmented and distributed over different sites.

When a federated mechanism is used, a few of the things have to manage during the database access. They are presented below.

i) Data Models

During the time global schema, the schema should take care of the data model. Because distributed database means different databases with their physical and logical structure. So, the federated schema should be compatible with all these types of systems and also should handle the query.

ii) Constraints

Each database of the distributed system has its process of defining the data constraints and has its method of accessing the data. So, the federal schema should handle these constraints.

iii) Query Language

In a distributed database, the databases are varying from site to site. So, the query languages are also varied from site to site. Hence federated schema should develop a common language that is compatible with all the query languages.

iv) Data Transfer Cost

In a distributed database, databases are distributed. So, the table of the databases is also distributed. Even some tables are fragmented. So, during the time of query processing; it may need to access the tables at the different databasesor different locations. This demands a request and transfer cost for the datawhich needs to optimize.

To explain data transfer cost,let'syou have two distributed database tables namely IDOL_EMP and IDOL_DEPT. The IDOL_EMP has a table ÉMP which is present in one location (location 1) of the distributed system, and IDOL_DEPT has another table DEPT in another location(location 2) of the distributed system. The EMP contains the basic information of the employee where the DEPT table contains the name of the department where the employee works. Let's you have 500 data of size 50 bytes in your EMP table where DEPT table has 10 data of size 10 bytes. Consider you have processed a query to find the name of the employee and department from another location

(location 3).The result of this query will include 500 records, assuming that every employee is related to a department. Suppose that each record in the query result is 40 bytes long. In this situation, you can execute your query based on the three costs, and accordingly, you can choose the optimized cost.

CASE I. You are executing your query from location 3. For this case, the cost is as bellow.

i) Cost of transferring EMP data: 500 records * 50 bytes = 25,000 bytes.

ii) Cost of transferring DEPT data: 10 records * 10 bytes = 100 bytes.

iii) Therefore, total cost = 25,000 bytes + 100 bytes = 25,100 bytes

CASE II: You can shift the data of the EMP table from location 1to location 2 and then you process it and transfer the data to location 3. For this case, the cost is as bellow.

i) Cost of transferring EMP data: 500 records * 50 bytes = 25,000 bytes

ii) Cost of transferring the result: 500 records * 40 bytes = 20,000 bytes.

iii) Therefore, total cost = 25,000 bytes + 20,000 bytes = 45,000 bytes

CASE III: You can shift the DEPT data of the EMP table from location 2 to location 1 and then you process it and transfer the data to location 3. For this case, the cost is as bellow.

i) Cost of transferring DEPT data: 10 records * 10 bytes = 100 bytes

ii) Cost of transferring the result: 500 records * 40 bytes = 20,000 bytes.

Therefore, total cost = 100 bytes + 20,000 bytes = **20,100 bytes**

Now, if you compare the cost of CASE I, CASE II, and CASE II, the cost of CASE III is the minimal one and it is optimized. Using this method, you can perform your query in the distributed database at a minimal cost.

2.8 CONCURRENCY AND RECOVERY IN DDB

During the time of concurrency control and recovery distributed databases face lots of issues. They are presented below.

i) Multiple copies of data:
A distributed system may have a copy of the database in each site to increase the availability of the system. To make a copy consistent and to maintain consistency among the copies of the database, concurrency and recovery are important, but it is not easy to maintain consistency.

ii) Failure of a site:
In a distributed system, the database of one site may fail. But the DDBMS should work with the other sites and it will try to recover the sites and make its date up to date.

iii) Failure of Communication Network:
The DDBMS must deal with the communication failure and will try to maintain the concurrency and recover the sites as soon as possible. If network portioning occurs due to network failure, then it is hard to recover the sites and maintain consistency.

iv) Distributed Commit:
The problems occur when a commit transaction is done in DDBMS where the database is present in a failed system. The two-phase commit protocol is often used to deal with this problem.

v) Distributed Deadlock:
Sometimes deadlock may occur in a distributed system. So, it is necessary to main consistency and recovery in the deadlock system.

The techniques which deal with concurrency control in DDBMS are explained below.

I. Lock based protocol:
When two transactions are present in the database, a read-write lock can apply in one transaction to avoid the concurrency issue where others can access the data. This lock

II. Shared lock system (Read lock):

The shared lock system is a read lock. The lock is shared between the transaction. Any one of the transactions can activate the shared lock for reading purposes.

III. Exclusive lock:
In this technique, an exclusive lock is activated for a transaction for the read and write operation. In this technique, no other lock can apply for the read and write operation on the same data.

Lock-based concurrency protocol locks the data. A lock is a variable that controls the read-write operation on data. It is two types.

i) One phase Locking Protocol:

In this technique, a lock is applied by a transaction on data before it uses and releases after the transaction is complete.

ii) Two-phase locking protocol:

In the two-phase locking protocol, a transaction adopts all the locks in the first phase and does not release any locks until finish all read and write operations. In the second phase, the transaction releases all the locks and never requests any locks.

Recovery is the most important process in a DDBMS. It is required to recover the information from a site. The recovery is required due to the following reasons.

i) The receiver site may down
ii) The location of the receiver site may crash.
iii) The communication link between the sender and receiver site may break.

A two-phase commit protocol is used to overcome the issue of the data recovery on DDBMS. This atomic protocol coordinates the process of DDBMS which decides to commit or terminate a transaction. It provides the automatic recovery option in case of a site failure. The original place of transaction is known as coordinator and other places of the transactionare known as a cohort. The protocol executes in two phases.

i) Commit request: In the commit phase, the coordinator prepares the list of cohorts and asksto commit the transaction.
ii) Commit phase: Based on the responses from the cohorts, the coordinator can decide to commit or terminate a transaction.

