Purpose of Database System – Views of data – Data Models – Database System Architecture – Introduction to relational databases – Relational Model – Keys – Relational Algebra – SQL fundamentals – Advanced SQL features – Embedded SQL– Dynamic SQL

DATA -

Collections of facts and figures which can be processed to provide information.

DATABASE

It is a collection of related data. Mostly data represents recordable facts.

STUDENT

Name	RollNo	class	Department
Rajesh	44	II	CSE
Ramesh	67	II	ECE

SUBJECT

Sub_name	Sub_code	Credit_hrs	Dept
DBMS	CS6302	3	CSE
CNS	IT2352	3	IT

GRADE REPORT

Roll_no	Sub_code	Grade
44	CS6302	A
75	IT2352	B

DBMS

A database-management-system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data is usually referred to as the database.

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Characteristics of a good database

1. Should be able to store all kinds of data which exists in the real world.
2. Should be able to relate the entities/tables in the database by means of a relationship.
   (ie) Any two tables should be related.
3. There should not be any duplication of data in the database. If repetition of data occurs, it would be an unnecessary waste of DB space. [Less Redundancy required].
4. DBMS has a strong query language. Once the database is designed, it helps the user to retrieve and manipulate the data.

5. Concurrency: Multiple users should be able to access the same database simultaneously, without affecting the other user.

6. It supports multiple views to the user, depending on his role.

7. Database should also provide security at different levels.

8. Database should support ACID property. [Atomicity, Consistency, Isolation & Durability]

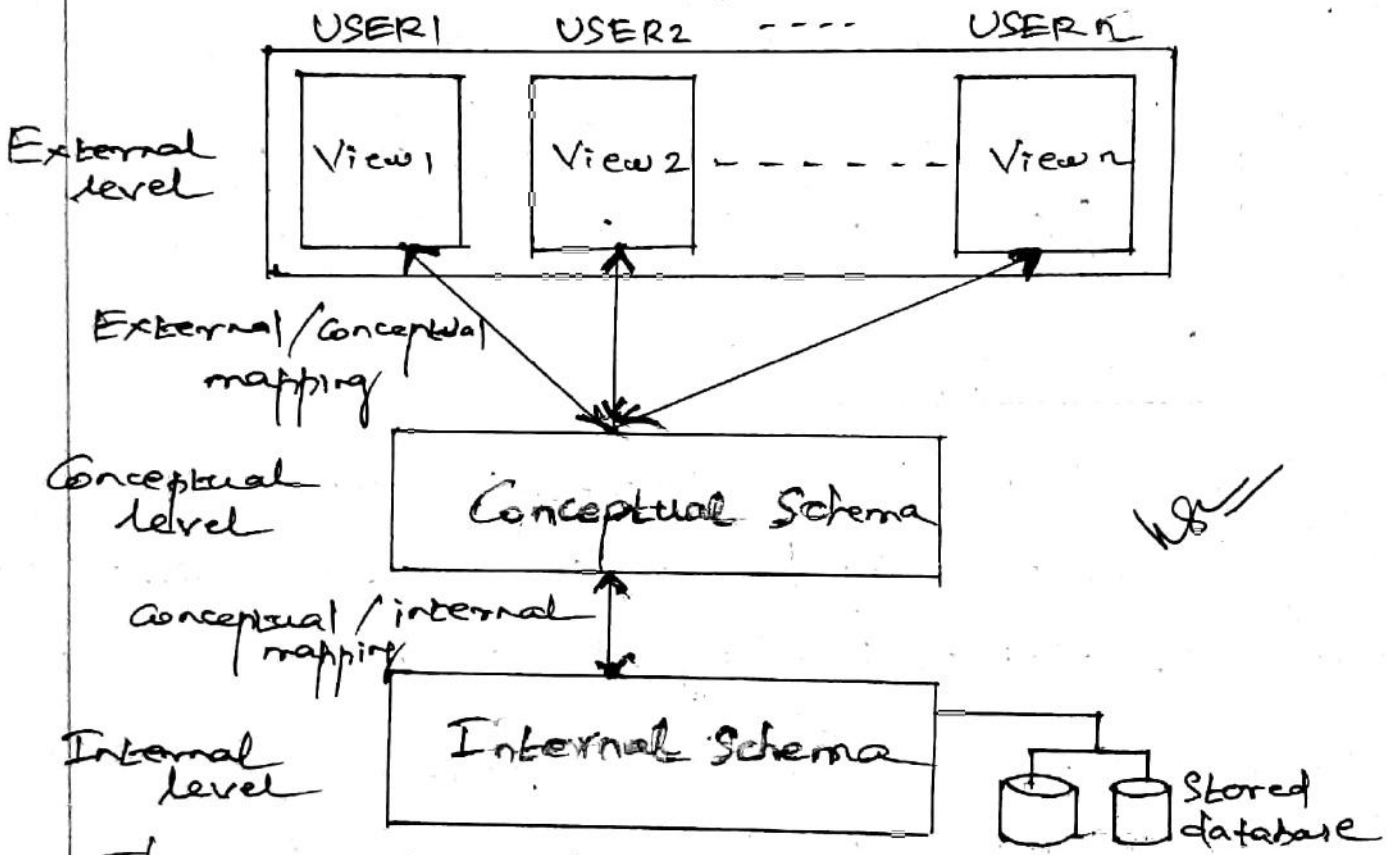
Purpose of database systems

Database management systems were developed to handle the following difficulties of typical, traditional file-processing systems.

1) Data redundancy and inconsistency
2) Difficulty in accessing data
3) Data isolation
4) Integrity problems
5) Atomicity problems
6) Concurrent-access anomalies
7) Security problems

Views of Data [Three-Schema Architecture]

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal complex details from users. This process of hiding complex details from user is called data abstraction.



There are three levels of abstraction

Physical level [Internal level]

This is the lowest level of data abstraction It describes how data is actually stored in database. You can get the complex data structure details at this level.

[The process of transforming requests and results between levels are called mappings]

Logical level [Conceptual level]

This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

View level [External level]

This is the highest level of data abstraction. This level describes the user interaction with database system.

Example

Let's say we are storing customer information in a customer table. At physical level, these records can be described as blocks of storage (bytes, gigabytes, terabytes etc) in memory. These details are often hidden from the programmers.

At the logical level, these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented.

At view level, the user interact with the system with the help of GUI and enter

the details at the screen. They are not aware of how the data is stored and what data is stored. Such details are hidden from them.

Data models

A data model provides a way to describe the design of a database at the physical, logical and view levels. The data models can be classified into four different categories:

Relational model:

The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model. Record based models are so named because the database is structured in fixed-format records of several types.

Using certain integrity rules, two different tables can be related with each other using a common field in these tables. In relational model, the relational information can be retrieved by relating a data in one table with other table.

Entity-Relationship model

The entity-relationship (E-R) model uses a collection of entities (tables) and relationships among these tables. The E-R model is widely used in database design.

It helps you to analyze data requirements systematically to produce a well-designed database.

Object-based data model

In object based data models, the focus is on how data is represented. The data is divided into multiple entities each of which have defining characteristics. Moreover, these data entities are connected with each other through relationships.

The object-relational data model combines the features of the object-oriented data model and relational data model.

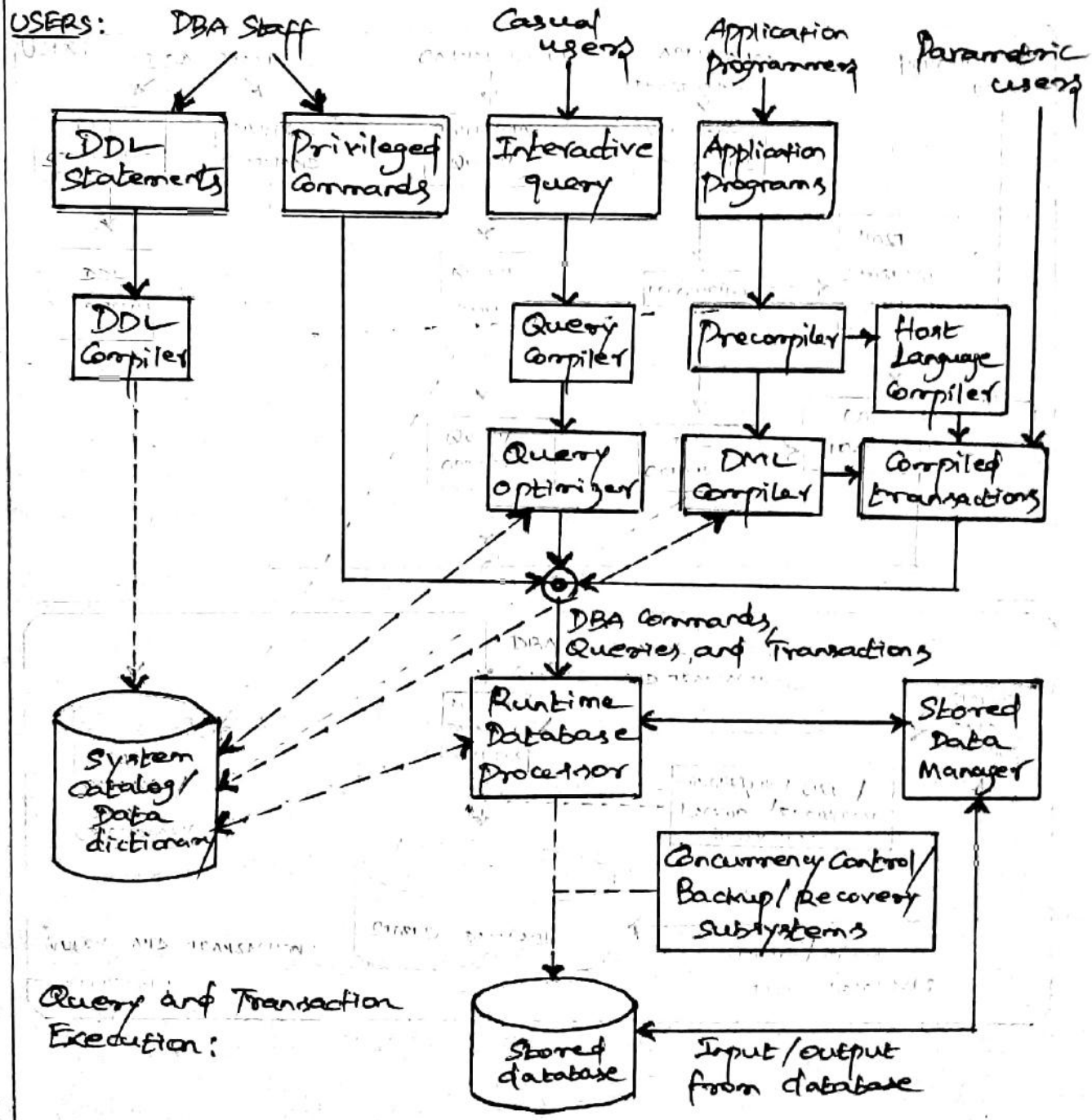
Semi-structured data model

Semi structured data does not have the same level of organization and predictability of structured data. The data does not reside in fixed fields or records, but

Contains elements that can separate the data into various hierarchies.

The Extensible Markup Language (XML) is widely used to represent semi structured data.

Database Architecture



The above figure illustrates the simplified form of the typical DBMS components. The figure is divided into two halves. The top half of the figure refers to various users of

database environment and their interfaces.

The lower half shows the internal structure of DBMS, which is responsible for storage of data and processing of transactions.

The database and the DBMS catalog are usually stored on hard disk. Access to disk is primarily controlled by the operating system.

Stored Data Manager module

It controls the access to DBMS information on disk at a higher level.

DBA staff

They work on defining the db and tuning it by making changes to its definition using DDL & other privileged commands.

DDL Compiler

It process schema definitions specified in DDL and stores the meta-data in the DBMS catalog.

Casual users

They interact with some interactive query interface for the need of information.

Query Compiler

It parses and analyses the queries generated by the casual users using query interface for correctiveness. Then the compiler

compiles it to an internal form which is understandable by the DBMS.

Query optimizer

It attempts to determine the most efficient way to execute a given query by eliminating the redundancies and using of correct algorithms and indexes during execution

Application programmers

They write programs in host languages such as Java, C which will be submitted to a precompiler.

Precompiler

It converts the source program into its appropriate SQL function calls for DBMS.

DML compiler

It converts the SQL commands from precompiler to an internal form of DBMS. Insertion, Deletion, Modification commands are handled here.

Host language compiler Data sub language

After extracting the SQL functions using precompiler, the rest of the program is sent to the host language compiler. It compiles the rest into its equivalent object code. [low level instructions]

The DML commands and the rest of the program are linked, forming a combined transaction to be processed by a runtime database processor.

Parametric Users

They use the combined transactions by supplying parameters. These parameters are called runtime parameters.

(Eg) Bank withdrawal transaction, where the account number and the amount are supplied as parameters.

Runtime Database Processor

It executes the DBA commands, queries and combined transactions with parameters. It works with the system dictionary for updation.

The DB processor works with the stored data manager, which inturn uses basic operating system services for carrying out low-level input/output operations between disk and main memory.

Concurrency control and back-up recovery systems are separately integrated with the working of run-time database processor for transaction management.

Relational Algebra

* It is a procedural query language.
* It consists of a set of operations which take one or two relations as input and produce a new relation as their result.

The fundamental operations are:

1. Select (σ)
2. Project (Π)
3. Union ($\cup$)
4. Set difference ($-$)
5. Cartesian product ($\times$)
6. Rename (ρ)

Fundamental operations

The select, project and rename operations are called unary operations, because they operate on one relation.

The other three operations operate on pairs of relations and therefore called binary operations.

Select operation σ

It selects tuples that satisfy the given predicate [Condition] from a relation.

Notation $\longrightarrow \sigma_p r$

Where

$\sigma \rightarrow$ stands for selection predicate

$r \rightarrow$ relation

$p \rightarrow$ prepositional logic formula which may use connections like and, or and not.

These terms are relational operators like $=, \neq, <, >, \leq, \geq$

Books

subject	author	price	year
Database	Ramez	450	2015
CA	Patterson	475	2014
DS	Allen	550	2005
TPDE	Veera	350	2012
EVS	Gilbert	275	2010

Eg

$\sigma_{subject = "database"}(Books)$

o/p

database	Ramez	450	2015

$\sigma_{subject = "database" \wedge price = "450"}(Books)$

o/p

database	Ramez	450	2015

$\sigma_{subject = "database" \vee price = "450"}(Books)$

o/p

No rows selected.

Project operation (Π)

It projects columns which satisfy a given predicate.

Notation $\longrightarrow \Pi_{A1, A2, \ldots An}\, r$

where, $A_1, A_2 \longrightarrow$ attribute names

$r \longrightarrow$ relation

Duplicate rows are automatically eliminated, as relation is a set.

Eg

$\Pi_{subject, author}$ (Books)

o/p

database	Ramey
CA	Patterson
PDS	Allen
TPDE	Veera
EVS	Gilbert

Subject

sub-code	sub-name	semester	year
CS6301	PDS	Third	Second
CS6302	DBMS	Third	Second
CS6401	OS	Fourth	Second
CS6402	DAA	Fourth	Second
CS6501	IP	Fifth	Third
CS6502	OOAD	Fifth	Third
CS6301	PDS	Fifth	Third

Union operation U

The result of this operation, denoted by R ∪ S, is a relation, which includes all the tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

$\Pi_{sub\text{-}code}(\sigma_{semester = \text{"fifth"} \wedge year = \text{"third"}}(Subject))$

o/p
CS6501
CS6502
CS6301

$\Pi_{sub\text{-}code}(\sigma_{semester = \text{"Third"} \wedge year = \text{"second"}}(Subject))$

o/p
CS6301
CS6302

$\Pi_{sub\text{-}code}(\sigma_{semester = \text{"fifth"} \wedge year = \text{"Third"}}(Subject)) \cup \Pi_{sub\text{-}code}(\sigma_{semester = \text{"third"} \wedge year = \text{"second"}}(Subject))$

o/p
CS6501
CS6502
CS6301
CS6302

Duplicate tuple containing CS6301 is eliminated.

Set difference (minus)

The result of this operation is denoted by R-S, is a relation which includes all tuples that are in R but not in S.

Eg

$\Pi_{sub\text{-}code}(\sigma_{semester = \text{"Fifth"} \wedge year = \text{"third"}}(Subject)) - \Pi_{sub\text{-}code}(\sigma_{semester = \text{"Third"} \wedge year = \text{"second"}}(Subject))$

op
CS6501
CS6502

Intersection

The result of this operation, denoted by $R \cap S$, is a relation, which includes all tuples that are in both R and S.

$\Pi_{sub\text{-}code}(\sigma_{semester = "Fifth" \wedge year = "Third"}(Subject)) \cap \Pi_{sub\text{-}code}(\sigma_{semester = "Third" \wedge year = "Second"}(Subject))$

o/p

CS6301

Cartesian Product X

The cross product (or) Cartesian product operation returns all possible combinations of rows in r with rows in s.

In other words, the result is every possible pairing of the rows of r and s.

Notation → $r \times s$

where r and s are relations and their o/p will be defined as

$$r \times s = \{qt \mid q \in r \text{ and } t \in s\}$$

r

A	B
α	1
β	2

s

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

$\gamma \times \beta$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Rename Operation ρ

The results of relational algebra are also relations but without any name. The rename operation allows to rename the output relation.

Notation → $\rho_x E$

where the result of expression E is saved with the name of x.

$\rho_{OldName = NewName}(\gamma)$.

OldName → Name of the attribute

NewName → New name of the attribute

γ → relation

γ

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

Eg

ρ (myRelation, (r-s))

A	B
α	1
β	1

→ myRelation

ρ (myRelation (A → A2), (r-s))

myRelation

A2	B
α	1
β	1

Relational databases

A relational database consists of a collection of tables, each of which is assigned a unique name.

A table is referred to as a relation in the sense that it is a collection of objects of the same type (rows). Data in a table can be related according to common keys or concepts, and the ability to retrieve related data from a table is the basis for relational database.

Integrity rules

1. The rows in a relational table should be distinct (unique).

2. The column value must not be repetitive.
3. There should not be any null value. To make each row unique, primary key is used.

Employee

Emp_no	Name	Place	Car_no
1001	Ram	Chennai	5
1083	Ravi	Trichy	12
1099	Rajesh	Madurai	77
1035	Arul	Chennai	45

Cars

Car_no	License_Plate	Mileage	Year
5	ABC123	5000	1996
12	DFE123	7500	1999

A distinguished feature of a relational database is that it is possible to get data from more than one tables.

It is possible to get the names of employees, who have cars with their license_no, mileage and year using relational database.

There must be one column which appears in both tables in order to relate them to each other. That must be primary key in one table which will be foreign key in other table.

Here Car_no is the primary key in cars table and foreign key in employee table.

If Car_no 5 is deleted from cars table, then the row of Car_no 5 should automatically be removed from Employee table.

Query

Select Employee.Name, Cars.Licence_plate, Cars.Mileage, Cars.Year from Employee, Cars where Employee.Car_no = Cars.Car_no

Name	License_plate	mileage	Year
Ram	ABC123	5000	1996
Ravi	DFE123	7500	1999

Database Keys

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

Super key

It is defined as a set of attributes within a table that can uniquely identify each record within a table.

Consider the table below

Student

Student_id	Name	Phone	Age
1	Raj	8978767543	15
2	Ravi	9845678900	18
3	Ram	9900800801	17

Super keys — Student_id①, phone②, {student_id③, name}, {student_id④, name, phone}, {Student_id⑤, name, phone, age}, {phone⑥, name}, {phone⑦, age}, {phone⑧, name, age}

Candidate key

The minimal set of super keys which can uniquely identify a tuple is called candidate key.

∴ From the above 8 super keys available, minimal set of super keys are student_id & phone.

Candidate keys — {student_id}, {phone}

Primary key

Any one key which can be chosen from the available candidate keys to make a tuple unique in a table is called primary key.

Primary keys — {Student_id} or
{phone}

Alternate key (Secondary key)

The candidate key other than the primary key is called alternate key.

Alternate key — If Student_id is P.K, phone is A.K
If phone is P.K, Student_id is A.K

Foreign key

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

Embedded SQL

SQL statements can be embedded into general purpose programming language such as C, Java, .NET, PHP. The programming language is called host language.

An embedded SQL statement is distinguished from programming language statements by prefixing it with the keywords EXEC SQL, so that

Preprocessor (or precompiler) can separate embedded SQL statements from the host language code. The SQL statements can be terminated by a semicolon (;) or a matching END EXEC.

Within the 'c' language to embed SQL code, some special variables are used which are called "shared variables". These variables can be used in both 'c' program and the embedded SQL statements. Shared variables are prefixed by a colon (:), when they appear in SQL statements.

Consider the example which has a C program, to process the COMPANY database. We need to declare program variables to match the types of database attributes that the program will process.

Shared variables are declared within declare section in the program

SQL data types INTEGER, SMALL INT, REAL & DOUBLE are mapped into C types long, short, float & double respectively. Fixed-length & varying length strings (CHAR[i], VARCHAR[i]) in SQL can be mapped to arrays of characters

(char[i+1], varchar[LEN+1]) in C that are one character long than SQL type.

The variables SQLCODE & SQLSTATE are used to communicate errors and exception conditions between database and program.

After each database command is executed, DBMS returns a value in SQLCODE. '0' value indicates the execution of SQL statement is successful. If SQLCODE > 0, indicates no more records are available. If SQLCODE < 0, indicates some error has occured.

A value of '000000' in SQLSTATE indicates no error (or) exceptions, other values indicate errors (or) exceptions.

The program reads (inputs) a PANNO value and then receives/retrieves the EMPLOYEE tuple with PAN from the database via the embedded SQL command. The INTO clause specifies the shared variables into which the attribute values of database are retrieved.

```
CONNECT TO <Server name> AS <connection name>
AUTHORIZATION <User account name and
                                password>;
int loop;
EXEC SQL BEGIN DECLARE SECTION;
Varchar dname[16], fname[16], lname[16],
        address[31];
char pan[10], gender[2];
float salary, raise;
int dno;
int SQLCODE;
char SQLSTATE[6];
EXEC SQL END DECLARE SECTION;
loop = 1;
while (loop) {
prompt("Enter PAN number:", pan);
EXEC SQL
Select fname, lname, address, salary into
  :fname, :lname, :address, :salary from
EMPLOYEE where pan = :pan;
if (SQLCODE==0) printf(fname, lname, address,
                       salary);
else printf("PAN number does not exist:", pan);
Prompt("More PAN numbers (Enter 1 for YES,
        0 for NO):", loop);
}
```

SQL TYPES INTEGER, SMALL INT, REAL & DOUBLE are mapped into C types long, short, float and double.

CHAR[i], VARCHAR[i] in SQL can be mapped into arrays of characters (Char[i+1], Varchar[i+1]) in C.

SQLCODE = 0 → Successful SQL execution
SQLCODE > 0 → No more SQL records available.
SQLCODE < 0 → Some error occured

Dynamic SQL

When the pattern of database access is known in advance then static SQL is very adequate. Sometimes, in many applications, we may not know the pattern of database access in advance. It requires an advanced form of static SQL known as dynamic SQL.

Using host variables, we can achieve a little bit of dynamicness in embedded SQL (Static SQL).

Eg

```
exec sql select fname, gender from teacher
                    where salary > :sal;
```

Here the salary will be asked on run time. But getting column name (or) table name at run time is not possible with embedded SQL. For having such feature, dynamic SQL is needed.

Dynamic SQL Concepts

* In dynamic SQL, the SQL statements are not hard coded in the programming language. The text of the SQL statement is asked at run time.
* In dynamic SQL, the SQL statements that are to be executed are not known until run time. So DBMS can't get prepared for executing the statements in advance.

Dynamic statement Execution (Execute Immediate)

The Execute Immediate statement provides the simplest form of dynamic SQL. This statement passes the text of SQL statements to DBMS and asks the DBMS to execute the SQL statements immediately.

For using the statement our program goes through the following steps.

1. The program constructs a SQL statement as a string of text in one of its data areas (called a buffer).
2. The program passes the SQL statements to the DBMS with the EXECUTE IMMEDIATE statement.
3. The DBMS executes the statement and sets the SQLCODE / SQLSTATE values to flag the finishing status same like, if the statement had been hard coded using static SQL.

SET OPERATIONS

UNION, INTERSECTION & MINUS

∪ → union
∩ → Intersection
− → Minus / set-difference

STUDENT

Fn	Ln
Ganesh	Balaji
Ram	Kumar
Arun	Prasad
Alfred	Paul
Sai	Tarun

INSTRUCTOR

Fname	Lname
John	Smith
Ramey	Elmasri
Alfred	Paul
Ram	Kumar

STUDENT ∪ INSTRUCTOR

Fn	Ln
Ganesh	Balaji
Ram	Kumar
Arun	Prasad
Alfred	Paul
Sai	Tarun
John	Smith
Ramey	Elmasri

STUDENT ∩ INSTRUCTOR

Fn	Ln
Alfred	Paul
Ram	Kumar

STUDENT − INSTRUCTOR

Fn	Ln
Ganesh	Balaji
Arun	Prasad
Sai	Tarun

INSTRUCTOR − STUDENT

Fname	Lname
John	Smith
Ramey	Elmasri

Prepared by

K. Kaiser
Verified by

24/4/17
Approved by

Entity-Relationship model – E-R Diagrams – Enhanced-ER Model – ER-to-Relational Mapping – Functional Dependencies – Non-loss Decomposition – First, Second, Third Normal Forms, Dependency Preservation – Boyce/Codd Normal Form – Multi-valued Dependencies and Fourth Normal Form – Join Dependencies and Fifth Normal Form

22/11/17

Entity-Relationship Model (E-R model)

E-R model is a high level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them.

ER modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

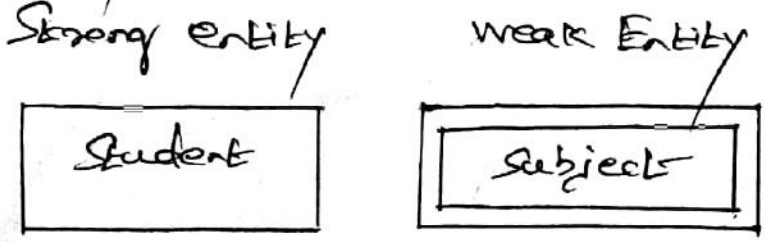
ER diagrams

E-R diagrams displays the relationships of entity set stored in a database. E-R diagrams help you to explain the logical structure of databases. E-R diagram includes special symbols, and its meanings make this model unique.

Components of E-R diagram

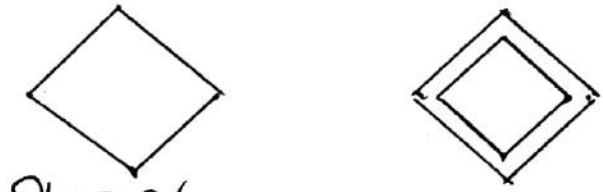
Entity, Attributes, Relationships etc form the components of E-R diagrams.

Entity



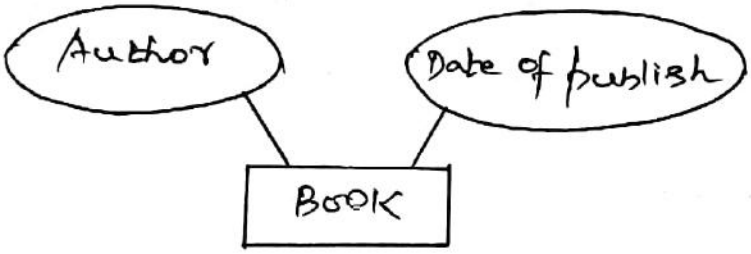
Entity which doesn't have any key attrib of its own is weak entity.

Relationships between Entities



Relationship between a strong entity and a weak entity is weak relationship

Attributes for an entity



Key attribute (prime key attribute)

Derived attribute

Derived attributes are those which are derived based on other attributes.
Eg. Age derived from date of birth.

Multi-Valued attribute

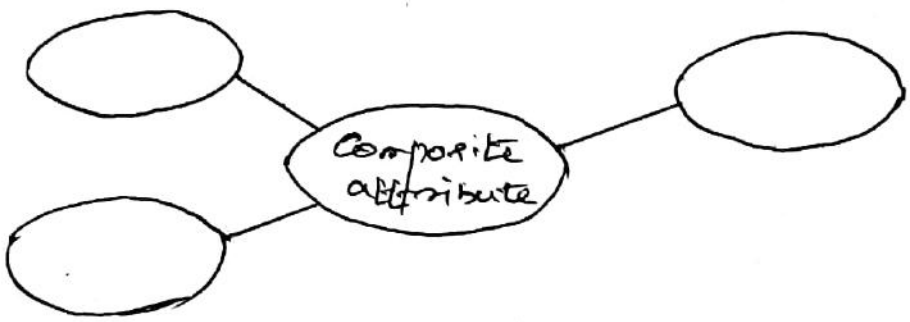
Attribute which has more than one value.

Eg: Mobile_no

Composite attribute

An attribute which is a combination of two or more attributes.

Eg: Address : State, City, Zip



Relationship

A relationship describes relation between entities.

Degree of relationship

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

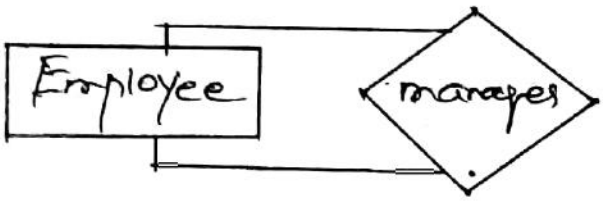
1. Unary [Recursive relationship]
2. Binary relationship
3. Ternary relationship

Unary relationship [Recursive relationship]

When there is only one entity participating in a relationship, it is unary relationship.

Eg: A person marries a person

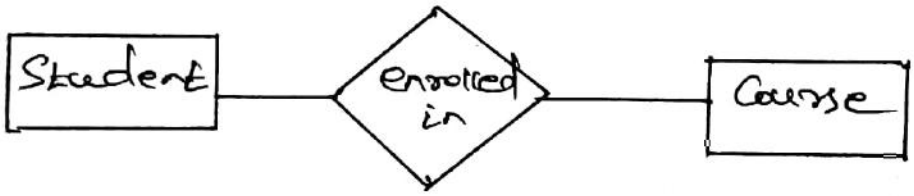
An employee manages an employee



Binary relationship

When there are two entities participating in a relationship, the relationship is called as binary relationship.

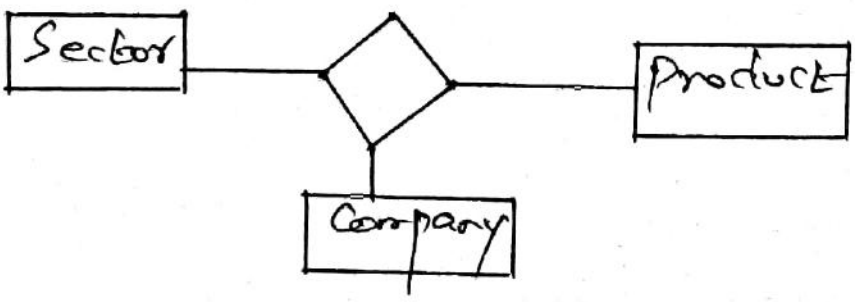
Eg: Student enrolled in course



Ternary relationship [n-ary relationship]

When there are n-ary entities participating in a relationship, the relationship is called ternary relationship

Eg: Company operates in Sector, producing products



Cardinality of relationship

The number of times an entity participates in relationship with other entities is cardinality. They are

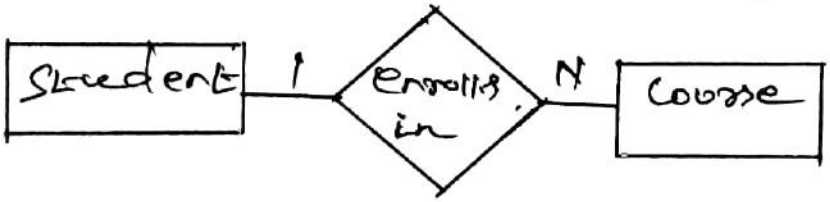
1. One to one relationship
2. one to many relationship
3. Many to one relationship
4. Many to many relationship

One to one relationship

Eg: One student enrolls in one course

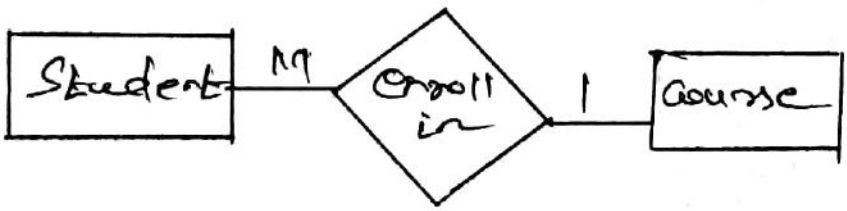
One to many relationship

Eg: one student enrolls in many courses



Many to one relationship

Eg: Many students enrolled in one course



Many to many relationship

Eg: Many courses enrolled by many students
Many students enroll in many courses

Entity Relationship (ER) Modeling -

Here we are going to design an Entity Relationship (ER) model for a college database . Say we have the following statements.

1. A college contains many departments
2. Each department can offer any number of courses
3. Many instructors can work in a department
4. An instructor can work only in one department
5. For each department there is a Head
6. An instructor can be head of only one department
7. Each instructor can take any number of courses
8. A course can be taken by only one instructor
9. A student can enroll for any number of courses
10. Each course can have any number of students

Good to go. Let's start our design.(Remember our previous topic and the notations we have used for entities, attributes, relations etc)

Step 1 : Identify the Entities

What are the entities here?

From the statements given, the entities are

1. Department
2. Course
3. Instructor
4. Student

Step 2 : Identify the relationships

1. One department offers many courses. But one particular course can be offered by only one department. hence the cardinality between department and course is One to Many (1:N)
2. One department has multiple instructors . But instructor belongs to only one department. Hence the cardinality between department and instructor is One to Many (1:N)
3. One department has only one head and one head can be the head of only one department. Hence the cardinality is one to one. (1:1)
4. One course can be enrolled by many students and one student can enroll for many courses. Hence the cardinality between course and student is Many to Many (M:N)
5. One course is taught by only one instructor. But one instructor teaches many courses. Hence the cardinality between course and instructor is Many to One (N :1)

Step 3: Identify the key attributes

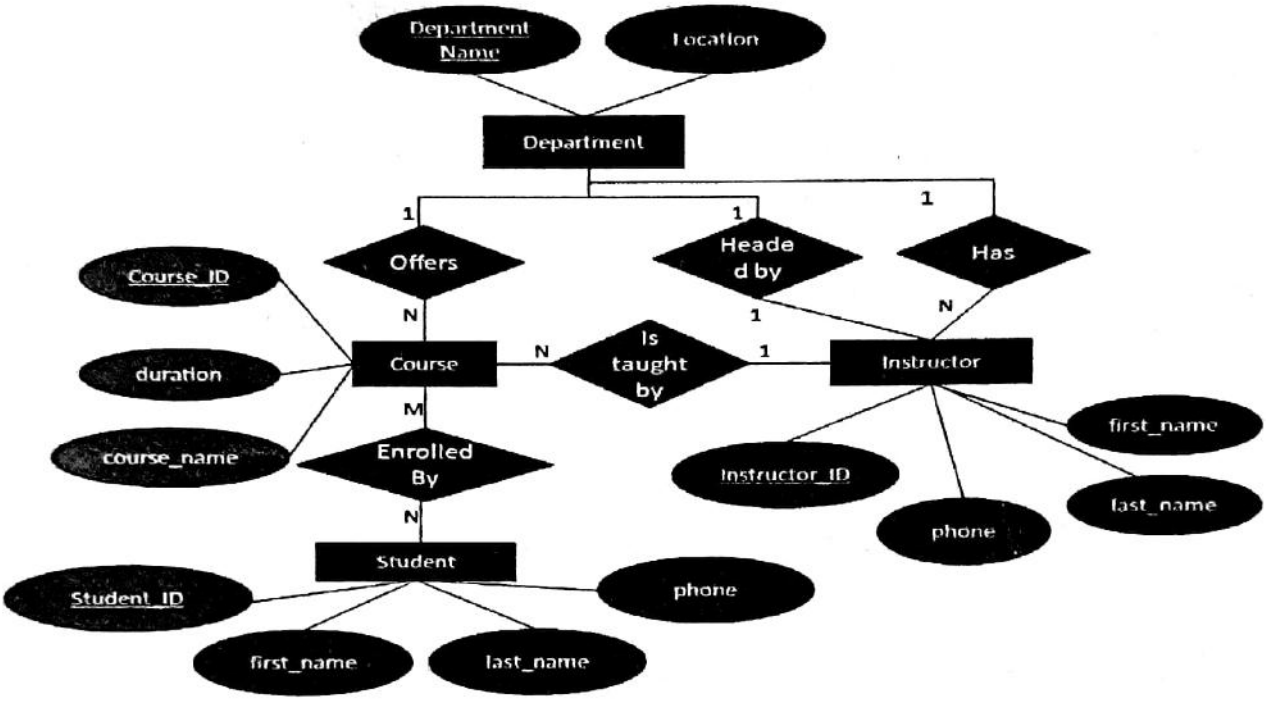
- "Departmen_Name" can identify a department uniquely. Hence Department_Name is the key attribute for the Entity "Department".
- Course_ID is the key attribute for "Course" Entity.
- Student_ID is the key attribute for "Student" Entity.
- Instructor_ID is the key attribute for "Instructor" Entity.

Step 4: Identify other relevant attributes

- For the department entity, other attributes are location
- For course entity, other attributes are course_name,duration
- For instructor entity, other attributes are first_name, last_name, phone
- For student entity, first_name, last_name, phone

Step 5: Draw complete ER diagram

By connecting all these details, we can now draw ER diagram as given below.



Enhanced Entity Relationship Model (EER Model)

EER is a high level data model that incorporates the extensions to the original ER model.

It is a diagrammatic technique for displaying the following concepts

* Sub class and super class
* Specialization and generalization
* Union or Category
* Aggregation

Features of EER Model

* It creates a design more accurate to database schemas.
* It reflects the data properties and constraints more precisely.
* It includes all modeling concepts of the ER model
* Diagrammatic technique helps for displaying the EER schema.
* It includes the concept of specialization and generalization.
* It is used to represent a Collection of objects (i.e) Union of objects of different entity types.

Sub class and Super class

* Sub class and super class relationship leads the concept of inheritance.