Check Your Progress-III

6. All sites in a distributed database commit at exactly the same instant. TRUE/FALSE
7. Fill in the blanks.

i) The real use of the Two-phase commit protocol is _____.

ii) Read one, write all available protocol is used to increase _____ in a distributed database system.

iii) Commit and rollback inDDB are related to .....

iv) If a distributed transactions are well-formed and 2-phasedlocked, then ..... is the correct locking mechanism in distributed transaction as well as in centralized database.

v) A distributed transaction can be ..... if queries are issued at one or more nodes.

2.9. SUMMING UP

1) The distributed database is a collection of structured information. Among all database systems, a distributed database is one where files are stored in different computer systems or sites. These sites are connected through a communication network.
2) Data in DDB is physically stored in multiple sites and the data is managed by a local DBMS in the site which is independent of the other sites.
3) A distributed database integrates transaction processing.
4) In DDBMS, if one of the systems fails, then other systems will provide the service to the user. The other system can complete the task of the failure system.
5) Fragmentation is a normal process of diving the database into different tables in DBMS. In a distributed database, the entire database is divided into different subtables or sub relations so that each subtable or sub relation can be saved in different sites of the distributed system.

6) You have 3 methods for data fragmenting of a table and they are.
    a) Horizontal Fragmentation.
    b) Vertical Fragmentation.
    c) Hybrid Fragmentation
7) The process of storing data or information in more than one site or system in a distributed system is known as data replication. It is useful in improving the availability of data.
8) Two types of data replication are present.
    a. **Synchronous Data Replication**: In this type of replication, once the changes are made in a table of the database, the data replication is done immediately.
    b. **Asynchronous Data Replication**: In asynchronous replication, the data replication is done after the commit operation of the database.
9) Data allocation is a process to decide where exactly you want to store the data. It involves at per which data has to be stored at what location.
10) There are two types of distributed databases are found and they are homogenous database and heterogeneous database. a) **Homogeneous Database b) Heterogeneous Database**
11) In a distributed database system, query processing is done at the end of the user site and server site. A query comes from the user site, so it is checked and optimized at the user site i.e. it is at the local level. The query comes to the server, so it is processed and optimized at the server site i.e. it is at the global level.
12) During the time of concurrency control and recovery distributed databases face lots of issues. They are presented below.
    a) Multiple copies of data:
    b) Failure of a site:
    c) Failure of Communication Network:
    d) Distributed Commit:
    e) Distributed Deadlock:
13) A lock is a variable that controls the read-write operation on data. It is two types.
    a) One phase Locking Protocol:

In this technique, a lock is applied by a transaction on data before it uses and releases after the transaction is complete.

b) Two-phase locking protocol:

In the two-phase locking protocol, a transaction adopts all the locks in the first phase and does not release any locks until finish all read and write operations. In the second phase, the transaction releases all the locks and never requests any locks.

2.10 ANSWERS TO CHECK YOUR PROGRESS

1) The distributed database is a collection of structured information. Among all database systems, a distributed database is one where files are stored in different computer systems or sites. These sites are connected through a communication network.
2) In DDBMS, reliability means if one of the systems fails, then other systems will provide the service to the user. The other system can complete the task of the failure system.
3) The goals of DDB are as follows.
   I. Reliability
   II. Availability
   III. Performance
4) i) TRUE ii) TRUE
5) i)

S_Roll_No	S_Name	Branch
2020001	A	MSc. IT
2020002	B	MSc. IT
2020003	C	MSc. IT

ii)

S_Roll_No	Branch
2019001	BSc. IT
2019002	BSc. IT

6) FALSE
7)
   i) Atomicity, i.e, all-or-nothing commits at all sites
   ii) Both Availability and Robustness
   iii) Data Consistency
   iv) A two-phase locking.

v) partially read-only

2.11 POSSIBLE QUESTIONS

Short answer type questions:

i) What is a distributed system?
ii) What is distributed database?
iii) What are the goals of the distributed database?
iv) What is the reliability and availability of distributed database?
v) Difference between one phase and two-phase locking protocol.
vi) What are the types of distributed database systems available?
vii) What are the different modes of data replication in a distributed system?
viii) Difference between lock-based and shared lock systems.
ix) What is data replication in DDBMS? What are the types?

Long answer type questions:

i) Explain the distributed database system with an example.
ii) Explain with examples the fragmentation of tables in the distributed system.
iii) How are concurrency and recovery achieved in the distributed database?
iv) Explain data replication and allocation in DDBMS.
v) Explain the query processing in DDBMS.

2.12 FURTHER READINGS

i) *Principles of Distributed Database Systems*. Author: M. Tamer Özsu.

ii) *Distributed System: Concepts, Design, and Applications*Publisher: *O,Reilly*, Author: *S.K.Singh*

UNIT 3: Image and Multimedia Database

Unit Structure:

3.1 INTRODUCTION

This unit gives an overview of the multimedia database,especially about the image database. Image means the collection of pixels. Pixels have information about the images. The process of storing images in a database is discussed in this unit. The unit also discusses the contents of the multimedia database. Challenges of the multimedia database are also discussed in this unit.The contents of the multimedia database are also pointed in this chapter. Finally, the different applications of the multimedia database are reported in the unit.

3.2 UNIT OBJECTIVES

After learning this unit, you willbe able to learn

i) About the definition of multimedia database

ii) About types of multimedia database including an image.

iii) About image and multimedia database.

iv) About the challenges and contents of the multimedia database.

v) About the challenges of the multimedia database.

vi) About the applications of the multimedia database.

3.3 CONCEPT OF IMAGE

An image is multimedia data. It consists of the pixel. The pixel of an image contains all the necessary information about the image. An image may be color, grayscale, or black and white. You can extract the information of color from the pixel of an image. Apart from color, other features such as texture and shape are also possible to extract from an image. These features can be stored in a database. Images are used in different fields, so it is necessary to store the images in the database. The database where images are stored is known as a multimedia database.

3.4 IMAGE AND MULTIMEDIA DATABASE

An imagecan not directly store in a database using a standard SQL insert command. The embedded SQL is used to insert the images into a database. A database should support an image to insert an

image in the database. The images are stored in binary form in the cell of a table of a database and the data type of the cell is Binary Large Object (BLOB). It is a MySQL datatype that is not onlyused to storethe image data but also used to storethe other data type. For tightly coupled database such as employee database, student database needs to upload the image in the database, so this type of databases are known as multimedia database and storing of an image one of the part of this database.

Let's explain the BLOB in MySQL using python.Her, you will learn about the process of insertion and deletion of multimedia files such as images, video, or songs in a multimedia MySQL database using python.To Store and retrieve multimedia data, i.eBLOB data in a MySQL table, you should have a table containing binary data or you can update your table by inserting one extra column in the database for the BLOB data. You can execute the following queries for the BLOB data.

i) **Table Creation Query:**CREATE TABLE `idol_emp` (`emp_id` INT NOT NULL , `emp_name` TEXT NOT NULL , `emp_photo` BLOB NOT NULL , `emp_biodata` BLOB NOT NULL , PRIMARY KEY (`id`))

In query (i), the emp_photo and emp_biodata, these two fields require the BLOB data. So their data types are BLOB.

ii) **Data Insertion Query:** As BLOB is MySQL datatype and it has the following four BLOB datatype depending on the length of the data that they can hold.

a) TINYBLOB
b) BLOB
c) MEDIUMBLOB
d) LONG BOB

To insert the data into 'idol_emp' using BLOB and python, you need to perform the following steps.

a) You need to install MySQL-Python connector using pip and then need to establish the connection.
b) You need a python function that converts images and other multimedia data into binary data.
c) Then define your insert query and execute the query using the cursor.execute() function.
d) After the query execution, you need to commit your database changes.
e) Then you need to close your cursor and database connection.
f) Finally, verify your result.

The code of insertion into the database using the BLOB is given below.

```
import mysql.connector

def multimediaToBinary(filename):
   with open(filename, 'rb') as file:
binaryData = file.read()
   return binaryData

def insertBLOB(emp_id, emp_name, emp_photo, emp_biodata):
print("Inserting multimedia data into idol_emp")
   try:
      connection = mysql.connector.connect(host='localhost',
                                 database='idol_db',
                                 user='idol',
                                 password='idolidol')
```

```
        cursor = connection.cursor()
sql_insert_blob = """ INSERT INTO idol_emp
                        (emp_id, emp_name, emp_photo, emp_biodata)
VALUES (%s,%s,%s,%s)"""

emp_photo = convertToBinaryData(emp_photo)
emp_biodata = convertToBinaryData(emp_biodata)
insert_blob = (emp_id, emp_name, emp_photo, emp_biodata)
        result = cursor.execute(sql_insert_blob, insert_blob)
connection.commit()
print("Image and biodata has inserted successfully ", result)

    except mysql.connector.Error as error:
print("Failed inserting multimedia data {}".format(error))

    finally:
        if connection.is_connected():
cursor.close()
connection.close()
print("MySQL connection is closed")

insertBLOB(1, "idol_emp1","path of the image",
           "path of the text")
```

After data insertion in a database using the BLOB in MySQL, you can retrieve the data from the database as given below. For the same, MySQL and python connector is required as stated above. But to executethe select query cursor.execute() function is required. Then you can use cursor.fetchall() to retrieve the data from the database as below.

```
import mysql.connector
def write_file(data, filename):
Disk
   with open(filename, 'wb') as file:
file.write(data)



def readBLOB(emp_id, emp_photo, emp_bioData):
print("Reading data from idol_emp table")

   try:
      connection = mysql.connector.connect(host='localhost',
                                    database='idol_db',
                                    user='idol',
                                    password='idolidol')

      cursor = connection.cursor()
sql_fetch = """SELECT * from idol_emp where id = %s"""

cursor.execute(sql_fetch, (emp_id,))
      record = cursor.fetchall()
      for row in record:
print("Employee Id = ", row[0], )
print("Employee Name = ", row[1])
Employee image = row[2]
Employee biodata = row[3]
print("Storing employee's photyo and biodata in the local PC")
write_file(Employee image, emp_photo)
write_file(Employee biodata, emp_biodata)

   except mysql.connector.Error as error:
```

```
print("Failed to read data from idol_emp {}".format(error))


    finally:
        if connection.is_connected():
cursor.close()
connection.close()
print("MySQL connection is closed")

readBLOB(1, "path of the image",
```

Check Your Progress-1

1. What do you mean by multimedia database?
2. What are the BLOB data types?
3. State truth or false
    i. TINY BLOB is Blob data type
    ii. MEDIUMBLOB is not a blob data type.
    iii. Images consist of pixel
4. What is cursor.execute() function?
5. What is the role of cursor.fetchall() ?