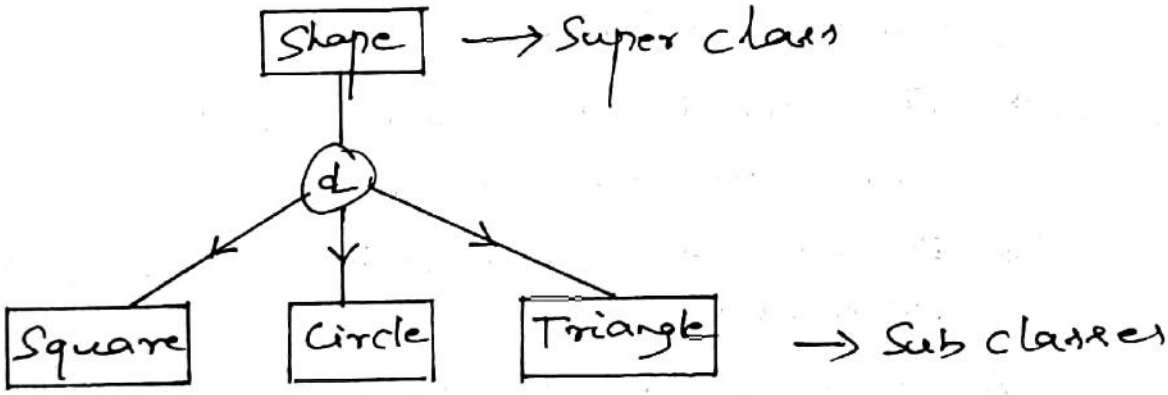
* The relationship between sub class and super class is denoted with (d) symbol.

Super class

* Super class is an entity type that has a relationship with one or more sub-types.
* An entity cannot exist in database merely by being member of any super class.

Sub class

* Sub class is a group of entities with attributes
* It inherits properties and attributes from its super class.

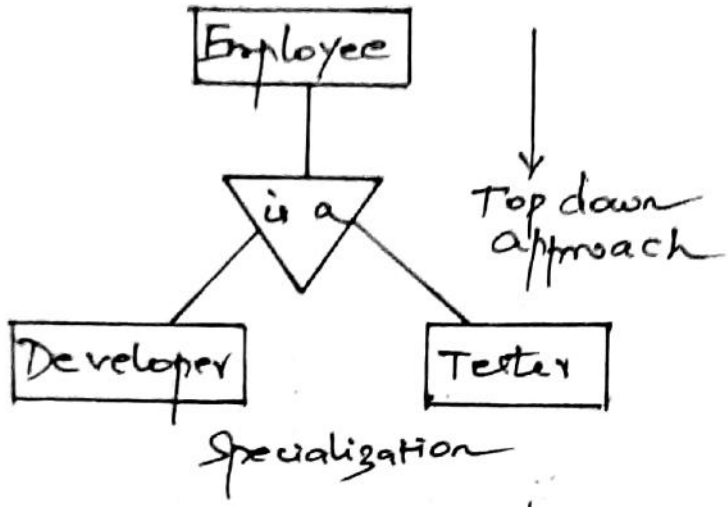


Specialization

* It is a process which defines a group of entities which is divided into sub groups based on their characteristics.
* It is a top down approach, in which one higher entity can be broken into two lower level entity
* It maximizes the difference between the members of an entity by identifying the unique

characteristics or attributes of each member.

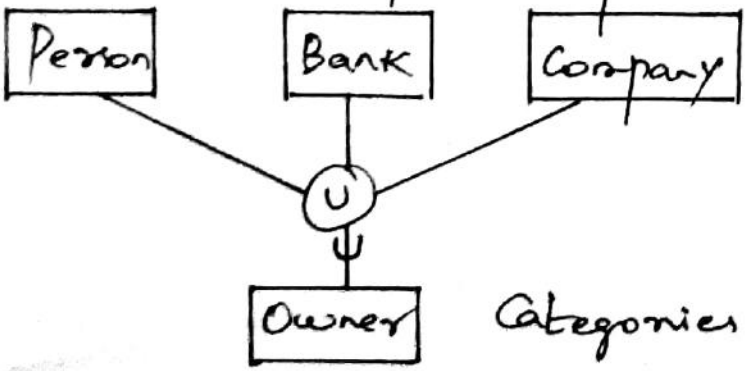
* It defines one or more sub class for the super class and also forms the super class/ sub class relationship.



Category or Union

* Category represents a single super class or sub class relationship with more than one super class.
* It can be a total or partial participation.

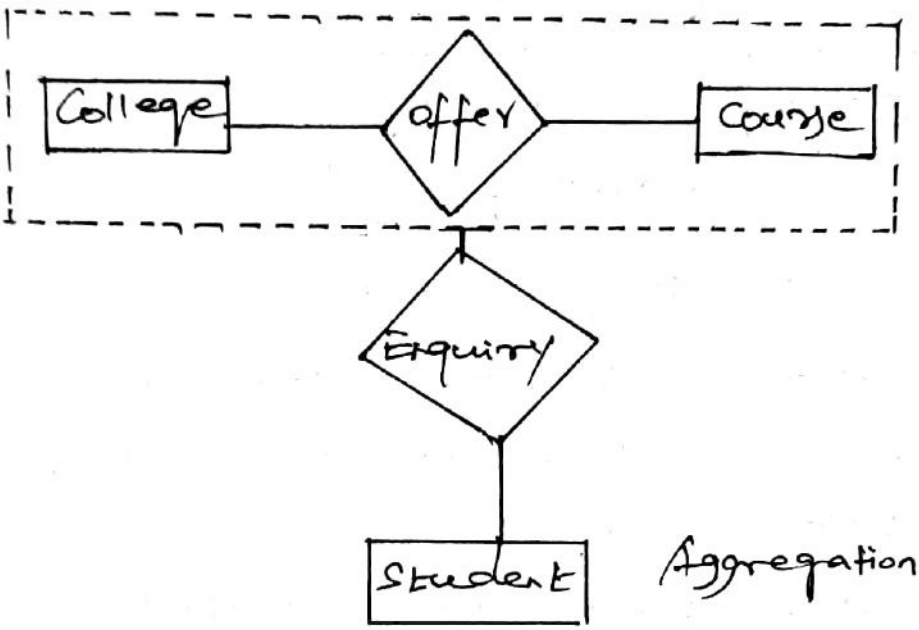
For example, Car booking: Car owner can be a person, a bank (holds a possession on a car) or a company. Category (sub class) → Owner is a subset of the union of the three super classes → Company, Bank and Person. A category member must exist in at least one of its super classes.



Categories (Union type)

Aggregation

* It is a process that represents a relationship between a whole object and its component parts.
* It abstracts a relationship between objects and viewing the relationship as an object.
* It is a process when two entity is treated as a single entity.



Aggregation

In the above example, the relation between College and Course is acting as an Entity in relation with Student.

Functional Dependency

Functional dependency is a term derived from mathematical theory, which states that for every element in the attribute (which appears on some row), there is a unique corresponding element (on the same row).

Let us assume that rows (tuples) of a relational table T is represented by the notation r1, r2, ... and individual attributes (columns) of the table is represented by letters A, B, ...

We can say that A → B, A functionally determines B (or) B is functionally dependent on A. In other words, we can say that, given two rows R1 and R2, in table T, if R1(A) = R2(A), then R1(B) = R2(B).

A can sometimes called as determinant whereas B is called dependent.

The following example illustrates the concept of functional dependency.

STUDENT

StudentID	Semester	Subject	Lecturer
1234	6	AI	Arun
1221	4	DBMS	Rajesh
1234	6	TOC	Peter
1201	2	BEEE	Ravi
1201	2	MII	Ram

We notice that whenever two rows in this table feature the same StudentID, they also necessarily have the same semester values. This basic fact can be expressed by a functional dependency,

StudentID ⟶ Semester

If you know the StudentID, you can definitely know the semester.

Decomposition

1. Decomposition is the process of breaking down in parts or elements
2. It replaces a relation with a collection of smaller relations.
3. It breaks the table into multiple tables in a database.

4. It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.

5. If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

6. Decomposition helps in eliminating some of the bad design problems such as redundancies, inconsistencies and anomalies.

There are two types of decomposition:

1. Lossy decomposition
2. Lossless join decomposition

Lossy decomposition

The decomposition of relation R into R1 and R2 is lossy, when the join of R1 and R2 does not yield the same relation as in R.

Consider the table STUDENT:

STUDENT

Roll_no	Sname	Dept
111	Kumar	Computer
222	Kumar	Electrical

This relation is decomposed into two relations No_name and Name_dept

No_NAME NAME_DEPT

NO_NAME

Roll_No	Sname
111	Kumar
222	Kumar

NAME_DEPT

Sname	Dept
Kumar	Computer
Kumar	Electrical

In lossy decomposition, spurious tuples are generated when a natural join is applied to the relations in the decomposition.

STU_JOINED STU_JOINED

Roll_No	Sname	Dept
111	Kumar	Computer
111	Kumar	Electrical
222	Kumar	Computer
222	Kumar	Electrical

The above decomposition is a bad decomposition or lossy decomposition

Spurious tuples — It is a record in a database which is created when two tuples are joined badly [without primary key or foreign key]

Lossless join Decomposition [Non-loss decomposition]

The decomposition of a relation R into R1 and R2 is lossless, when the join of R1 and R2 yield the same relation as in R.

If the Student table is decomposed into two relation Stu_name and Stu_dept:

STU_NAME

Roll_no	Sname
111	Kumar
222	Kumar

STU_NAME

STU_DEPT

STU_DEPT

Roll_no	Dept
111	Computer
222	Electrical

When these two relations are joined on the common attribute Roll_no [primary key], the resultant relation will look like the original Student table.

STU_JOINED

STU_JOINED

Roll_no	Sname	Dept
111	Kumar	Computer
222	Kumar	Electrical

In lossless decomposition, no spurious tuples are generated when a natural join is applied to the relations.

Dependency Preservation

If we decompose a relation R into relations R1 and R2, All dependencies of R either must be a part of R1 or R2 (or) must be derivable from combination of FD's of R1 and R2

For eg.

A relation R(A, B, C, D) with FD set $\{A \rightarrow BC\}$ is decomposed into R1(ABC) and R2(AD) is dependency preserving because FD $\{A \rightarrow BC\}$ is a part of R1(ABC)

Eg 2 R(A, B, C, D) order $f = \{A \rightarrow B, B \rightarrow C\}$

Decomposition is R1(AB), R2(AC) and R3(AD)

FD $\{A \rightarrow B\}$ is covered in R1(AB), but FD $\{B \rightarrow C\}$ is uncovered in the decomposition.

∴ The decomposition is not dependency preserving

If it is decomposed into R1(AB), R2(BC) & R3(AD) then,

FD $\{A \rightarrow B\}$ is covered (preserved) in R1(AB)

FD $\{B \rightarrow C\}$ is covered in R2(BC).

Hence the decomposition is dependency preserving.

Need for Normalization

Anomalies in DBMS

There are 3 types of anomalies that occur when the database is not normalized. There are insertion, updation and deletion anomaly.

Eg. Consider the following relation

EMPLOYEE EMPLOYEE

Emp_id	Emp_name	Emp_address	Emp_dept
101	Raj	Delhi	D001
101	Raj	Delhi	D002
123	Ravi	Agra	D890
166	Kumar	Chennai	D900
166	Kumar	Chennai	D004

The above table is not normalized. The following problems exist when a table is not normalized.

Update anomaly:

In the above table, we have two rows for employee Raj, as he belongs to two departments. If we want to update the address of Raj, then we have to update the data in both rows, or the data will become inconsistent.

If somehow the correct address gets

updated in one department, but not in other department, then as per the database, Raj would have two different addresses, which is incorrect and would lead to inconsistent data.

Insert anomaly:

Suppose a new employee joins the company, who is currently under training and not assigned to any department, then we would not be able to insert the data into the attribute emp_dept, since null values cannot be allowed.

Delete anomaly:

Suppose, if at a point of time, the company closes the department D890, then deleting the rows that have emp_dept as D890 would also delete the information of employee Ravi, since he is assigned only to this department.

To overcome these anomalies, we need to normalize the data.

Normalization

It is the process of removing all kinds of anomalies from database.

Various normalization forms are

1. First Normal form (1NF)

2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce Codd Normal Form (BCNF)
5. Fourth Normal Form (4NF) — MVD
6. Fifth Normal Form (5NF) — MVD

First Normal Form (1NF)

"A relation R is in 1NF, if it does not have any composite attributes, multi-valued attribute or their combination." In other words,

"All attributes (column) in the entity (table) must be single valued."

Repeating or multi-valued attributes are moved into a separate entity (table) and a relationship is established between the two tables or entities.

CUSTOMER CUSTOMER

cid	name	Address		Contact_no
		street	city	
C01	aaa	ABC colony	chennai	{12345678}
C02	bbb	xyz colony	Delhi	{123, 333, 456}
C03	ccc	Lloyds street	Bangalore	

The above table is not normalized.

Solution for Composite attribute

Insert separate attributes in a relation for each sub-attribute of a composite attribute

CUSTOMER CUCTOMER

cid	name	Street	city	Contact_no
C01	aaa	ABC colony	Chennai	{123456789}
C02	bbb	xyzy colony	Delhi	{23, 333, 456}
C03	ccc	Lloyds street	Bangalore	

Solution for multi-valued attribute

Remove the multi-valued attribute that violates INF and place it in a separate relation along with the primary key of given original relation.

CUSTOMER CUSTOMER

cid	name	Street	city
C01	aaa	ABC colony	Chennai
C02	bbb	xyzy colony	Delhi
C03	ccc	Lloyds street	Bangalore

CUSTOMER_CONTACT CUSTOMER_CONTACT

Cid	Contact_No
C01	123456789
C02	123
C02	333
C02	456

The above tables are now in normalized form. (1NF)

Second Normal Form (2NF)

A table is said to be in 2NF, if it holds the following two conditions.

1. Table should be in 1NF.
2. No non-prime attribute is dependent on the proper subset of any candidate key of the table. [dependent on part of primary key]

Consider the following relation:

TEACHER TEACHER

Teacher_id	Subject	Teacher_age
111	Maths	38
111	physics	38
222	Biology	38
333	physics	40
333	chemistry	40

Candidate keys: {teacher_id, subject}

Non-prime attribute: teacher_age

The above table is in 1NF, because no multivalued attributes are present. All the attributes contains only one value. [Atomic value]

However, it is not in 2NF, because non-prime attribute teacher_age is dependent on teacher_id alone, which is a proper subset of candidate key. This violates the rule for 2NF, as the rule says "no non-prime attribute is dependent on the proper subset of any candidate key".

To make the table satisfies 2NF, the relation is split into two tables like this:

TEACHER_DETAILS TEACHER_SUBJECT

TEACHER_DETAILS

Teacher_id	Teacher_age
111	38
222	38
333	40

TEACHER_SUBJECT

Teacher_id	Subject
111	maths
111	physics
222	Biology
333	physics
333	Chemistry

Now the table is in 2NF.

The non-prime attribute teacher_age is fully dependent on primary key teacher_id, and no subset of candidate key.

Third Normal Form (3NF)

A table is said to be in 3NF, if it contains the following conditions:

1. It should be in 2NF.
2. Transitive functional dependency should be removed. [Every non-prime attribute of a table must be dependent only on primary key. In other words, a non-prime attribute should not be dependent on another non-prime attribute].

Transitive functional dependency.

A functional dependency is said to be transitive, if it is indirectly formed by two functional dependencies. For eg.

$X \rightarrow Z$ is a transitive dependency, if the following three functional dependencies hold true:

1) $X \rightarrow Y$
2) Y does not $\rightarrow X$
3) $Y \rightarrow Z$

Eg

Book	Author	Age
DBMS	Elmasri	66
CA	Moris Mano	49
Java	Herbert Schildt	66

{Book} ⟶ {Author} [If we know the book, we know the author name]

{Author} does not ⟶ {Book}

{Author} ⟶ {Age}

∴ As per the rule of transitive dependency, {Book} ⟶ {Age}. If we know the book name we can know the author's age.

3NF

Consider the following table:

STUDENT_DETAIL

STUDENT_DETAIL

Student_id	Student_name	DoB	Street	City	Zip

In this table Student_id is the primary key, Non prime attribute Student_name, DOB depends on Student_id, but Street and City depends on Zip [non-prime attribute]. The dependency between Zip and other fields is called transitive dependency. Hence to apply 3NF, we need to move Street, City to new table with Zip as primary key

STUDENT_DETAIL

STUDENT_DETAIL

Student_id	Student_name	DoB	Zip

ADDRESS

ADDRESS

Zip	Street	City

Now the relations are in 3NF.

Boyce Codd Normal Form (BCNF) – 3.5NF

The official qualifications for BCNF are:

1. A table is already in 3NF.
2. All determinants must be superkeys.

All determinants that are not superkeys are removed to place in another table.

A relational schema R is considered to be in **Boyce–Codd normal form (BCNF)** if, for every one of its dependencies X → Y, one of the following conditions holds true:

- X → Y is a trivial functional dependency (i.e., Y is a subset of X)
- X is a superkey for schema R

Example

Let's take a look at this table, with some typical data. The table is not in BCNF.

Author	Nationality	Book title	Genre	Number of pages
William Shakespeare	English	The Comedy of Errors	Comedy	100
Markus Winand	Austrian	SQL Performance Explained	Textbook	200
Jeffrey Ullman	American	A First Course in Database Systems	Textbook	500
Jennifer Widom	American	A First Course in Database Systems	Textbook	500

The nontrivial functional dependencies in the table are:

```
author → nationality
book title → genre, number of pages
```

We can easily see that the only key is the set {author, book title}.

The same data can be stored in a BCNF schema. However, this time we would need three tables.

Author	Nationality
William Shakespeare	English
Markus Winand	Austrian
Jeffrey Ullman	American
Jennifer Widom	American

Book title	Genre	Number of pages
The Comedy of Errors	Comedy	100
SQL Performance Explained	Textbook	200
A First Course in Database Systems	Textbook	500

Author	Book title
William Shakespeare	The Comedy of Errors
Markus Winand	SQL Performance Explained
Jeffrey Ullman	A First Course in Database Systems
Jennifer Widom	A First Course in Database Systems

The functional dependencies for this schema are the same as before:

```
author → nationality
book title → genre, number of pages
```

The key of the first table is {author}. The key of the second table is {book title}. The key of the third table is {author, book title}. There are no functional dependencies violating the BCNF rules, so the schema is in Boyce-Codd normal form.

Fourth Normal Form

A table is in the fourth normal form (4NF) if:

- It is in BCNF.
- It does not have any independent multi-valued parts of the primary key.

Let's say we have table Teacher which gives information about:

1. A teacher can teach many teachers.
2. A teacher may know many languages.

Table: **Teacher**

Teacher_Name	Teacher_Subject	Teacher_Language
Narendra	Science	Hindi
Narendra	Maths	Hindi
Narendra	Science	English
Narendra	History	English
Alok Sharma	Science	Hindi
Alok Sharma	Physical Education	English

We can see that Narendra is teaching three subjects and knows two languages, thus there are two independent multi-valued dependencies. We can split the table into two tables.

Table: TeacherSubject

Teacher_Name	Teacher_Subject
Narendra	Science
Narendra	Maths
Narendra	History
Alok Sharma	Science
Alok Sharma	Physical Education

Table: TeacherLanguage

Teacher_Name	Teacher_Language
Narendra	Hindi
Narendra	English
Alok Sharma	Hindi
Alok Sharma	English

Fifth Normal Form / Projected Normal Form (5NF):

A relation R is in 5NF if and only if every join dependency in R is implied by the candidate keys of R. A relation decomposed into two relations must have loss-less join Property, which ensures that no spurious or extra tuples are generated, when relations are reunited through a natural join.

Properties – A relation R is in 5NF if and only if it satisfies following conditions:

- R should be already in 4NF.
- It cannot be further non loss decomposed (join dependency)

Example – Consider the above schema, with a case as "if a company makes a product and an agent is an agent for that company, then he always sells that product for the company". Under these circumstances, the ACP table is shown as:

Table – ACP

AGENT	COMPANY	PRODUCT
A1	PQR	Nut
A1	PQR	Bolt
A1	XYZ	Nut
A1	XYZ	Bolt
A2	PQR	Nut

The relation ACP is again decomposed into 3 relations. Now, the natural Join of all the three relations will be shown as:

Table – R1

AGENT	COMPANY
A1	PQR
A1	XYZ
A2	PQR

Table – R2

AGENT	PRODUCT
A1	Nut
A1	Bolt
A2	Nut

Table – R3

COMPANY	PRODUCT
PQR	Nut
PQR	Bolt
XYZ	Nut
XYZ	Bolt

Result of Natural Join of R1 and R3 over 'Company' and then Natural Join of R13 and R2 over 'Agent'and 'Product' will be table ACP.