3.5REQUIREMENT OF MULTIMEDIA DATABASE

Like other DBMS, the multimedia database should address the requirement issues.

i) Integration: It indicates that data of a multimedia database should not be duplicate.

ii) Concurrency control: Like other DBMS, a multimedia database should control the concurrency of the

transaction. Otherwise, consistency issues will be arises.

iii) Data Independency: In the multimedia database, data of the different multimedia should be independent. It should be managed from the user side.

iv) Persistence: Data of a multimedia database should be saved and reused by the other transactions.

v) Recovery: Data should be recovered at the time of failure. A system may fail due to different reasons, but the recovery option of a multimedia database should recover the data at the time of need.

3.6 CHALLENGES OF MULTIMEDIA DATABASE

Like other DBMS, multimedia databases also have some challenges. They are presented below.

i) The designing of the multimedia database is not so easy as the different format of data is present in the multimedia database.

ii) As the multimedia database consists of images, text, video, mp3, etc. So conversion of one file format to another format is not so easy.

iii) Storing multimedia data requires more amounts of space. Designing a large dataset is not so easy.

iv) Processing is another issue of multimedia databases because the processing of data requires more amount of time.

v) Multimedia query processing and execution is another issue of the multimedia database.

Check Your Progress-II

6. What is concurrency in multimedia database?
7. State truth or false
    i. Integration is a requirement of multimedia database.
    ii. Data independecy should be a part of multimedia database.
    iii. All multimedia shoud not be recovered.
8. State two challenges of multimedia database.

3.7CONTENTS OF MULTIMEDIA DATABASE

The multimedia database store the multimedia information. It contains the following information.

i) The multimedia database contains the multimedia data like audio, video, text, animations, and images.

ii) The media of the multimedia information contains the sampling rate, the frame rate of the data signal.

iii) The keyword data is used to represent the data of a multimedia database such as image keyword means its date, time, and description of the image.

iv) The media feature data contains the features of a data. For example, an image means its color, texture, and shape.

3.8APPLICATION OF MULTIMEDIA DATABASE

The multimedia database can be applied in the following areas.

i) The Multimedia databases can be applied in the area of documentand record management systems such as insurance claim records.

ii) Multimedia databases can be applied in digital libraries. For example IR@inflibnet.

iii) The Multimedia databases are used in the video on demand. For example. Netflix

iv) A Multimedia database isused in music. For example. Ganna.

v) The multimedia database is used in GIS. For example. Landsat 8.

Check Your Progress-III

9. State truth or false
   i. Audio is not a part of multimedia database.
   ii. Ganna.com is an example of multimedia database.
10. State two applications multimedia database.
11. What is a netflex?
12. What is inflibnet?

3.9 SUMMING UP

I) A Multimedia database is a collection of multimedia data such as texts, images, videos, audios, etc.

II) An image is multimedia data. It consists of the pixel. The pixel of an image contains all the necessary information about the image.

III) An image may be color, grayscale, or black and white.

IV) The embedded SQL is used to insert the images into a database. A database should support an image to insert an image in the database.

V) The images are stored in binary form in the cell of a table of a database and the data type of the cell is Binary Large Object (BLOB). It is a MySQL data type that is not only used to store the image data but also used to store the other data type.

VI) **Table Creation Query:** CREATE TABLE `idol_emp` (`emp_id` INT NOT NULL , `emp_name` TEXT NOT NULL , `emp_photo` BLOB NOT NULL , `emp_biodata` BLOB NOT NULL , PRIMARY KEY (`id`))

VII) **Data Insertion Query:** As BLOB is MySQL datatype and it has the following four BLOB data type depending on the length of the data that they can hold.

a) TINY BLOB
b) BLOB
c) MEDIUMBLOB
d) LONG BOB

VIII) Like other DBMS, the multimedia database should address the requirement issues.

1. Integration:
2. Concurrency control
3. Data Independency
4. Persistence:
5. Recovery:

IX) Like other DBMS, multimedia databases also have some challenges. They are.

1. Designing of Multimed a database.
2. One File Format for multimedia data.
3. Processing is another issue of multimedia databases because the
4. Multimedia query processing and execution.

X) The multimedia database store the multimedia information. It contains the following information.

1. Audio, video, text, animations, and images.
2. The sampling rate, the frame rate of the data signal.
3. Image keyword means its date, time, and description of the image.
4. Features of data.

XI) The multimedia database can be applied in the following areas.

1. Insurance claim records.
2. Inflibnet.
3. Netflix
4. Ganna.
5. Landsat 8

3.10 ANSWERS TO CHECK YOUR PROGRESS

1. A Multimedia database is a collection of multimedia data such as texts, images, videos, audios, etc.
2. The four BLOB data types are

i. TINY BLOB
ii. BLOB
iii. MEDIUMBLOB
iv. LONG BOB

3. i) True ii) False iii) True
4. To insert data in the multimedia database, the insert queries are executed using the cursor.execute() function.
5. Using the cursor.fetchall(), one can retrieve the data from the multimedia database.
6. In a database management system (DBMS), concurrency control manages simultaneous access to a database. Like other DBMS, a multimedia database should control the concurrency of the transaction. Otherwise, consistency issues will be arises.
7. i) True ii) True iii) False
8. Like other DBMS, multimedia databases also have some challenges. They are presented below.
    i. The designing of the multimedia database is not so easy as the different format of data is present in the multimedia database.
    ii. As the multimedia database consists of images, text, video, mp3, etc. So conversion of one file format to another format is not so easy
9. i) False ii) True
10. The multimedia database can be applied in the following areas.
    i. Multimedia databases can be applied in digital libraries. For example IR@inflibnet.
    ii. The Multimedia databases are used in the video on demand. For example. Netflix