Hence, in this example, all the redundancies are eliminated, and the decomposition of ACP is a lossless join decomposition. Therefore, the relation is in 5NF as it does not violate the property of lossless join.

24/1/15

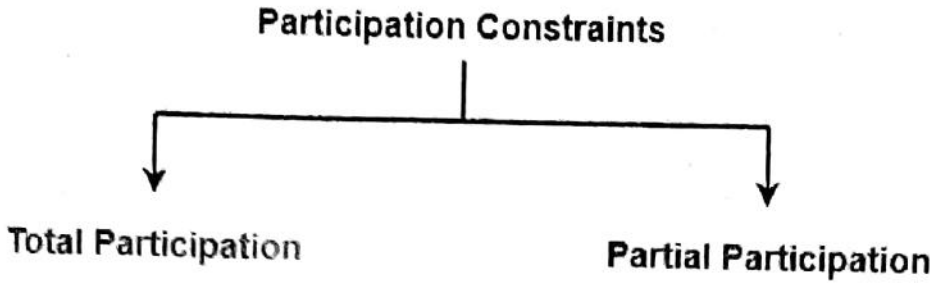
(30)

Prepared by

Verified by

Approved by

There are two types of participation constraints-



1. Total participation
2. Partial participation

1. Total Participation-

- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- That is why, it is also called as **mandatory participation.**
- Total participation is represented using a double line between the entity set and relationship set.

Total Participation

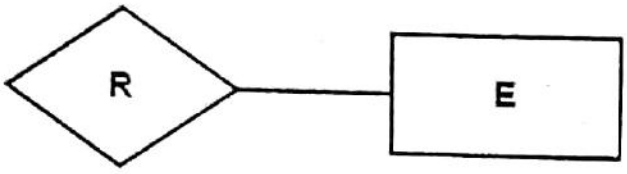
Example-

Here,

- Double line between the entity set "Student" and relationship set "Enrolled in" signifies total participation.
- It specifies that each student must be enrolled in at least one course.

2. Partial Participation-

- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- That is why, it is also called as **optional participation.**
- Partial participation is represented using a single line between the entity set and relationship set.



Partial Participation

Example-

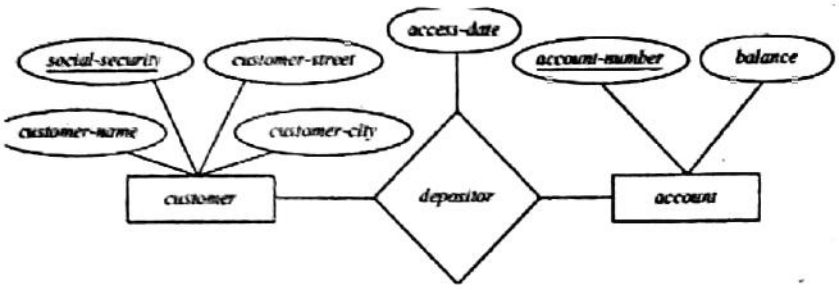
Here,

- Single line between the entity set "Course" and relationship set "Enrolled in" signifies partial participation.
- It specifies that there might exist some courses for which no enrollments are made.

• An entity set is a set of entities of the same type that share the same properties.

• A relationship is an association among several entities Example: Hayes depositor A-102 customer entity relationship set account entity

Relationship Sets (Cont.) • An attribute can also be a property of a relationship set. For instance, the depositor relationship set between entity sets customer and account may have the attribute access-date social-security customer-street customer-name customer-city customer balance depositor



Transaction Concepts – ACID Properties – Schedules – Serializability – Concurrency Control – Need for Concurrency – Locking Protocols – Two Phase Locking – Deadlock – Transaction Recovery – Save Points – Isolation Levels – SQL Facilities for Concurrency and Recovery

Single - user Vs Multi - user systems

DBMS – is a single user, if at most, one user at a time can use the system.

– is a multi user, if many users can use the system, and hence can access the database concurrently.

Multi-programming operating systems

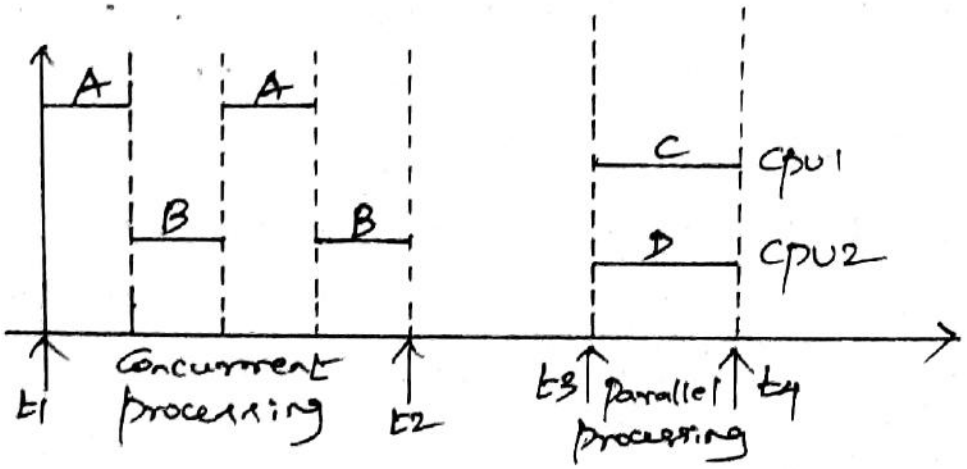
If only a single cpu exists, it can execute at most one process at a time.

But multi-programming OS execute some commands from one process, then suspend and execute other set of commands from other processes

Interleaving process Vs parallel process

In multi-programming, the process is resumed at a point where it was suspended, whenever the process gets the turn to use the cpu again.

This leads to the interleaving of different processes which run concurrently.



In the figure, two processes A and B are executed concurrently in an interleaved manner.

When a process requires an IO operation, such as reading a block from disk, the CPU is switched to execute another process rather than remaining idle during I/O time.

Hence interleaving keeps the processor always busy and also prevents a long process from delaying other processes.

In computers with multiple h/w processors, parallel processing of multiple processes is possible.

Transactions

A transaction is an executing program which forms a logical unit of database processing.

A transaction includes one or more database operations which includes, insertion, deletion, modification or retrieval operations.

One way of specifying the transaction boundaries is by specifying explicit statements

Begin transaction
&
End transaction

```
T1 : read (A)
     A := A - 50
     write (A)
     read (B)
     B := B + 50
     write (B)
```

in an application program. All database operations between these two boundaries are considered as a single transaction. A single application program may contain more than one transaction, if it contains several transaction boundaries.

Consider the basic database access operations that a transaction can include as follows:

read_item(x): Reads a database item named 'x' into a program variable 'x'.

write-item(x): Writes the value of program variable 'x' into the database item named 'x'.

Block → the basic unit of data transfer from disk to main memory is one block.

Executing a read_item(x) command includes the following steps:

1. Find the address of the disk block which contains item X.
2. Copy that disk block into a main memory buffer [If that disk block is not already in some main memory buffer]
3. Copy item X from the buffer to the program variable named x.

Executing a write-item(x) command includes the following steps:

1. Find the address of the disk block which contains item X.
2. Copy that disk block into a buffer in main memory (If that block is not already in some main memory buffer).
3. Copy item x from program variable x into its correct location in the buffer.
4. Store the updated block from the buffer back to disk.

Steps of actually updates the database on disk. In some cases, the buffer is not immediately stored to disk, in case additional changes are to be made to the buffer.

A transaction is a logical unit of work on a database.

Eg:

```
Begin transaction
read_item (x)
write_item (x)
End transaction
```

Properties of Transaction

The properties of a transaction can otherwise be called as ACID properties. They are:

Atomicity

A transaction is an atomic unit of processing. It should be performed either completely (or) not performed at all. There should not be any partial execution of transaction.

Consistency

Transaction should consistently preserve the information. (ie) The transaction should be completely executed from the beginning to end without any interference from other transactions. The information in database should be consistent.

Isolation

Although many transactions are executed concurrently, every transaction should appear that it is being executed in isolation from other transactions.

Durability

The changes applied to a database by a committed transaction should be permanently saved in database. The changes must not be lost because of any failure.

Need for Concurrency Control

Several problems can occur when concurrent transactions execute in an uncontrolled manner.

1. Lost update problem

A lost update is a typical problem in transaction processing in SQL. It happens when two queries access and update the same data from a database.

Consider, A is processing an order for a client for 150 items. He checks from the ITEMS table that there are 300 available items.

So he starts placing the order. After few seconds B gets an order for 200 items. He also checks ITEMS table and finds that there are 300 items available. So he also starts placing the order. Meanwhile A confirms the order for 150 items and updates the ITEMS table and sets the quantity to 150. A few seconds later B confirms the order and updates the ITEMS table and sets the quantity to 100.

This problem is called lost update problem. Because both the orders of users A and B have been accepted, but there is not enough space items available. Hence the updates are lost.

Lost update

T_1	T_2
Read (X)	300
X = X - 150	
	Read (X) 300
	X = X - 200
Write (X) 150	----> This update is lost
	Write (X) 200 ---> only this update succeeds
COMMIT	
	COMMIT

Uncommitted data (or) dirty read problem (or) Temporary update

Consider A is processing an order for a client for 150 items. He checks from the ITEMS table that there are 300 available items. So he starts placing the order. A confirms the order for 150 items and updates the ITEMS table and sets the quantity to 150.

Now B receives an order for 200 items. He checks the ITEMS table to find that there is not enough items (150 available) and rejects the order. Now, due to some reason client asks A to cancel the order, so A cancels the order, rolls back and updates the ITEMS table back to 300 items. This problem is called dirty read because B saw the uncommitted update of A.

Uncommitted update (Dirty read)

T_1	T_2
Read (X)	300
X = X - 150	
Write (X)	150
	Read (X) 150 ----> Reads the value of X, which it should not have seen.
	X = X - 200
	Write (X)
ROLLBACK	
	COMMIT

Inconsistent analysis (or) Incorrect Summary problem

If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.

X = 10 Y = 15

T_1	T_2
Read (x)	10
X = X - 5	5
(Not committed)	
	Read (x) --> reads X as 10
	Y = X + Y 25
	Commit Y = 25
Commit	
Read (Y)	
X = X + Y --> 5 + 25 = 30	

Since T_1 did not commit the updated value of X as 5, T_2 reads X as 10 and hence the total changes for items X and y are wrong.

Types of transaction failures

1. A computer failure (System Crash)

A h/w, s/w, or n/w error occurs in the computer system during transaction execution. H/w crashes are usually media failures - eg. main memory failure.

2. A transaction or system error

Some operation in the transaction may cause it to fail, such as integer overflow or division by zero. It may also occur because of erroneous parameter values (or) because of logical programming error.

3. Local errors or exception conditions detected by the transaction

During transaction execution, certain conditions may occur that necessiate cancellation of the txn. Eg: Insufficient account balance in a banking database may cause a txn, such as a fund withdrawal to be cancelled

4. Concurrency control enforcement

The concurrency control method may decide to abort the transaction, because it violates serializability or several transactions are in a state of deadlock.

5. Disk failure

Some blocks may lose their data because of read or write malfunction (or) because of disk read/write head crash. This may happen during a read/write operation of the txn.

Transaction states and Additional Operations

A txn comprised of the following operations:

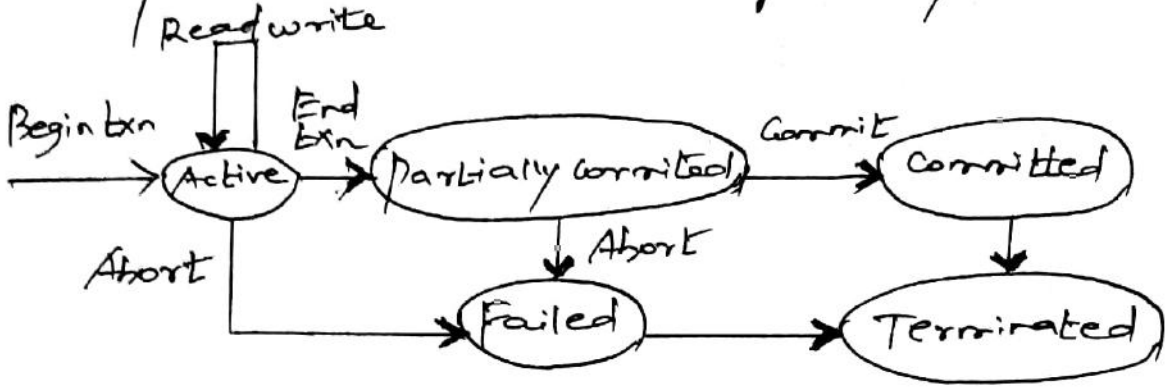
1. Begin transaction: This marks the beginning of txn execution
2. READ OR WRITE: These specify read or write operations on the db items that are executed as part of a txn.
3. End transaction: This marks the end of txn execution.
4. COMMIT_TRANSACTION: This signals a successful end of txn, that any changes (updates) executed by the txn can be safely committed to the db and will not be undone.
5. ROLLBACK (or ABORT): This signals that the txn has ended unsuccessfully, so that any changes or effects of the txn applied to the database must be undone.

Transaction states

A txn goes into an active state immediately after it starts execution, where it can issue READ and WRITE operations. When the txn ends, it moves to the partially committed state.

At this point, some recovery protocols need to ensure that a system failure will not result in an inability to record the changes of the txns permanently. Once this check is successful, the txn is said to have reached its commit point and enters committed state.

However, a txn gets into the failed state, if one of the checks fails or if the txn is aborted during its active state. The txn may then have to be rolled back to undo the effect of its WRITE operations on the db. The terminated state corresponds to the txn leaving the system.



Scheduling of transactions

schedule : A chronological execution sequence (predefined order) of a txn is called a schedule. A schedule can have many txns in it, each comprising of a number of instructions / tasks.

Types of schedules

1. Complete schedule:

A schedule that contains either an abort or commit for each txn, whose actions are listed in it is called a complete schedule. Either the txns should be fully completed (or) not fully completed.

2. Serial Schedule:

It is a schedule in which, after the completion of one txn, second txn takes place. For example, consider a schedule of txns T_1 & T_2. Assume that A = 1000, B = 2000

S_1

T_1	T_2
A = A + 100	
B = B − 100	
	A = A * 0.1
	B = B * 0.1

When Txn T_1 completes its execution, then the Txn T_2 takes place. This is called serial schedule.

3. Non-Serial schedule

It is a schedule, in which the operations from a set of Txns will be executed in an interleaved manner. It is called non-serial schedule.

S_2

T_1	T_2
A = A + 100	
	A = A * 0.1
B = B − 100	
	B = B * 0.1

4. Serializable schedule

A schedule is said to be serializable, if

i) it is a non-serial schedule

ii) it produces the same result as that of its equivalent serial schedule.

Eg

S_1

T_1	T_2
A = A + 100	
B = B − 100	
	A = A * 0.1
	B = B * 0.1

S_2

T_1	T_2
A = A + 100	
	A = A * 0.1
B = B − 100	
	B = B * 0.1

Assume
A = 1000
B = 2000

The o/p of both the schedules S_1 & S_2 for A = 110 and B = 190 are same. Hence, the non-serial

schedule S2 is serializable with its Equivalent serial schedule S1.

S2 = S1 [Serializable]

Serializability

A schedule 'S' of 'n' txns is serializable, if it is equivalent to some serial schedule of the 'n' txns. This property is called serializability.

Example of Serializable schedule or not?

Consider schedule S

T_1	T_2
W(A)	
	R(A)
	R(B)
W(B)	

Read A after updation
Read B before updation

Let us consider schedules S1, S2 & S3 and check whether they are serializable with S or not.

S1:

T_1	T_2
W(A)	
W(B)	
	R(A)
	R(B)

Read A after updation
Read B after updation ✗
$S \neq S_1$ [Not Serializable]

S2

T_1	T_2
	R(A)
	R(B)
W(A)	
W(B)	

Read A before updation ✗
Read B before updation
$S \neq S_2$ [Not Serializable]

S3

T_1	T_2
W(A)	
	R(B)
	R(A)
W(B)	

⇒ It is a non-serial schedule
Hence $S \neq S_3$ [Not serializable]

Conflict Equivalence & Conflict Serializability

A pair of operations are said to be in conflict, if they satisfy the following conditions:

1) Both operations belong to different trans.
2) They access the same data
3) At least one of the operations is write operation.

Eg:

1. T_1 : R(A) T_2 : R(A)

T_1	T_2
R(A)	
	R(A)

All the above conditions are not satisfied only the first two conditions are satisfied

So, Non-conflict pair.

2. T_1 : R(A) T_2 : W(A)

T_1	T_2
R(A)	
	W(A)

All the 3 conditions are satisfied. Hence, conflict pair.

3. $R_1(A)$; $W_2(B)$ } → Not conflict pairs. Because they
$W_1(A)$; $R_2(B)$ } access different data.

Swapping of operations, is possible if the pair of operations are Non-conflict pairs.

Eg

S1

T_1	T_2
R(A)	
	R(B)

Swapped ⟹

S2

T_1	T_2
	R(B)
R(A)	

Non-conflict pairs

S1 = S2

T_1	T_2
R(A)	
	W(A)

Swapped ⟹

T_1	T_2
	W(A)
R(A)	

Conflict pairs

S1 ≠ S2

When are 2 schedules Equivalent?

There are 3 types of schedule equivalences:

1. Result equivalence
2. Conflict equivalence
3. View equivalence

Based on the types of equivalence, different types of serializability are defined. They are

1. Result serializability
2. Conflict serializability
3. View serializability

Result Equivalence & Result serializability

In result equivalence, the end result of schedules heavily depend on i/p of schedules. The final values are calculated from both schedules (given and serial) and check whether they are equal. Result serializability is not generally used because of lengthy process.

S_1	S_2
$x = 100$	$x = 100$
$R(x)$	$R(x)$
$x = x + 10$	$x = x * 1.1$
$W(x)$	$W(x)$
$= 110$	$= 110$

For $x = 100$, $S1 \overset{R}{=} S2$

If $x = 200$, they are not equal

$S1 \overset{R}{\neq} S2$

Conflict Equivalence

Schedules are conflict equivalence, if they can be transformed one into another, by interchanging a sequence of non-conflicting adjacent operations.

Eg. check whether the schedules S1 & S2 are conflict equivalent or not.

S1: $\frac{1}{R_1(A)}$; $\frac{2}{R_2(B)}$; $\frac{3}{w_1(A)}$

S2: $\frac{4}{R_1(A)}$; $\frac{5}{w_1(A)}$; $\frac{6}{R_2(B)}$

Solution:

S1

T_1	T_2
1 $R_1(A)$	
	$R_2(B)$ 2
3 $w_1(A)$	

S2

T_1	T_2
1 $R_1(A)$	
2 $w_1(A)$	
	$R_2(B)$ 3

In S2, [2] & [3] are interchangeable, ∵ they are non-conflicting.

Thus $S1 \overset{c}{=} S2$

S1 is in conflict equivalence with S2.