11. Netflix is an example of a multimedia database. It is a streaming service that offers a wide variety of award-winning TV shows, movies, anime, documentaries, etc.
12. Information and Library Network (INFLIBNET) is an example of a multimedia database and is an autonomous Inter-University Centre of the University Grants Commission (UGC) that provides access to e-resources to colleges, universities, and centrally funded technical institutions

3.11POSSIBLE QUESTIONS

Short answer type questions:

1. What is a multimedia database?
2. Define the terms image, video, and audio.
3. What is BLOB?
4. Why do you need BLOB?
5. What are the data types of BLOB?
6. What is an embedded query?
7. How do create a multimedia table?
8. How do you insert queries in a multimedia table?
9. State two issues of a multimedia database?
10. State two design issues of the multimedia database.
11. State two challenges of the multimedia database.
12. State two applications of a multimedia database.

Long answer type questions:

1. Explain the data insertion and retrieve in a multimedia database using python.
2. Explain the different challenges of a multimedia database.
3. Explain the requirements of a multimedia database.

3.12 FURTHER READINGS

1. Multimedia Database Management Systems, Author: Prabhakaran, Publisher: Springer
2. Multimedia Database Management Systems, Author: Guojun Li.

UNIT 4: SPATIAL DATABASE

Unit Structure:

4.1 INTRODUCTION

This unit gives an overview of the spatial database. A spatial database is one in which the geographic location information is saved. The concept of a spatial database is explained here along with its indexing techniques. Content-based indexing is also discussed in this unit and along with the retrieval techniques. Content-based image indexing means the properties of an image are saved in the database and it is retrieved based on its properties. Finally, the different indexing techniques such as R trees, R+ Tress, and KD tree is discussed in the unit.

4.2UNIT OBJECTIVES

After learning this unit, you will be able to learn

i) About the concept spatial database

ii) About the process of saving location in database

iii) About the Content-Based Indexing (CBI) and its retrieving.

iv) About the indexing technique such as R tree, R+ tree, and KD tree.

4.3 SPATIAL DATABASE CONCEPT

The spatial data is one where the geographic location such as a village, town. Cities or locations are associated. The spatial database is where you can be saved this type of information in terms of some location object data. Technically, a spatial database is optimized for storing and querying object data in a geometric space. The Spatial database stores geometric objects like points, lines, and polygons but some databases are also saved 2d objects, linear networks, etc. It is a part of the GIS database (Fig.3.1)

Let's understand the concept of a spatial database with the help of the following example. Let's have a satellite image of a road. Though it is an image it has geographic information such as points, lines, and polygons which represent the building, rod, etc. So, this image, the spatial data is represented by vector data and raster data. You can say that spatial data are two types.

i) Vector data: The data is represented using lines, points, and polygons.

ii) Raster data: The data is presented using the matrix. For example, data for building.

The database system which manages the spatial data is known as Spatial Data Base Management System (SDBMS). The SDBMS plays a prominent role in the management of queries of spatial data. Spatial data is used in many disciplines such as geography, remote sensing, urban planning, and natural resource management. As mentioned above, the spatial database established a specification to represent the above two types of spatial data. Using this specification, spatial queries are processed using the SQL (For example PostgreSQL. PostGIS, QGIS).

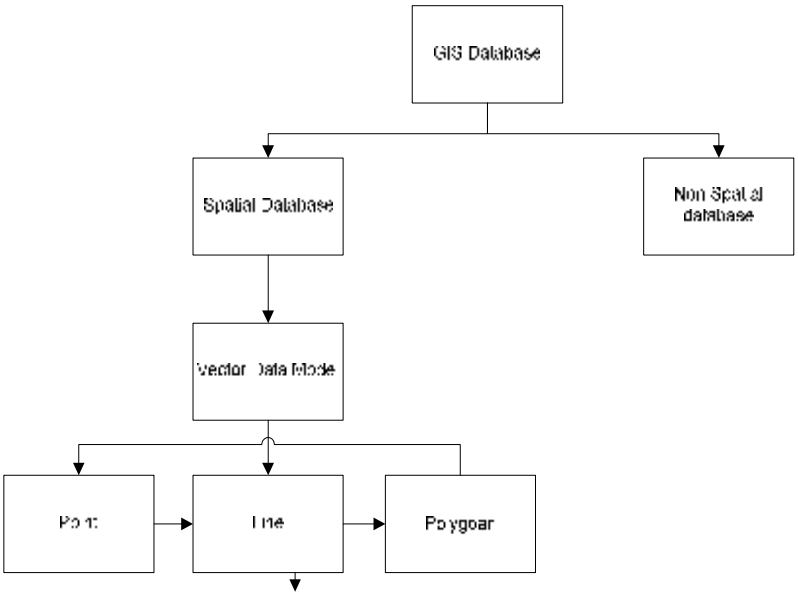


Fig.3.1. Spatial Database Types

Other database uses indexing technique to access the data faster and search the data most efficiently. But these direct indexing techniques may not work properly in the case of spatial databases. So, it needs a special type of indexing known as content indexing. The spatial indexes such as R tree, R+tree, etc. are designed for spatial indexing. It is required to retrieve spatial data from a large database. Apart from indexing, spatial databases offer spatial data types in their data model and query language. This special data type is required to model the spatial database.