Eg: Check whether the schedules S1 & S2 are conflict equivalent or not

S1: $\frac{1}{R_2(A)}$; $\frac{2}{w_2(A)}$; $\frac{3}{R_3(C)}$; $\frac{4}{w_2(B)}$; $\frac{5}{w_3(A)}$; $\frac{6}{w_3(C)}$;
$\frac{7}{R_1(A)}$; $\frac{8}{R_1(B)}$; $\frac{9}{w_1(B)}$; $\frac{10}{w_1(C)}$

S2: $\frac{1}{R_3(C)}$; $\frac{2}{R_2(A)}$; $\frac{3}{w_2(A)}$; $\frac{4}{w_2(B)}$; $\frac{5}{w_3(A)}$; $\frac{6}{R_1(A)}$;
$\frac{7}{R_1(B)}$; $\frac{8}{w_1(B)}$; $\frac{9}{w_1(C)}$; $\frac{10}{w_3(C)}$

S_1

T_1	T_2	T_3
	$R_2(A)$	
	$W_2(A)$	
		$R_3(C)$
	$W_2(B)$	
		$W_3(A)$
		$W_3(C)$
$R_1(A)$		
$R_1(B)$		
$W_1(B)$		
$W_1(C)$		

S_2

T_1	T_2	T_3
		$R_3(C)$ 1
2	$R_2(A)$	
3	$W_2(A)$	
4	$W_2(B)$	
		$W_3(A)$ 5
6 $R_1(A)$		
7 $R_1(B)$		
8 $W_1(B)$		
9 $W_1(C)$		
		$W_3(C)$ 10

In S_2,

1 & 2 can be interchanged, then after that 2 & 3 can be interchanged

6,7,8,9 & 10 can be interchanged with each other.

After completing, the swapping of non-conflict pairs of S_2, finally S_1 & S_2 are conflict equivalent.

$$S_1 \overset{c}{=} S_2$$

Conflict Serializable schedule

A schedule is conflict serializable, if it is conflict equivalent to any of serial schedule.

Testing for Conflict serializability

Method 1:

1. First write the given schedule in a linear way.
2. Find the conflict pairs (RW, WR, WW) on same variable by different transactions.
3. Whenever conflict pairs occur, write the dependency relation like $T_i \rightarrow T_j$, if conflict pair

is from T_i to T_j ($R_1(A), W_1(A), R_2(A)$) = $T_1 \rightarrow T_2$

4. Check to see if there is a cycle formed,
   * if Yes, then not Conflict Serializable
   * if no, then conflict serializable

Example:

Check whether the schedule is conflict serializable or not?

S: R1(A); W1(A); R2(A); R1(B); W1(B); R2(B)

Solution

T_1	T_2
1 R1(A)	
2 W1(A)	
	R2(A) 3
4 R1(B)	
5 W1(B)	
	R2(B) 6

Conflict pairs:

(2,3) W1(A) : R2(A)
(5,6) W1(B) : R2(B)

Dependency relation

W1(A) : R2(A) => $T_1 \rightarrow T_2$
W1(B) : R2(B) => $T_1 \rightarrow T_2$

No cycle is formed, ∴ the given schedule is conflict serializable schedule.

Method 2:

To test the conflict serializability, we can draw a Graph G=(V,E) where

V = Vertices = no. of txns
E = Edges = for conflicting pair

Steps:

1. Create node for each txn.
2. Find the conflict pair.
3. Draw edge for the conflict pair.
   (Eg) W2(B): R1(A) is a conflict pair, draw edge from T2 to T1.
4. Testing conditions for Conflict serializability
   * If precedence graph is cyclic, the given schedule is not conflict serializable.
   * If precedence graph is acyclic, the given schedule is conflict serializable.

Example

check whether the schedule is conflict serializable or not?

S: R1(A); R2(A); R1(B); R2(B); R3(B); W1(A); W2(B)
   1 2 3 4 5 6 7

Step 1:
Create a node for each txn.

Step 2: Find the conflict pairs

T_1	T_2	T_3
1 R1(A)		
	R2(A) 2	
3 R1(B)		
	R2(B) 4	
		R3(B) 5
6 W1(A)		
	W2(B) 7	

Conflict pairs

(5, 7) R3(B) : W2(B)
(2, 6) R2(A) : W1(A)
(3, 7) R1(B) : W2(B)

Step 5:

Draw an edge for each conflict pair.

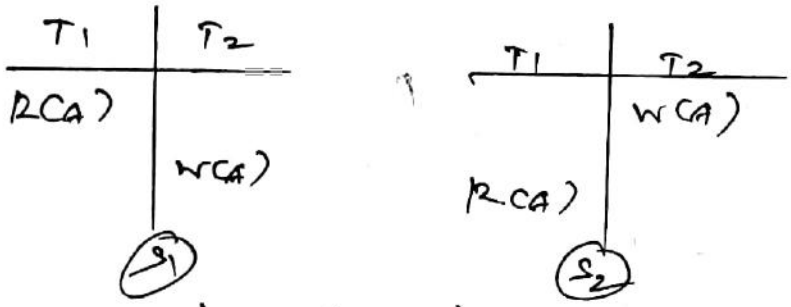
The precedence graph is cyclic, hence the given schedule is not conflict serializable.

View Equivalent schedule

Consider two schedules S1 & S2, they are said to be view equivalent, if following conditions are true.

Condition 1:

Initial read must be same



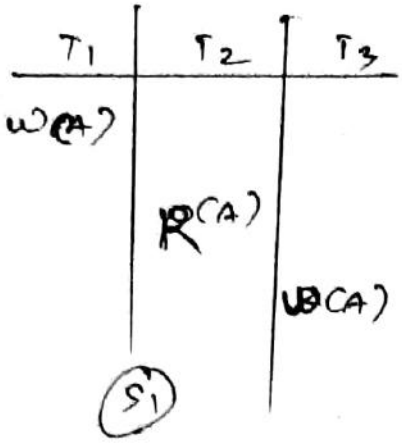
S1: T1 reads A from database

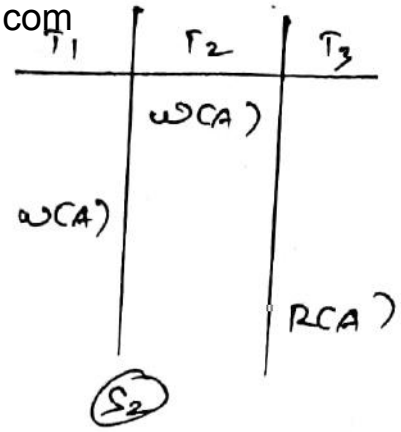
S2: T1 reads A from T2

$\therefore S1 \neq S2$

Condition 2:

If there are two txns T_i and T_j, the schedules S1 and S2 are view equivalent, if in schedule S1, T_i reads A and then updated by T_j (i.e) (RW) sequence, then in schedule S2, T_i must read A, which should be updated by T_j. (*) Read-Write (RW) sequence must be same between S1 & S2.





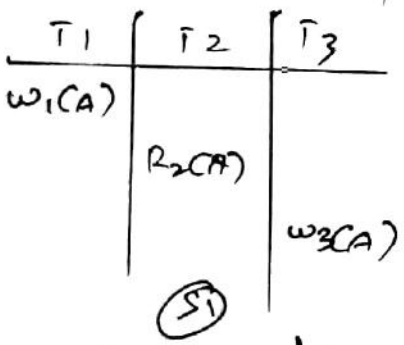
S1: Read_write Sequence is $R_2(A)\ W_3(A)$

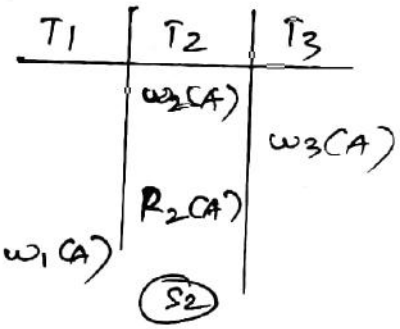
S2: No read_write sequence.

$\therefore$ S1 ≠ S2

Condition 3:

Final write operations should be same in S1 & S2.





S1: Final update of A is done in T3

S2: Final update of A is done in T1

$\therefore$ S1 ≠ S2

View Serializable Schedule

A schedule is view serializable, if it is view equivalent to any serial schedule.

Eg:

Check whether the schedule is view serializable or not?

S: R2(B); R1(A); R3(A); W1(B); W2(B); W3(B)

Solution: With 3 transactions, the number of schedules possible are

$= 3! = 6$

<T1 T2 T3>
<T1 T3 T2>
<T2 T1 T3>
<T2 T3 T1>
<T3 T1 T2>
<T3 T2 T1>

T_1	T_2	T_3
	$R_2(B)$ 1	
2 $R_1(A)$		
		$R_3(A)$ 3
4 $W_1(B)$		
	$W_2(B)$ 5	
		$W_3(B)$ 6

Step 1: Final update on data item A (or) B

In the given schedule, the final update is done on data B in T_3. Hence out of the 6 serial schedules <T2 T1 T3> & <T1 T2 T3> are the schedules which made final update by T_3.

Hence <T1 T2 T3>
<T2 T1 T3>

Step 2: Initial read on data item A (or) B

out of the above given schedule, the initial read on data B is done in T2 $[R_2(B)]$. So, from the available serial schedules, initial read should be done by <T2 T1 T3>.

Step 3: Read write sequence on data item A or B

In the given schedule, the read write sequence is $R_2(B)$: $W_1(B)$ (ie) $T_2 \rightarrow T_1$

Hence in the available only equivalent serial schedule also has the flow as $T_2 \rightarrow T_1 \rightarrow T_3$.

Hence the given schedule is view serializable.

Concurrency Control

When multiple txns are trying to access the same shared resource, there could arise many problems, if the access control is not done properly. There are some important mechanisms to which access control can be maintained. The Concurrency control is implemented theoretically using serializability. Practically, it can be implemented by 2 mechanisms namely,

1) Lock based protocols
2) Time stamping protocols

Lock based protocols → user is responsible to write consistent concurrent txn to implement Concurrency control.

Time stamp protocols → System itself tries to detect possible inconsistency ~~using~~ during concurrent execution and either the inconsistency is recovered (or) avoided.

Classification of Concurrency Control protocol

Lock based protocols

1. Binary locks
2. Shared / Exclusive (or) Read / write locks
3. 2 phase locking protocol

Time stamp protocol

1. Time stamp ordering protocol

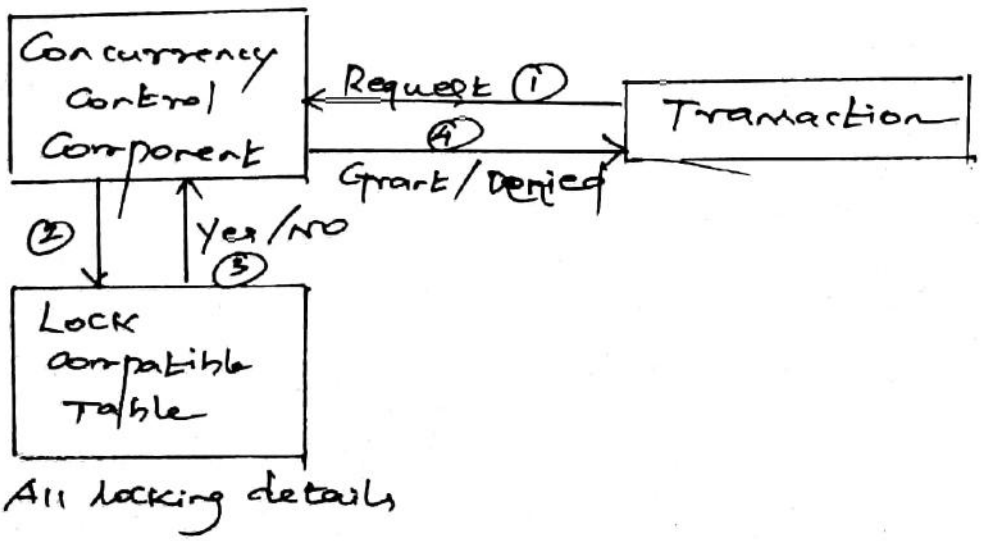
Locking

1. A lock variable is associated with each data item which is used to identify the status of the data item. (whether the data is in use or not).
2. When a txn, intends to access the data item, it must first examine its associated lock.
3. If no other txn holds the lock, the scheduler lock the data item for T.
4. If another txn T1, wants to access the same data item, then the txn T1, has to wait until the previous txn releases the lock.
5. Thus at a time, only one txn, can access the data item.

Eg:

T1

LCA) → Locks the data item A
RCA) → Reads the data item A
WCA) → writes the data item A
UCA) → unlocks the data item A
LCB) → Locks the data item B



All locking details

Binary locks

A binary lock can have 2 states or values

* Locked (or 1) and
* Unlocked (or 0)

The current state (or value) of the lock associated with data item X as Lock(X).

Operations used with Binary locking

1. lock_item: A txn requests access to an item by first issuing a lock_item(x) operation.
   * If Lock(X)=1 or L(X): the txn is forced to wait
   * If Lock(x)=0 or U(x): it is set to 1 (the txn locks the item) to access it.
2. unlock_item: After the completion of access with the data item, the txn issues an operation unlock(x), which sets the operation Lock(x)=0. (ie) unlocks the data item X, so that it can be accessed by other txns.

Rules

1. A txn T must issue the lock(x) operation before any read(x) or write(x) operations in T.
2. A txn T must issue the unlock(x) operation after all read(x) and write(x) operations in T.
3. If a txn T already holds the lock on item x, then T will not issue a lock(x) operation.
4. If a txn T does not hold a lock on item x, then T will not issue an unlock(x) operation.

Example

T_1	T_2
lock(A)	
write(A)	
unlock(A)	
	lock(A)
	read(A)
	unlock(A)
	lock(B)
	read(B)
	unlock(B)
lock(B)	
read(B)	
write(B)	
unlock(B)	

Implementation of Binary locks

It is implemented using 3 fields plus a queue for txns. They are

1. Data_item_name
2. Lock
3. Locking_txn

Shared/Exclusive (or) Read/write lock

The binary lock is too restrictive for data items because at most one txn can hold on a given item, whether the txn is reading or writing. To improve it we have Shared/Exclusive lock, in which more than one txn can access the same data item for reading purposes. ie. the read operations on same data item by different txns are not conflicting.

Two kinds of lock are supported:

* Exclusive (or) Write locks
* Shared (or) Read locks

Shared locks

If a txn T_i has locked the data item A in shared mode, then a request from another txn T_j on A for:

* W(A) → denied, T_j has to wait until T_i unlocks A.
* R(A) → Allowed

Exclusive locks

If a txn T_i has locked the data item A in exclusive mode, then a request from another txn T_j on A for:

* W(A) → denied
* R(A) → denied

Compatible table for S/E locks

Txn T_j requests for data item A on:	Txn T_i holds A on: S	Txn T_i holds A on: X
R	YES	NO
W	NO	NO

S → Shared mode
X → Exclusive mode
R → Read
W → Write

Lock compatible table holds the status of a data item, whether it is locked or not.

Implementation of S/E locks

It is implemented using 4 fields:

1. Data_item_name
2. Lock
3. No. of records &
4. Locking_txn

Data items that are not in the lock table are considered to be unlocked. The system maintains only those records for data items that currently locked.

Value of Lock(A): Read locked (or) Write locked

* If Lock(A) = write_locked: The value of locking txn is a single txn that holds the exclusive lock on A.
* If Lock(A) = read_locked: The value of locking txn is a list of one (or) more txns that hold the shared lock on A.

Rules

1. A txn T must issue the operation S(A) or read_lock(A) or X(A) or write_lock(A) before any read(A) operation is performed in T.
2. A txn T must issue the operation X(A) or write_lock(A), before any write(A) operation is performed in T.
3. After completion of all read(A) and write(A) operations in T, a txn T must issue an unlock(A) operation.
4. If a txn already holds a shared lock (or) exclusive lock on item A, then txn T will not issue an unlock(A) operation.
5. A txn that already holds a lock on item A, is allowed to convert the lock from one locked state to another under certain conditions.
   * upgrading the lock by issuing a write_lock(A) operation (or) conversion of read_lock() to write lock().
   * Downgrading the lock by issuing a read_lock(A) (or) conversion of write_lock() to read_lock().

2 phase locking

Binary locks or S/E locks does not guarantee serializability. To ensure serializability 2 phase locking (2PL) is used.

In this scheme, each txn makes lock & unlock request in 2 phases:

1. Growing phase (Locking phase): In this phase, new locks on the desired data item can be acquired but none can be released.
2. Shrinking phase (unlocking phase): In this phase, existing locks can be released, but no new locks can be acquired.

T

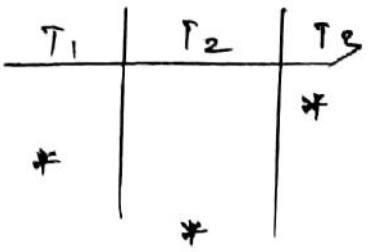
LCA)
LCB) } → Growing (or) Locking phase
LCC)

⋮ → Lock point [Last lock position (or) first unlock position]

UCB)
UCA) } → Shrinking (or) unlocking phase
UCC)

A 2PL always result serializable schedule, but it does not permit all possible serializable schedules. (ie) some serializable schedules will be prohibited by the protocol.

A txn T does not allow to request any lock if T has already performed some unlock operation and every equivalent serial schedule is based on the order of the Lock point.



order of lock point = $T_3 \rightarrow T_1 \rightarrow T_2$

Points about 2pl — Two phase locking

1. Every non serializable schedule failed to execute 2pl.
2. 2pl always results in serializable schedule.
3. Equivalent serial schedule is based on the order of lock point.
4. If a schedule is allowed to execute using 2pl, then the schedule is conflict serializable, but not vice versa.
5. If a schedule is non conflict serializable, then the schedule is not allowed to execute using 2pl.

Deadlock in transaction

Deadlock occurs when each txn T_1 in a set of 2 or more txns is waiting for some item that is locked by some other txn T_2 in the set. Hence each txn in the set is on a waiting queue, waiting for one of the other txns in the set to release the lock on an item.

Eg

T_1	T_2
LCA)	
RCA)	
	LCB)
	RCB)
	LCA) → Denied
Denied → LCB)	

where the deadlock?

The concurrency control technique called locking may lead to deadlock.

Locking:

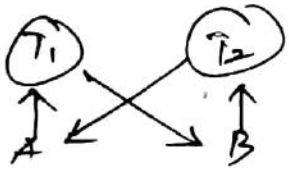
2pl
2pl + Binary locking } may lead to deadlock
2pl + S/E locking

Example:

T_1	T_2
S(A)	
R(A)	
	X(B)
	W(B)
Denied → S(B)	
	X(A) → Denied

(2pl + S/E locking)

T_1 is waiting for T_2 to get B
T_2 is waiting for T_1 to get A] → Deadlock



What is deadlock detection & prevention?

Deadlock detection is to check the system, whether deadlock state actually exists or not.

Deadlock prevention is to prevent the system from deadlock.

When to use deadlock detection and when to use deadlock prevention?

* If the txn is short and locks only a few items, then deadlock detection scheme should be used.
* If the txn is long and locks many data items, then deadlock prevention scheme should be used.

Deadlock detection schemes

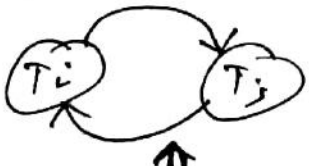
Wait-for-Graph (WFG)

To detect the state of deadlock, directed graph is to be constructed which is called wait-for-graph (WFG). The nodes of WFG are labelled with active txn names and an edge exists from $T_i \rightarrow T_j$ if txn T_i is waiting for txn T_j to release some lock.

Active txns : T_i & T_j

↳ Edge exist if T_i is waiting for txn T_j to release some lock.

For a deadlock to occur, WFG must have a cycle and a scheduler can detect deadlocks by checking for cycles in WFG.



cycle exists

∴ The above schedule of T_i & T_j is in deadlock state

* After the detecting of deadlock, it can be prevented by aborting a txn.
* the scheduler should select the txn among the txns involved in the cycle of wfg, where abortion involves only minimum cost. i.e it tries to select the younger txn.
* The chosen txn for abort is called victim txn.

Timeout method

Timeout method is more practical scheme for deadlock. In timeout method, if a txn waits for a period longer than a system defined timeout period, the system assumes that the txn may be in deadlock state and aborts it regardless of whether a deadlock actualy exist or not.

Deadlock Prevention Protocols

They are used to prevent from deadlock. There are various deadlock prevention protocols. They are

1. Conservative two phase Locking (Conservative 2pL)
2. Ordering all the items protocol
3. No waiting algorithm
4. Cautious waiting algorithm
5. Deadlock prevention using Concept of Timestamp ordering [TS CT)]

Conservative Two phase locking (Conservative 2PL)

* In conservative 2PL, each txn locks all the items in advance, therefore no deadlock occurs. If any of the items cannot be obtained, then none of the items are locked.
* The conservative 2PL is not a practical method, as it provides less concurrency and it can lead to starvation also.

Ordering all the items protocol

* This protocol also limits concurrency, because it involves ordering all the items in the db, and makes sure that the txn that needs several data items, will look them according to that order.
* It is also not a practical method, as the programmer (or the system) is aware of the chosen order of items.

No waiting algorithm

* In this scheme, if a txn is unable to obtain a lock, it is immediately aborted and then restarted after a certain time delay without seeing whether a deadlock will actually occur or not.
* This algorithm can cause txns to abort & restart needlessly.

Cautious waiting algorithm

To reduce the number of needless aborts and restarts, the cautious algorithm is used. Suppose there are two txns T_i & T_j. T_i is waiting for a data item X which is locked by T_j.

* If T_j : Blocked (waiting for some other locked data item) — Abort T_i
* If T_j : Not blocked (not waiting for some other locked data item) — T_i is blocked and allowed to wait.

The cautious waiting is deadlock free, because no txn will never wait for another blocked txn.

Deadlock prevention using the concept of Timestamp ordering [TS(T)]

Txn timestamp (TS(T)) is a unique identifier assigned to each txn based on the order in which txns are started.

Transaction timestamp, TS(T)

It is a unique identifier but not any kind of time. TS(T) can be understood as a priority identifier in which the lesser number identifies older txn and greater number identifies younger txn.

(i.e) if T_1 starts before T_2

then $TS(T_1) < TS(T_2)$

(less priority) (High priority)
(older txn) (younger txn)

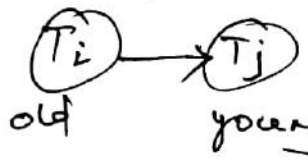
There are two methods for preventing deadlock using the concept of timestamp ordering:

1. Wait die
2. Wound wait

Suppose there are two txns T_i & T_j where T_i is older than T_j. ie $TS(T_i) < TS(T_j)$. The following rules are followed by these schemes:

Wait die:

* If txn T_i is waiting for a data item X, which is locked by T_j, then T_i is allowed to wait

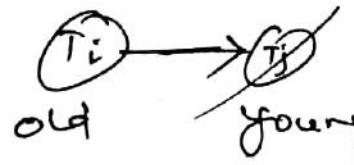


* If txn T_j is waiting for a data item X, which is locked by T_i, then abort T_j (T_j dies) and restart later with the same timestamp.

In a wait-die method, an older txn is allowed to wait for a younger txn, whereas a younger txn requesting an item held by an older txn is aborted and restarted.

Wound wait:

* If txn T_i is waiting for a data item X, which is locked by T_j, then abort T_j and restart it later with the same timestamp. (T_i wounds T_j)



by aborting T_j, the resources held by it become available, which can be used by T_i.

* If a txn T_j is waiting for a data item X, which is locked by T_i, then T_j is allowed to wait.

Time Stamp ordering protocol

Time stamp → It is an unique identifier created by DBMS to identify a txn in ascending order.

T_1	T_2	T_3	T_4
10	20	30	40
↓			↓
oldest txn			Youngest txn

Read Time Stamp (RTS)

Highest txn time stamp value (or) youngest txn that has performed read operation successfully.

10	20	30	40
T_1	T_2	T_3	T_4
$R_1(A)$			
$W_1(A)$			
		$R_3(A)$	
	$R_2(A)$		
			$W_4(A)$

$RTS(A) = \phi \; 10 \; 30 \; 20$
$R_1 \; R_3 \; R_2$

$\boxed{RTS(A) = 30}$

Write Time Stamp (WTS)

Highest txn time stamp value (or) youngest txn that has performed write operation successfully.

$WTS(A) = \phi \; 10 \; 40$
$W_1 \; W_4$

$\therefore \boxed{WTS(A) = 40}$

Basic rules

1. According to TS ordering, the concurrent execution of txns should be equal to the serial schedule based on the order of TS value.

Eg1

10	20	30
T_1	T_2	T_3

order of TS value = T_1 T_2 T_3
Hence, equivalent serial schedule order = T_1 T_2 T_3

Eg2

10	30	20
T_1	T_2	T_3

Equivalent serial schedule order = T_1 T_3 T_2

2. Conflict operations are allowed in TS ordering schedule, but they must have the order of TS. If they don't satisfy the order, then roll back that operation & restart it again.

Eg

10	30	20
T_1	T_2	T_3
1 $R_1(A)$		
		$W_3(A)$ 2
3 $W_2(B)$		
		$W_3(B)$ 4

TS order = T_1 T_3 T_2

Conflict order:

$R_1(A)$ $W_3(A)$ $\Rightarrow$ $T_1 \rightarrow T_3$

$W_2(B)$ $W_3(B)$ $\rightarrow$ $T_2 \rightarrow T_3$, which mismatches with TS order.

Hence, rollback $W_3(B)$ and restart as new txn T_4.

10 T_1	30 T_2	20 T_3	40 T_4
1 $R_1(A)$			
		2 $W_3(A)$	
	3 $W_2(B)$		
			4 $W_4(B)$

TS order = $T_1\ T_3\ T_2\ T_4$

Conflict order:

$R_1(A)\ W_3(A) \Rightarrow T_1\ T_3$

$W_2(B)\ W_4(B) \Rightarrow T_2\ T_4$

Now ## Conflict order & TS order matches.

Basic timestamp ordering protocol

1) $r_i(x)$: if $TS(T_i) < WTS(x)$, then reject it

otherwise $TS(T_i) \geq WTS(x)$, then schedule $r_i(x)$ and set $RTS(x) = \max(RTS(x), TS(T_i))$

Eg:

10 T_1	20 T_2
	$W_2(x)$
$R_1(x)$	

TS order = $T_1\ T_2$

Eq. Serial schedule = $T_1\ T_2$

$TS(T_1) = 10$ $\quad WTS(x) = \phi\ 20$

$TS(T_2) = 20$ $\quad RTS(x) = \phi\ 10$

Condition: $TS(T_i) < WTS(x)$, then reject & restart

$10 < 20$, hence rollback & restart

So, $R_1(x)$ of T_1 is rolled back and restarted with new Txn T_3.

10 T_1	20 T_2	30 T_3
	$W_2(x)$	
$R_1(x)$		$R_1(x)$

Rollback

Now TS order = $T_2\ T_3$

$TS(T_2) = 20$ $\quad WTS(x) = \phi\ 20$

$TS(T_3) = 30$ $\quad RTS(x) = \phi\ 30$

Condition: $TS(T_3) \geq WTS(x)$, then set $RTS(x) = \max(RTS(x), TS(T_i))$

$30 \geq 20$

Hence $\boxed{RTS(x) = 30}$

2. $W_i(x)$: If $TS(T_i) < RTS(x)$, then reject it
If $TS(T_i) < WTS(x)$, then reject it
otherwise schedule $W_i(x)$ and set
$RTS(x) = max(WTS(x), TS(T_i))$

Transaction recovery Using Savepoint

Transaction Control Language (TCL) Commands are used to manage transactions in the database. TCL Commands are
Commit, Rollback and savepoint.

Commit Command

Commit Command is used to permanently save any transaction into the database.

When DML commands like Insert, Update or Delete are used, the changes made by these commands are not permanent until the current session is closed. The changes made by these Commands can be rolled back (retrieved back)

To make these changes permanent COMMIT Command is used.

```
COMMIT;
```

Rollback command

This command restores the database to last commited state. It is also used with Savepoint Command to jump to a savepoint

in an ongoing transaction.

If update command is used to make some changes into the database and realised that those changes were not required, then Rollback command can be used to roll back to the changes. [If Commit command is not used.]

```
ROLLBACK TO savepoint_name;
```

Savepoint Command

Savepoint command is used to temporarily save a transaction, so that you can rollback to that point whenever required.

```
SAVEPOINT savepoint_name;
```

Class

id	name
1	Abhi
2	Adam
4	Alex

```
Insert into class values (5, 'Rahul');
Commit;
update class set name = 'Abhijit' where id = '5';
Savepoint A;
Insert into class values (6, 'Chris');
Savepoint B;
```

Insert into class values (7, 'Bravo');

Savepoint c;

Select * from class;

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	chris
7	Bravo

Rollback to B;

Select * from class;

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	chris

Rollback to A;

Select * from class;

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit

Transaction isolation levels

In order to maintain consistency in a database, it follows ACID properties. Among these four properties, Isolation determines how transaction integrity is visible to other users and systems. It means that a transaction should take place in a system in such a way that it is the only transaction that is accessing the resources in a database system. A transaction isolation level are defined by the following phenomena:

* Dirty read — A dirty read is the situation when a transaction reads a data that has not yet been committed. For e.g., lets say txn1 updates a row and leaves it uncommitted, meanwhile txn2 reads the updated row. If txn1 rolls back the change, txn2 would have read the data that is considered never to have existed.

* Non repeatable read — It occurs when a transaction reads same row twice, and get a different value each time. For e.g. T_1 reads a data. Due to concurrency, T_2 updates the same data and commit, Now if T_1 re-reads the same data, it will retrieve a different value.

* Phantom Read — It occurs when two same queries are executed, but the rows retrieved by the two are different. For e.g. suppose T_1 retrieves a set of rows that satisfies some search criteria. Now T_2 generates some new rows that matches the search criteria for T_1. If T_1 re-executes the statement that reads the rows, it gets a different set of rows this time.

Based on the above phenomena, the SQL defines four isolation levels:

1. Read Uncommitted — It is the lowest isolation level. In this level, one transaction may read not yet committed changes made by other transaction, thereby allowing dirty reads. In this level, transactions are isolated from each other.

2. Read Committed — This isolation level guarantees that any data read is committed at the moment it is read. Thus it does not allow dirty read. The transaction hold a read or write lock on the current row, and thus prevent other rows from reading, updating or deleting it.

3. Repeatable read — This is the most restrictive isolation level. The transaction holds a read locks on all rows it references and write locks on all rows it inserts, updates or deletes. Since other transaction cannot read, update or delete these rows, consequently it avoids non-repeatable reads.

4. Serializable — This is the highest isolation level. A serializable execution is guaranteed to execute set of operations, in which concurrently executing transactions are serial [one by one].

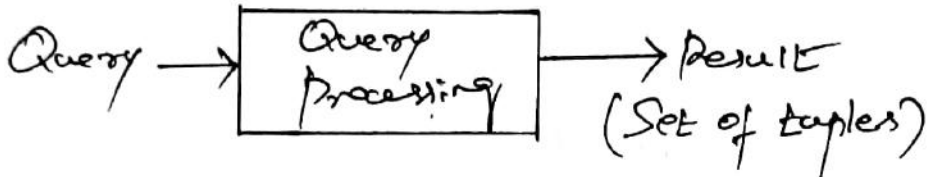
Relationship between isolation levels and read phenomena

Isolation level	Dirty reads	Non-repeatable reads	phantom reads
Read Uncommitted	May occur	May occur	May occur
Read Committed	Don't occur	May occur	May occur
Repeatable read	Don't occur	Don't occur	May occur
Serializable	Don't occur	Don't occur	Don't occur

RAID – File Organization – Organization of Records in Files – Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing – Query Processing Overview – Algorithms for SELECT and JOIN operations – Query optimization using Heuristics and Cost Estimation.

Query processing and optimization

It is the list of activities which are performed to obtain the required tuples that satisfy a given query.



Steps involved in query processing procedure:
3 basic units are used to process the query.

1. parser and translator
2. Query optimizer
3. Query evaluation engine

Parser and translator

1. It checks if the query is written in correct syntax.

Eg:

Emp

name	salary	address
Ravi	55000	Chennai
Ram	40000	Madurai
Raj	60000	Mumbai

Select name, salary from emp having name = 'abc';

The above query is syntactically wrong, since having keyword is used instead of where.

Hence the query is invalid and the parser & translator will reject this query.

2. It checks for correct schema elements. [relations & attributes]

E.g Select name, email from emp where salary > 50000;

The above query uses email as attribute name which is not the schema element of Emp relation. Hence the query is rejected.

Select name, salary from emp where salary > 50000;

The above query is syntactically correct and also the schema elements are valid. Hence the parser and translator will process the query.

3. It converts the query into its equivalent relational algebraic expression.

The relational algebraic expression for the above query is

$$\Pi_{name, salary}(\sigma_{salary > 50000}(emp)) \quad \text{— RA Ex1}$$

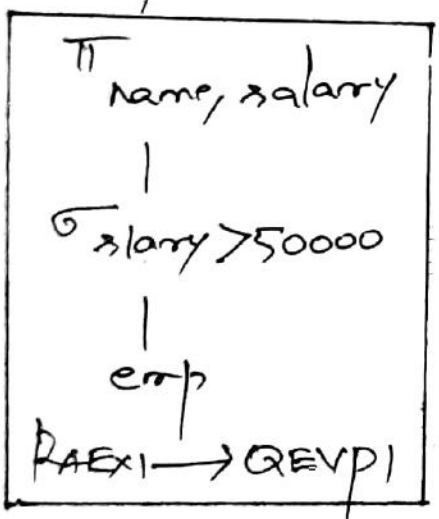
Query optimizer

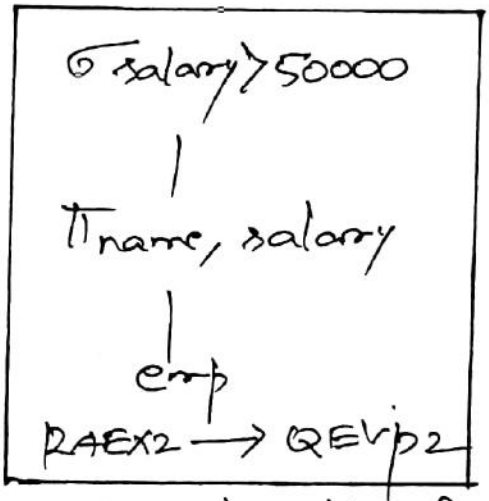
1. Query optimizer will find the all possible relational algebraic expression for the given query.

The other R.A. expression for the above query is

$\sigma_{salary>50000}(\Pi_{name,salary}(emp))$ — RAEX2

2. Then it converts the available R.A expression into its tree structure called query tree (or) Query Evaluation plan (or) Logical query plan.





In the above QEVP, the algorithm for each operation is specified. Take for instance—

QEVP1:

$\Pi_{name,salary}$ [Use Algorithm C]

$\sigma_{salary>50000}$ [Use Index A, Algorithm B]

emp

For all the QEVP available, the algorithms are specified for each operation.

3. The optimizer will construct the costs for all the available QEvp with the catalog information.

4. Finally based on the cost of each plan evaluated, the best QEvp is chosen and given to the evaluation engine.

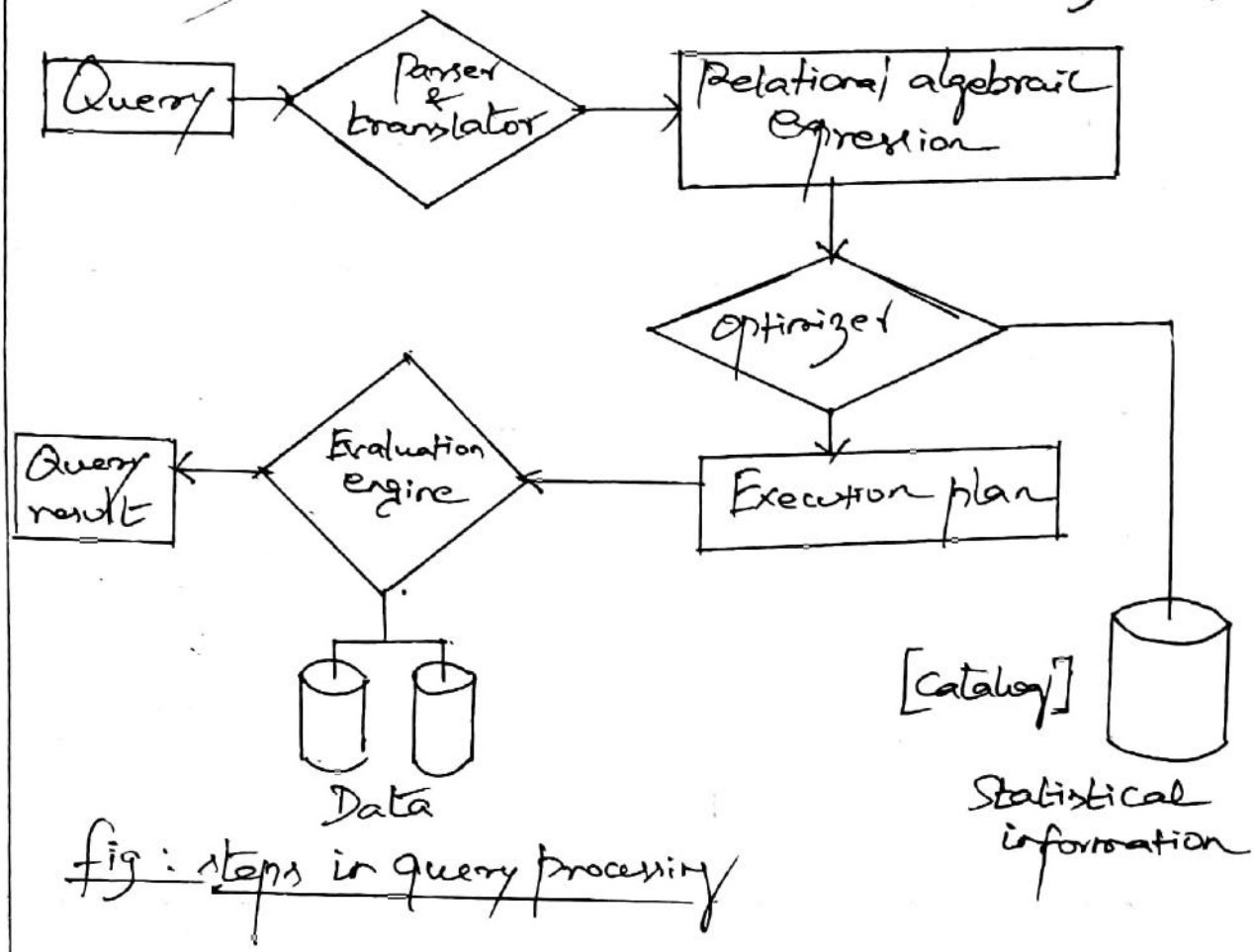


fig: steps in query processing

Evaluation Engine

The evaluation engine takes a query evaluation plan, executes that plan and returns the answers to the query.

Catalog information for cost estimation

$N_R \rightarrow$ No. of tuples in relation R

$B_R \rightarrow$ No. of blocks that contains tuples of the relation R.

$S_R \rightarrow$ Size of a tuple of R

$F_R \rightarrow$ Blocking factor, number of tuples from R, which fit into one block.