A spatial database can be applied in many areas. A few of the applications are presented below.

i) **Image and Multimedia databases:** In the multimedia database, the spatial database is applied such as content-based image retrieval, content-based video retrieval, medical database, etc.

ii) **Time-series databases**: In the management of time intervals, the spatial database is used.

iii) **Traditional DBMS:** In the case of data warehouses, the SDBMS is used.

iv) **Socio-Economic applications:** In Urban planning, the Route optimization problem, and the market analysis of the SDBMS are used.

v) **Environmental applications:** In the case of Fire or Pollution Monitoring, the SDBMS is used

vi) **Administrative applications**: In Public networks administration and vehicle navigation, the SBMS is used.

The SDBMS are necessary for the following requirements before its designs.

i) For the manipulation of very large amounts of data, e.g., terabytes of data per day from satellite images, the SDBMS is required.

ii) For data distinction, e.g., spatial and non-spatial (alphanumeric) data, the DBMS is necessary.

iii) For Complex spatial relationships and operations, e.g., topological, directional, metric relationships, the SDBMS is necessary.

iv) Complex spatial relationships, e.g., find all cities adjacent to a river, find all dark shapes left to the heart,

and find the 5 closest hospitals concerning a given location.

v) Spatial join: An expensive operation, e.g., Find the 5 closest hospitals concerning any highway.

Check Your Progress-I

1. What do you mean by spatial data and spatial database?
2. State few applications of spatial database.
3. Which classes does spatial data types in MySQL correspond to?
4. Stet true or false
   i) SPATIAL indexes cannot be created on NOT NULL spatial columns.
   ii) By *'spatial data'* we mean data that has position value.

4.4SPATIAL DBMS DATA MODELS

In section 4.3, the two types of the data model of SDBMS are already mentioned. They are

i) **Raster Model**: In the raster model, SDBMS spaces are subdivided into cells of regular size and shape such as square, triangle, hexagon, etc. Each cell of the raster is assigned the value of the attribute it represents and only one value is assigned for the same. Different attributes are stored in separate files (layers).

ii) **Vector Model**: In the vector model, the subdivision of the spaceis done based on the position of the geographic feature, i.e., irregular. The features are represented by (2-D space), such as Points (x,y), Lines (x1,y1, x2,y2, ..., xn,yn), Regions (x1,y1, ..., xn,yn, x1,y1).

4.5CONTENT-BASED INDEXING AND RETRIEVAL

Like another database system, SDBMS also needs indexing of spatial data for faster query processing and searching spatial data most efficiently. Content-based indexing is one where data is saved based on the properties of the data and it is retrieved based on these properties. "Content-based" means that the search analyses the contents of the data rather than the metadata such as keywords, tags, or descriptions associated with the data. It results in faster query processing and searching. For example, the Content-Based Image Indexing and Retrieval (CBIR) system are where images are saved in the database based on the properties of the image data. The properties of the image mean its color, texture, and shape. So based on these, you can index the images and retrieve also.

The content indexing and retrieving are applicable in many areas but image, video, text, music are very popular ones. Let's explain the CBIR with a help of an example. Let's you have many images in your database. In CBIR, query images will be there with you. Initially, the feature of the image such as color, texture, position, shape, etc can extract from the query image. The features are saved as a vector for the query image and the same process can be applied to the database also. When you have both the feature vector, just compare the feature vector of the query image with the feature vector of the database image.

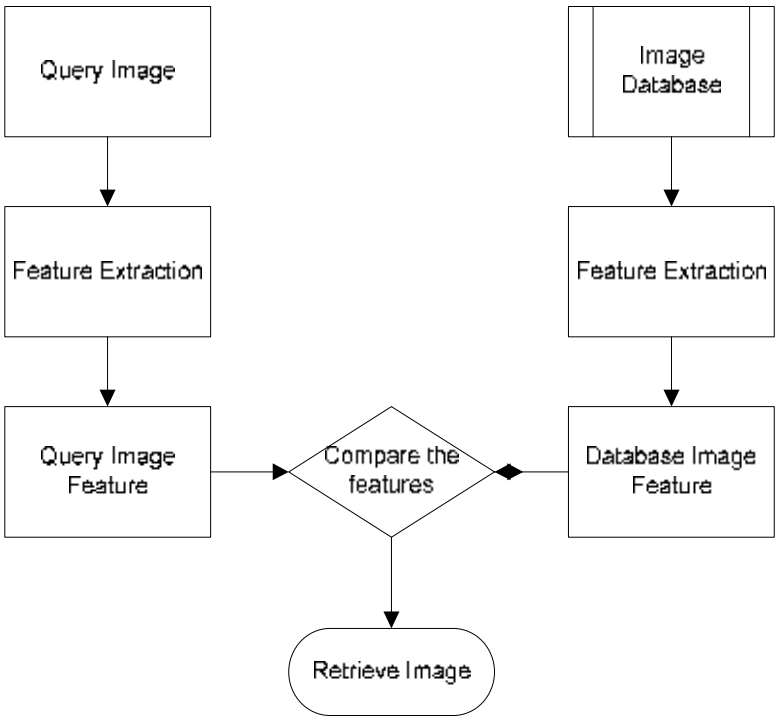


Fig. 3.2: CBIR system

Check Your Progress-II

5. What are the spatial data models?
6. Can we use image and video for CBIR system?
7. Give two examples of CBIR search engine.
8. What features of an image are considered for CBIR?