$F_R = \left[N_R / B_R \right]$

$V(A, R) \rightarrow$ No. of distinct values for attribute A in R.

Measures of Query Cost

Cost of QEvp can be measured in terms of

1. Disk access [No. of blocks transfer from disk to RAM]
2. Cpu time to execute query.
3. Cost of Communication.
4. Cost of algorithm depends on database buffer size. More memory for db buffer reduces disk accesses. Thus db buffer size is a parameter for estimating the cost.

Cost estimate of algorithm 'S' is repeted by cost(S).

Algorithms for SELECT Operation

S1 Linear search (brute force)

Retrieve every record in the file, and test whether its attribute values satisfies the selection condition

S2 Binary search

If the selection condition involves an Equality comparison on a key attribute on which the file

is ordered, binary search can be used.

S3 Single record search

If the selection condition involves an equality comparison on a key attribute with a primary index (or a hash key), use the primary index (or the hash key) to retrieve the record.

S4 Multiple search (primary index)

If the comparison condition is >, ≥, < or ≤ on a key field with a primary index, use the index to find the record satisfying the corresponding equality condition, then retrieve all subsequent records in the (ordered) file.

S5 Multiple search (clustering index)

If the selection condition involves an equality condition on a non-key attribute with a clustering index, use the clustering index to retrieve all the records satisfying the selection condition.

S6 Range query (B+-tree)

To be used to retrieve records on conditions involving >, >=, < or <=.

Can also be used for an equality comparison, for single record search, if the indexing field has unique values (i.e a key) or to retrieve multiple records if the indexing field is not a key.

S7 Conjunctive selection

If an attribute involved in any single simple condition in the conjunctive condition has an access path that permits the use of one of the methods S2 to S6, use that condition to retrieve the records and then check whether each record satisfies the remaining simple conditions in the conjunctive condition.

S8 Conjunctive selection using a composite index

If two or more attributes are involved in equality conditions in the conjunctive condition and a composite index (or hash structure) exists on the combined field, we can use the index directly.

S9 Conjunctive selection by intersection of record pointers

This method is possible if secondary indexes are available on all (or some of) the fields involved in equality comparison conditions in the conjunctive condition and if the indexes include record pointers (rather than block pointers)

Algorithms for JOIN operations

J1 Nested-loop join (brute force)

For each record t in R (outer loop), retrieve every record s from S (inner loop) and test

whether the two records satisfy the join condition $t[A] = s[B]$

J2 Single-loop join (using an access structure to retrieve the matching records):

If an index (or hash key) exists for one of the two join attributes, say B of S — retrieve each record t in R, one at a time and then use the access structure to retrieve directly all matching records from S that satisfy $s[B] = t[A]$

J3 Sort-merge join

J4 Hash-join

Cost based optimizer

1. A cost-based query optimizer first generates all possible QEVP's.
2. Next, the cost of each plan is estimated.
3. Finally, based on the estimation, the plan with the lowest estimated cost is chosen.
4. The quality depends on complexity and accuracy of cost-function.

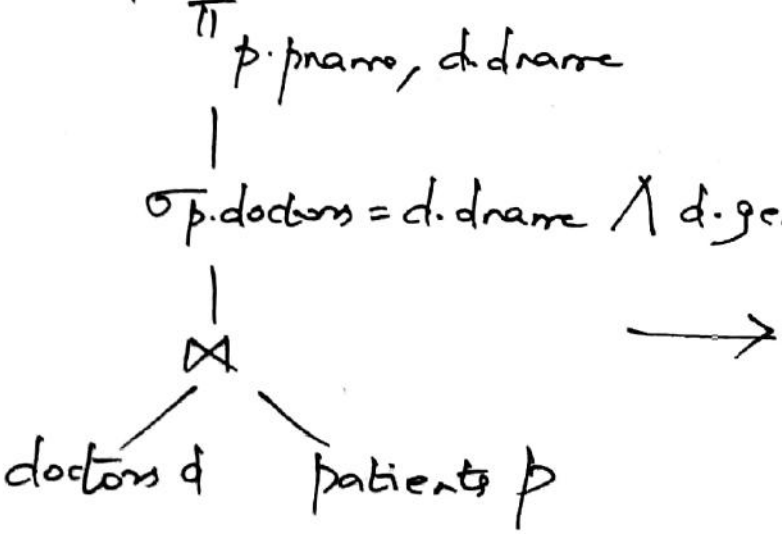
Eg:

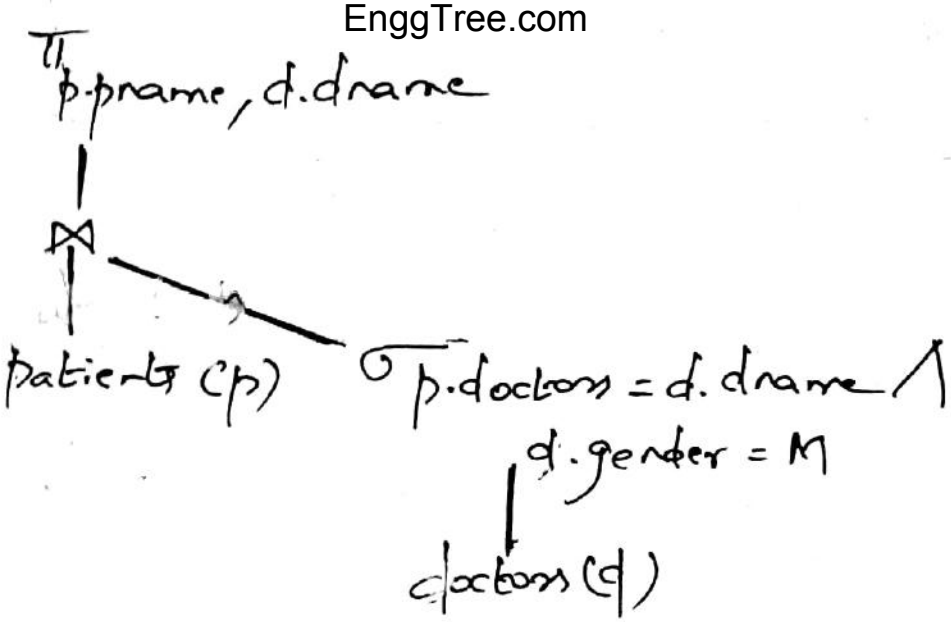
Select p.pname, d.dname from patients p, doctors d where p.doctors = d.dname and d.gender = 'M';

RAEX

$$\Pi_{p.pname,\, d.dname}\left(\sigma_{p.doctors = d.dname \,\wedge\, d.gender = M}(p \bowtie d)\right)$$

QEVP





An evaluation plan defines exactly what algorithm should be used for each operation, and how the execution of the operations are to be co-ordinated.

The cost of QExp is estimated using statistics estimated by catalog information coupled with cost estimates for various algorithms and evaluation methods.

The cost-based optimizer explores the space of all query-evaluation plans that are equivalent to the given query and chooses the least estimated cost.

Advantages

1. Rather than considering time constraints, adapts to client requirements.
2. Speed of query retrieval increases.

Disadvantages

1. Exploring the space of all possible plans may be too expensive for complex queries.
2. Not so accurate.

Heuristic based Optimization

Heuristic optimizer (or) Rule-based optimizer tries to minimize the number of accesses by reducing the number of tuples and number of columns to be searched.

It is less expensive than that of Cost based optimization. It is based on some heuristic rules, by which optimizer can decide Optimized QEVP.

Important heuristic rules are:

1. Perform selection early (reduces the number of tuples)
2. Perform projection early (reduces the number of attributes/columns)
3. Perform most restrictive selection and join operations before other similar operations with smallest result size.
4. Avoid cartesian products.

Example of two rules

1. perform selection as early as possible

original query:

Select * from branch, customer where branch.name = 'Adyar' and Customer.City = 'chennai';

Transformed Query:

Select * from (Select * from branch where branch.name = 'Adyar'), (Select * from Customer where customer.city = 'chennai');

Performance enhancement:

Suppose there are 100 tuples in branch and 100 tuples in customer table respectively.

Original Query:

100x100 => 10000 tuples fetched.

Optimized query:

Selection performed early, hence say only 10 and 20 tuples selected.

So 10x20 => 200 tuples fetched.

2. Perform projection as early as possible

Original query:

Selec branch.id, customer.id from branch, customer where branch.name = 'Adyar';

Optimized query:

Select * from (Select branch.id from branch) t, (Select customer.cid from Customer) where t.name = 'Adyar';

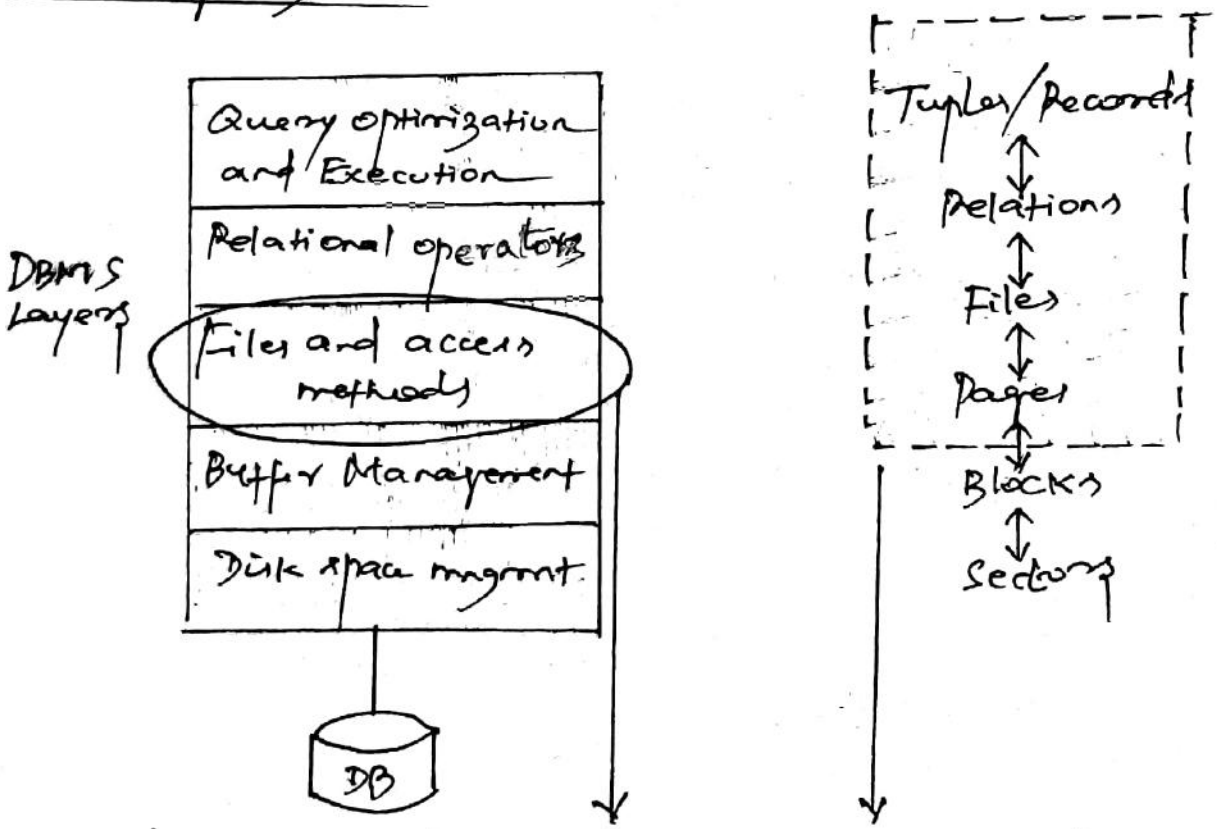
Performance Enhancement

1. Projection operations reduce size of relations.
2. Reduces the number of columns in relation & hence relation size reduces.

Disadvantages

1. Parser has certain constraints like it takes only DML queries (Select queries) and not any DDL queries
2. Also some of the clauses of SQL such as Exists, Not Exist and Order by is not taken into consideration.
3. Transparency for user application is not possible

File organization



* Stores records in a file in a collection of disk pages
* Keeps track of pages allocated to each file
* Tracks available space within pages allocated to file.

File Systems organization organizes data carefully to support fast access to desired subsets of records.

In a database, we have lots of data. Each data is grouped into related groups called tables. Each table will have lots of related records.

Any user will see these records in the form of tables in the screen. But these records are stored as files in the memory. Usualy one file will contain all the records of a table.

To access these files, we need to store them in certain order, so that it will be easy to fetch the records. It is same as indexes in books (or) a telephone directory.

Storing the files in certain order is called as file organization. There are various methods of file organizations. Some of them are

1. Sequential access file organization
2. Direct access file organization
3. Indexed sequential access file organization
4. Heap file organization

Sequential Access

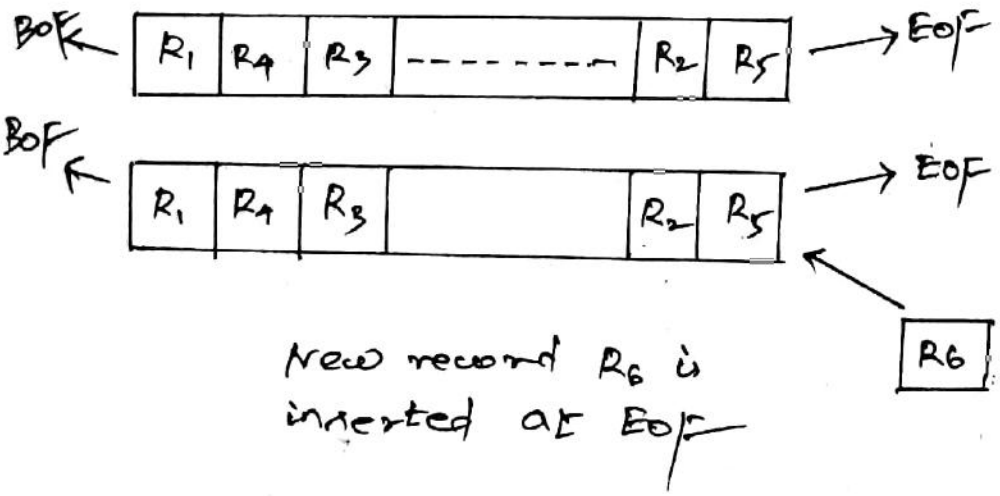
* It is one of the simple methods of file organization.
* Here each file/records are stored one after another in sequential order.

This can be achieved in two ways.

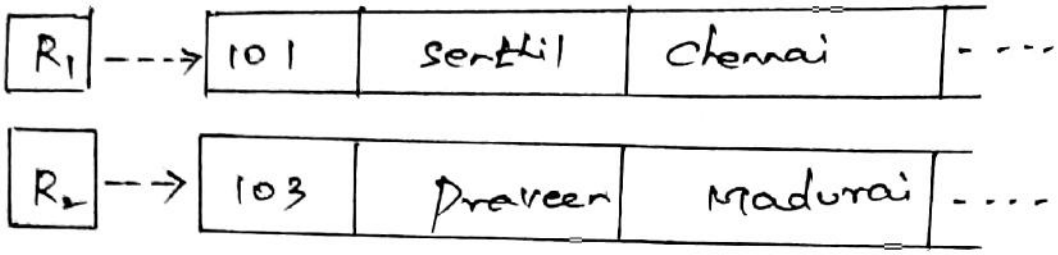
1. Pile file method
2. Sorted file method

Pile file method

* Records are stored one after another as they are inserted into the tables
* when a new record is inserted, it is placed at the end of the file.
* In case of any modification or deletion of record, the record will be searched in the memory blocks. once it is found, it will be marked for deleting and new block of record is entered.

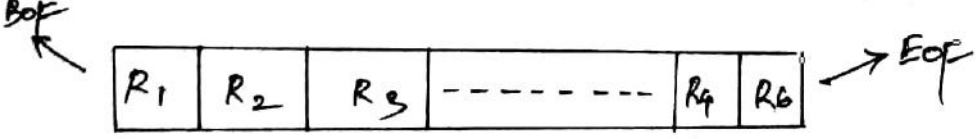


In the previous figure, $R_1, R_2, R_3, \ldots$ are the records. They contain all the attributes of a row. (i.e) when we say a student record, it will have his rollno, name, address, DoB, etc.

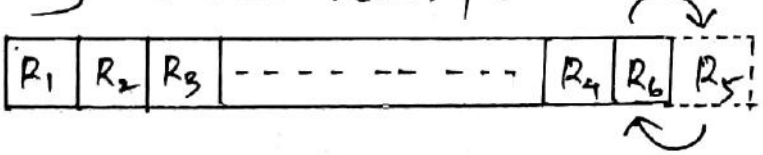


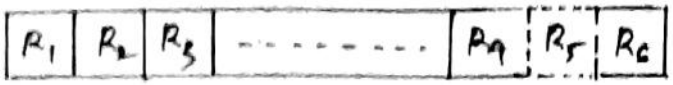
Sorted File method

* Records are sorted (either ascending or descending) each time they are inserted into the system.
* Sorting of records may be based on the primary key (or) any other column.
* Whenever a new record is inserted, it will be inserted at the Eof, and then it will sort, either ascending or descending based on key value and placed at the correct position.
* In case of update, it will update the record and then sort the file to place the updated record in the right place.
* Same is the case with delete.



Inserting a new record :





Advantages

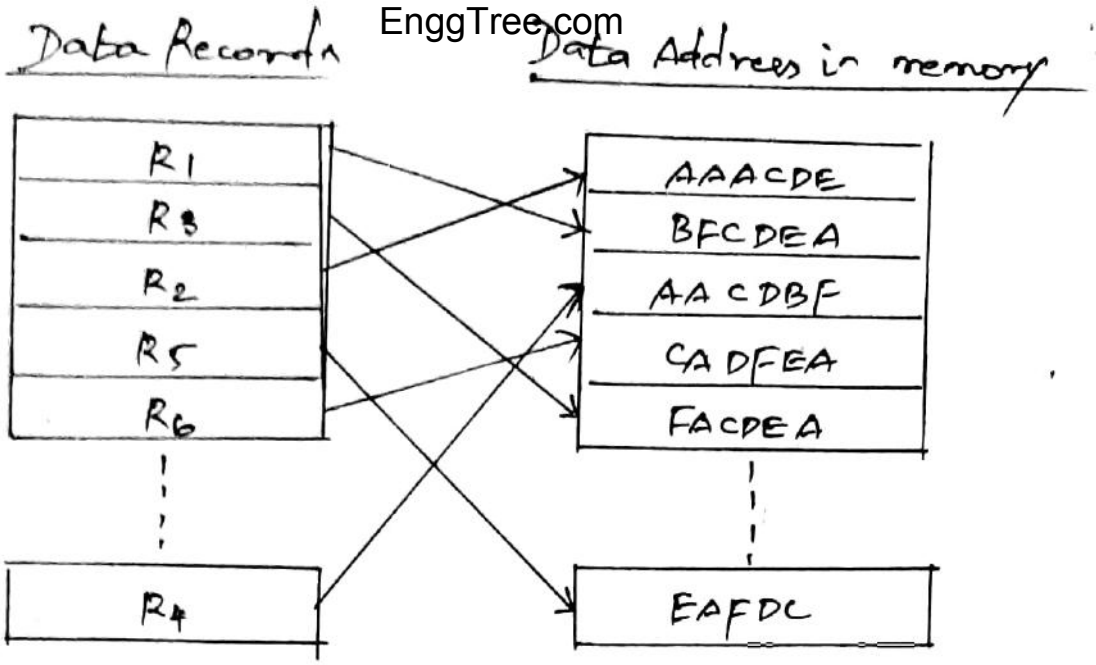
1. Simple to understand.
2. Easier to organize, maintain.
3. Economical

Disadvantages

1. Entire file has to be processed.
2. Transactions must be sorted in a particular sequence before processing
3. Time consuming searching.
4. High data redundancy.
5. Random access is not possible.