4.6 DIFFERENT INDEXING TECHNIQUES

For the indexing of spatial data, different indexing techniques are used.

i) R Tree

ii) R+ Tree

iii) KDTree

Let's explain these techniques with the help of examples.

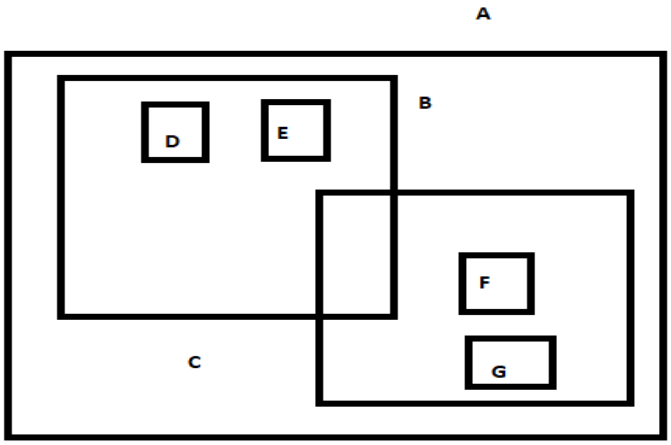
I. **R tree:** R-tree is a tree data structure to store the spatial data efficiently. It is used for storing spatial data indexes. It is useful for spatial data queries, storage, and indexing. are highly useful for spatial data queries and storage. Indexing multi-dimensional information. For example, handling of game data, virtual maps implementation, and handling geospatial coordinates, etc. The properties R tree are given below.

i. Consists of a single root with internal and leaf nodes.

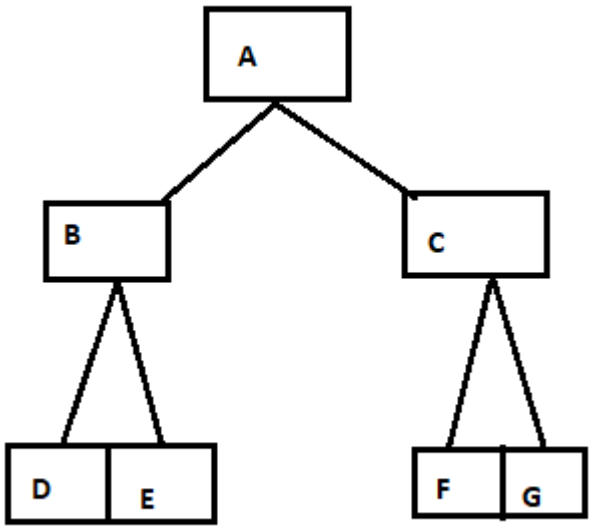
ii. The root node contains a pointer to the largest region.

iii. The parent nodes contain pointers to their child nodes where the region of child nodes completely overlaps with the regions of parent nodes.

iv. Leaf nodes contain the actual data within the Minimum Bounding region (MBR) to the current objects where the MBR is the sub-regions within the entire space that group data as efficiently.



(a)



(b)

Fig. 3.3 R tree in SDBMS

To locate an object, the search algorithm descends the tree from the root. The algorithm recursively traverses down the sub trees of bounding rectangles that intersect the query rectangle. When a leaf node is reached, bounding rectangles are tested against the query rectangle and their objects are fetched for testing if they intersect the query rectangle.

II. **R +Tree**: R+tree is a variant of R trees where data is indexed using (x,y) coordinates. It is a conciliation between the R tree and the KD tree. In the R+ tree, the nodes may not be half-filled and the internal nodes of the tree avoid overlapping by inserting an object into multiple leaves. In the R+ tree, the tree has minimal coverage and minimal overlap and it overcomes the overlapping issue of the R tree. The advantages and disadvantages of the R+ tree are presented below.

Advantages:

i) Due to no overlapped between the nodes, the point query performance benefits are covered by at most one node.

ii) A single path is identified to visit the nodes.

Disadvantages:

i) Since rectangles are duplicated, an R+ tree can be larger than an R tree built on the same data set.

ii) Construction and maintenance of R+ trees are more complex than the R trees and other variants of the R tree.

The duplication of objects or nodes in R+tree leads to the non-overlapping of entries. If the corresponding covering rectangles intersect the query region, then only the searching is possible in R+tree. The disjoint covering rectangles avoid the multiple search paths of the R-tree for point queries.

To insert an object, multiple paths may be traversed. At a node, the sub trees with covering rectangles that intersect with the object bounding rectangle must be traversed. On reaching the leaf nodes, the object identifier will be stored in the leaf nodes. Multiple noes of R+tree may store the same object. Three cases should take care of the insertion.

i) Insert an object into a node where the covering rectangles of all entries do not intersect with the object-bounding rectangle.

ii) The second one is when the bounding rectangle of the new object only

partially intersects with the bounding rectangles of entries.

iii) The third case is more serious in that the covering rectangles of some entries can prevent each other from expanding to include the new object.

III. **KD Tree**: KD tree is a binary search tree that is also known as K dimensional tree. In the KD tree, the data in each node of the tree represents the K dimensional point in space. So it is also known as space partitioning data structure. It represents the points or data in K dimensional space. The non-leaf node of the KD tree effectively divides the tree into two spaces, known as half-space. The data that is left of the root will go into a left subtree, data right of a root will go in a right subtree. Construction of the KD tree is as follows:

i) The axis used to generate splitting treesis cycled repeatedly.

ii) The nodes are selected by taking the median of the data being placed in the subtree.

Let's understand the basic concept of the KD tree by considering a 2D tree. In the KD tree, the left subtree contains those points whose coordinates are smaller than the root node, and the right subtree contains those points whose coordinates are grater-equal to the root node.

Let’s build a k-d tree with the points: (30,40),(5,25), (70,70), (50,30),(35,45). Let the root node is x aligned.

i) Take the first coordinates (30,40). As the tree is empty, so make it the root node of the tree.

ii) Now the 2nd coordinate is (5, 25). As the first x value of the 2nd coordinates is 5 and 5<30. So it will go to the left subtree.

iii) The 3rd coordinate is (70, 70). Now 70>30, so it will go right subtree.

iv) The 4th coordinate is (50, 30). First, compare with root 50>30. But already (70, 70) is in the right subtree. Now Compare 50 with the y value of 70. 50<70. So, it will be in the left sub tree of (70, 70).

v) The final coordinate is (35, 45). Comparing with root (35>30). It will go right sub tree. In the right sub tree, comparing with y coordinates of (70, 70), you find 35<70. So, it will go left sub tree of (70, 70). But in the left sub tree of (70, 70), the (50, 30) coordinate is present. Now compare 35 of (35, 45) with x of (50, 30). So, 35<45. So, it will be on the left of (50, 30).

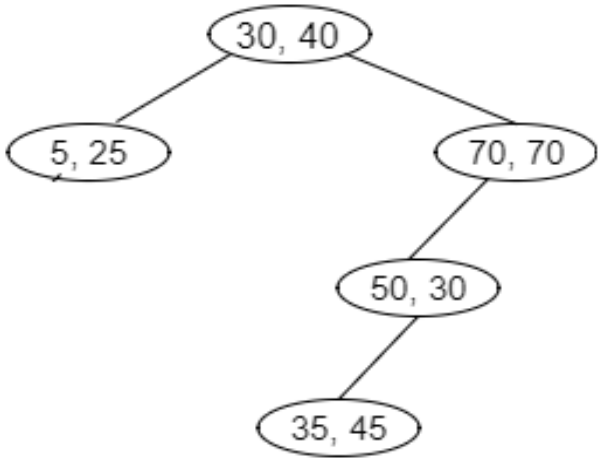


Fig. 3.4 KD tree in SDBMS

Check Your Progress-III

9. What is r tree and r+ tree?
10. In what time can a 2-d tree be constructed?
11. In a k-d tree, what is K meant?
12. Each level in a k-d tree is made of cutting and dimension (True or False)

4.7SUMMING UP

1. The spatial data is one where the geographic location such as a village, town. Cities or locations are associated. The spatial database is where you can be saved this type of information in terms of some location object data.
2. The spatial data are two types.
   i) Vector data: The data is represented using lines, points, and polygons.
   ii) Raster data: The data is presented using the matrix. For example, data for building.
3. The spatial queries are processed using the SQL (For example PostgreSQL. PostGIS, QGIS).
4. The SDBMS are necessary for the following requirements before its designs.
   i) For the manipulation of very large amounts of data.
   ii) For data distinction.

iii) For Complex spatial relationships and operations.

iv) Complex spatial relationships.

v) Spatial join.

5. Content-Based Image Indexing and Retrieval (CBIR) system are where images are saved in the database based on the properties of the image data.
6. R-tree is a tree data structure to store the spatial data efficiently. It is used for storing spatial data indexes. It is useful for spatial data queries, storage, and indexing. are highly useful for spatial data queries and storage. Indexing multi-dimensional information.
7. R+tree is a variant of R trees where data is indexed using (x,y) coordinates. It is a conciliation between the R tree and the KD tree.
8. KD tree is a binary search tree that is also known as K dimensional tree. In the KD tree, the data in each node of the tree represents the K dimensional point in space.
9. Construction of the KD tree is as follows:
    i. The axis used to generate splitting trees is cycled repeatedly.
    ii. The nodes are selected by taking the median of the data being placed in the subtree.

4.8 ANSWER TO CHECK YOUR PROGRESS

1. The spatial data is one where the geographic location such as a village, town. Cities or locations are

associated. The spatial database is where you can be saved this type of information in terms of some location object data.

2. The few applications of spatial DBMS are
    i) Image and Multimedia databases
    ii) Time-series databases
    iii) Traditional DBMS
3. OpenGIS
4. i) FALSE ii)TRUE
5. Two types of spatial data models are raster and vector model.
6. For CBIR, we can use only image.
7. eBay image Search and Google Image Search.
8. Color, shape, and texture.
9. R and R+ tree are spatial indexing techniques in spatial DBMS.
10. O(nlogn)
11. Number of dimensions.
12. True

4.9POSSIBLE QUESTIONS

Short answer type questions:

1) What do you mean by spatial data and spatial database system?
2) Explain raster and vector model in spatial DBMS.
3) What is CBIR? Give examples.
4) What are the applications of spatial DBMS?
5) What are the requirements of Spatial SBMS?
6) State the difference between R and R+ tree.
7) Give some examples of spatial query language.

8) What are advantages of R+ tree over R tree?
9) Why do need KD tree if you have R and R+ tree?
10) What are the requirements of spatial DBMS?

Long answer type questions:

1) Explain the CBIR system with an example and diagrams.
2) Explain the KD tree with an example.
3) What are the indexing techniques of Spatial DBMS? Explain.

4.10FURTHER READINGS

1) Spatial Databases: With Application to GIS, by Michel O. Scholl
2) Spatial Data Management by Nikos Mamoulis