Direct Access or Random Access

* In this method, hash function is used to calculate the address of the block to store the records.
* The hash function can be any simple or complex mathematical function.
* The hash function is applied on some columns / attributes — either key (or) non-key columns to get the block address.
* Hence each record is stored randomly irrespective of the order they come.
* If the hash function is applied on key column, then that column is called hash key.
* If the hash function is applied on non-key column, then the column is hash column.



* When a new record has to be inserted, the address is generated by applying the hash function to the key column and the record is directly inserted.
* Same is the case with delete and update.
* Each record will be stored randomly in the memory.

Advantages

1. Records need not be sorted after any transaction.
2. Since block address is known by hash function, accessing
3. Can handle multiple transactions, since each record is independent of other.
4. Suitable for online transaction systems like online banking, ticket reservation, etc.

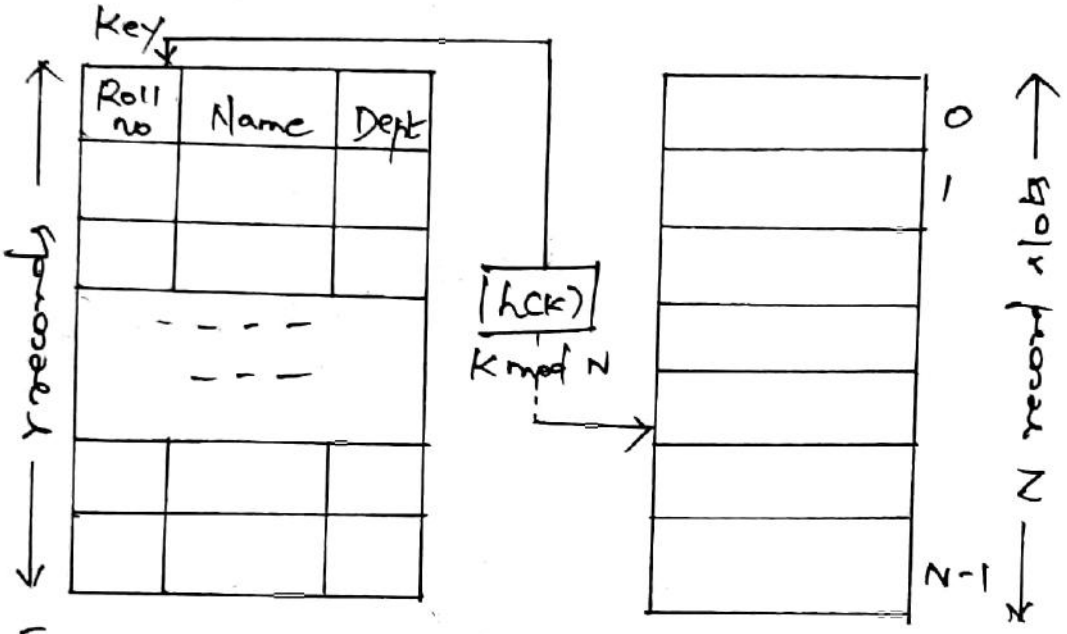
Disadvantages

1. This method may accidentally delete the data.

2. Since all the records are randomly stored, they are scattered in the memory. Hence memory is not efficiently used.

3. Expensive hard disks are needed to store the records.

Hash function h(K) = K mod N



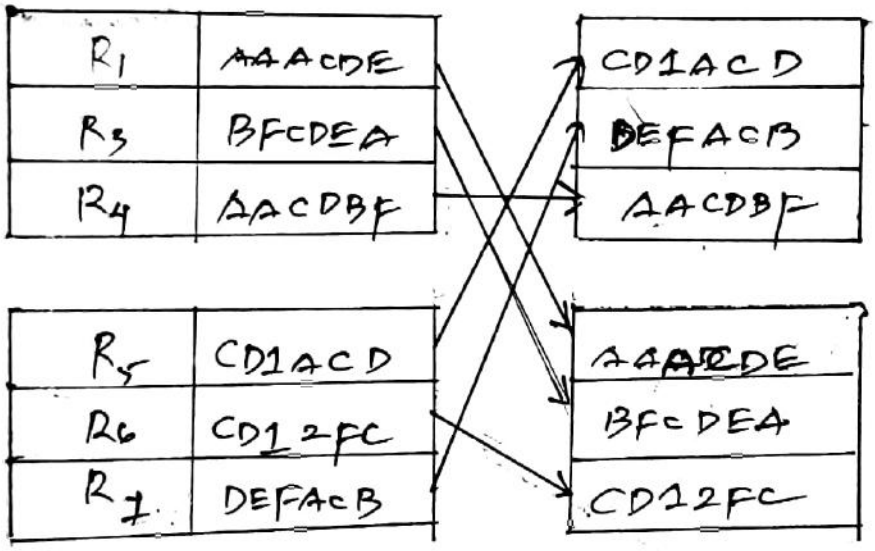
Eg:

* Key is Roll no (six digits)
* Assume we have N = 100,000 record slots which has addresses numbered from 00000 – 99999
* ∴ h(K) = Roll no mod 100000

085768 → 085768 mod 100000 → 85768
134281 → 134281 mod 100000 → 34281
101004 → 101004 mod 100000 → 1004
100000 → 100000 mod 100000 → 0

Indexed Sequential Method

* This is an advanced sequential file organization method.
* Here records are stored in order of primary key in the file.
* For each primary key, an index value is generated and mapped with the record.
* This index is nothing but the address of the record in the file.



* In this method, if any record has to be retrieved based on its index value, the data address is fetched and the record is retrieved from memory.

Features

1. It combines the features of both sequential and direct access file organizations.
2. Here records are stored randomly on a direct access device such as

magnetic disk by a primary key.

3. Hence we can access either sequentially or randomly using the index.

Advantages:

1. Both sequential and random access is possible.
2. Accessing of record is fast.

Disadvantages:

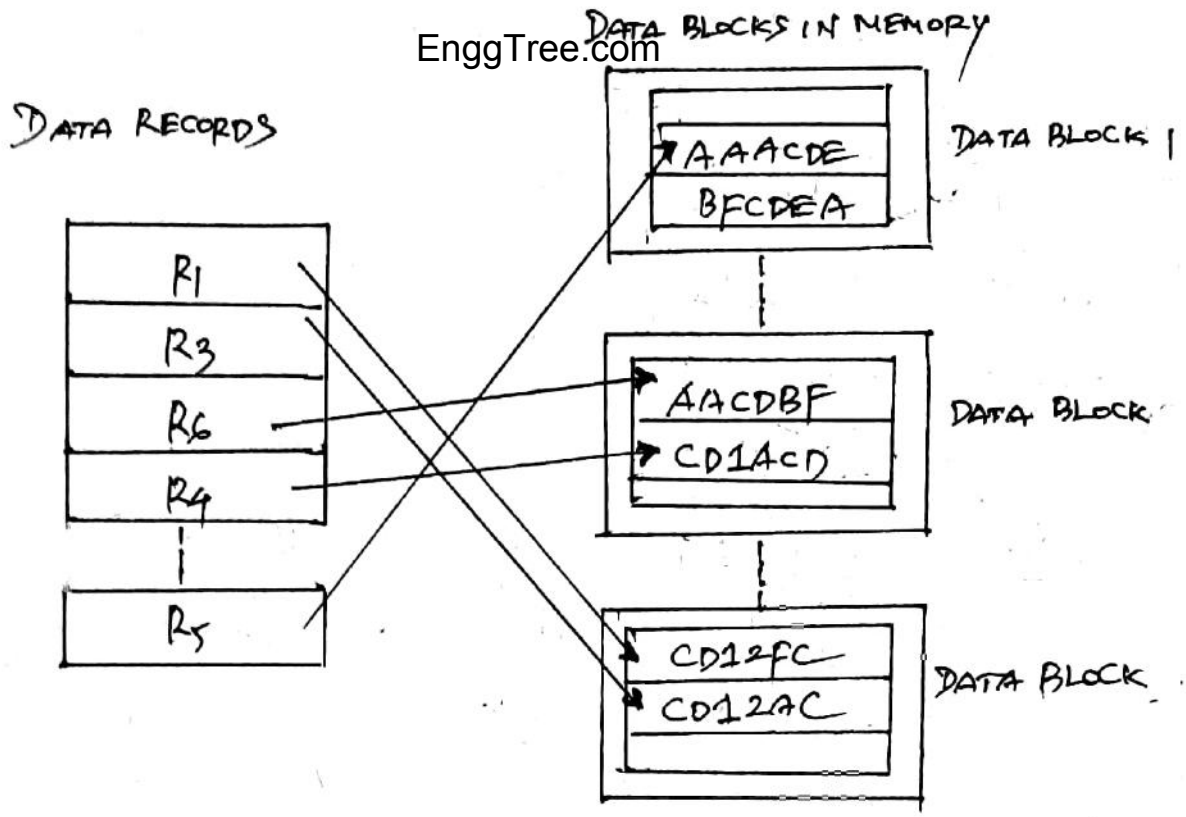
1. More storage space is needed because of the presence of index.
2. Less efficient in terms of storage space
3. It requires special software which is expensive.

Heap File Organization

This is the simplest form of file organization. Here records are inserted at the end of the file as they are inserted. There is no sorting or ordering of the records. Once the data block is full, the next record is stored in the new block. This block need not be the very next block.

This method can select any block, in the memory to store the new records. It is similar to the pile file in the sequential method, but here data blocks are not selected sequentially.

They can be any data blocks in the memory. It is the responsibility of the DBMS to store the records and manage them.



When a record has to be retrieved, we need to traverse from the beginning of the file till we get the requested record. Hence fetching the records in very huge tables is time consuming.

In case of deletion, the same process for retrieving should be followed. For small files, it can be deleted quickly and for larger files, it is time consuming.

In addition to that, while deleting a record, the record will be deleted from the data block. But it will not be freed and it cannot be reused. Hence as the number of record increases, the memory size also increases and hence the unused memory space should be freed by DBA periodically, to increase the efficiency of the database to perform better.

RAID [Redundant Array of Independent Disks]

* It is a disk organization technique which manage a large number of disks, providing a view of a single disk of
  - High capacity and high speed by using multiple disks in parallel.
  - High reliability by storing data redundantly, so that data can be recovered even if a disk fails.
* RAID employs the technique of disk mirroring or disk striping, which involves partitioning each drive's storage space into units ranging from a sector (512 bytes) upto several megabytes.
* The stripes of all the disks are interleaved and addressed in order.

Disk mirroring : is the replication of data to two or more disks. Disk mirroring is a good choice of applications which require high performance and high availability of data.

Disk striping: is the process of dividing a body of data into blocks and spreading the data blocks across multiple storage devices.

Fault tolerance: is the property that enables a system to continue operating properly in the event of failure, of some of its components.

The fault tolerance mechanism used here is the parity information. Data recovery is accomplished by calculating the Exclusive OR (XOR) of the information recorded on the other drives.

Eg:

A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	0

Consider A is the information stored in disk1 and B is the information stored in disk 2. Disk 3 contains o/p [parity information] ie x-ored value of disk1 and disk 2.

If any one of the disks failed, with the other disk and parity, the information of the failed disk can be retrieved back.

Logic: 0 X-OR 0 => 0

DISK1 X-OR DISK2 ⇒ parity disk

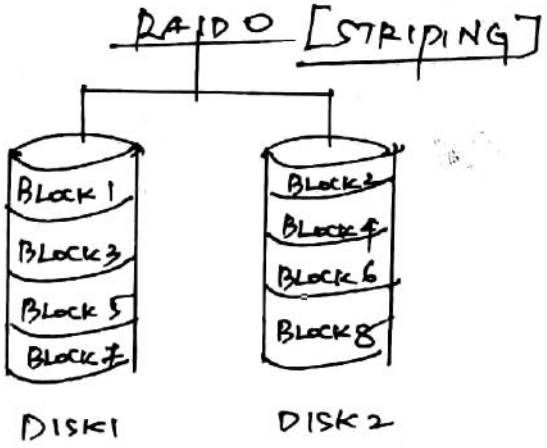
If DISK1 fails [inform 0 is lost], the o/p is 0 and the other i/p is also zero, ie [DISK2 & parity disk are available]. So using the above truth table the i/p $\frac{\text{disk2}}{\text{with}}$ 0 should definitely contain the other i/p as $\frac{\text{disk1}}{0}$ if the $\frac{\text{parity}}{\text{o/p}}$ is 0. Hence disk1 can be retrieved back.

Standard RAID levels

RAID 0: uses block-level striping.

Non redundant.

used in high-performance applications where data loss is not critical.

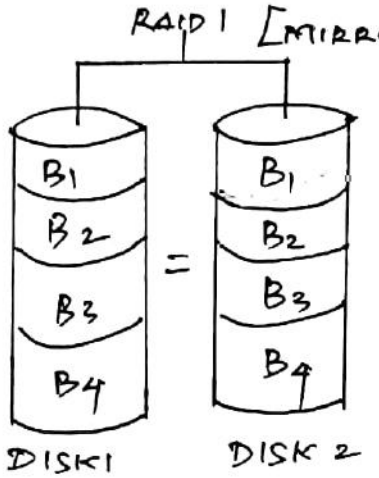


RAID 1: uses disk mirroring.

This configuration contains atleast two drives that duplicate the storage of data.

There is no striping.

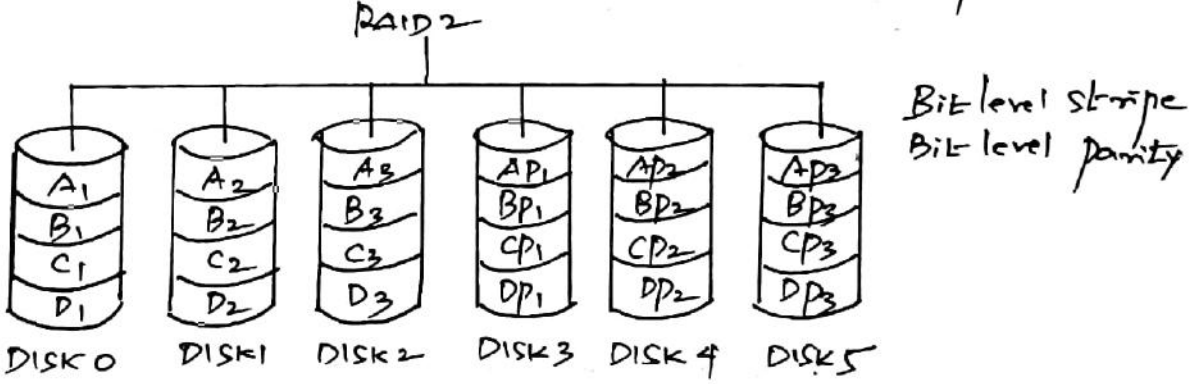
Read performance is improved, since either disk can be read at the same time.



RAID 2: which is rarely used in practice.

Stripes data at the bit level (rather than block level), and uses Hamming code for error correction.

Extremely high data transfer rates are possible.



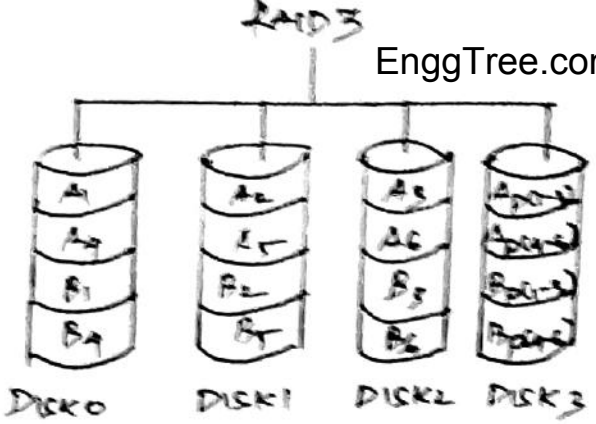
RAID 3: which is rarely used in practice.

Consists of byte-level striping with a dedicated parity disk.

Faster data transfer than with a single disk.

Suitable for applications which require higher transfer rates in long sequential reads & writes.

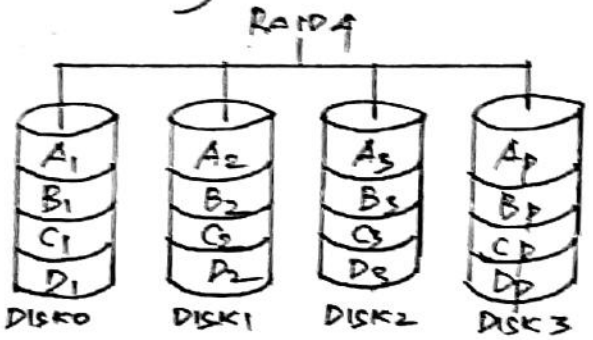
Eg: uncompressed video editing.



Byte level striping with one dedicated parity disk

RAID 4: Consists of block-level striping with a dedicated parity disk.

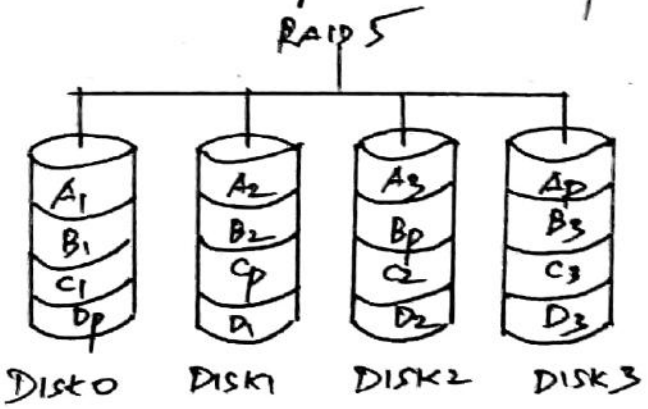
Provides good performance of random reads, while the performance of random writes is low due to the need to write all parity disk to a single disk.



Block level striping with single dedicated parity disk.

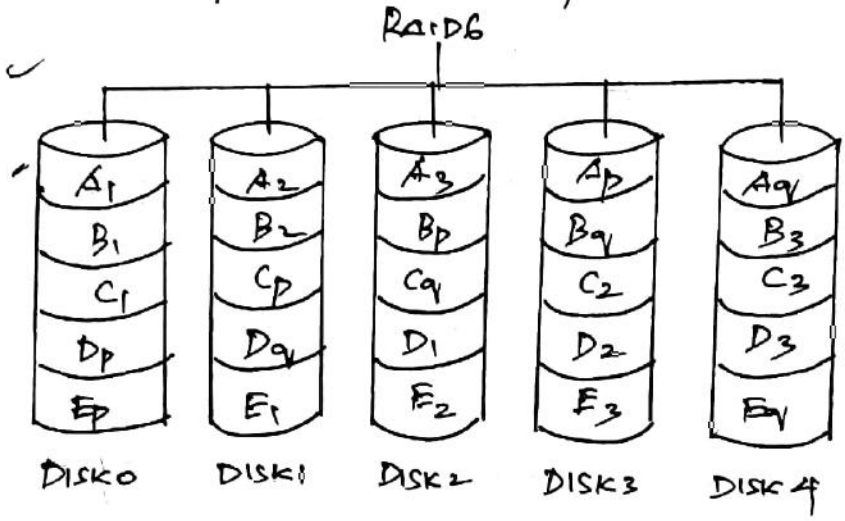
RAID 5: Consists of block-level striping with distributed parity.

Preferred for applications with low update rate, and large amounts of data.



Block level striping with distributed parity

RAID 6: extends RAID 5 by adding another parity block. It uses block-level striping with two parity blocks distributed across all member disks. Use of additional parity allows the disk array to function even if two disks fail simultaneously.



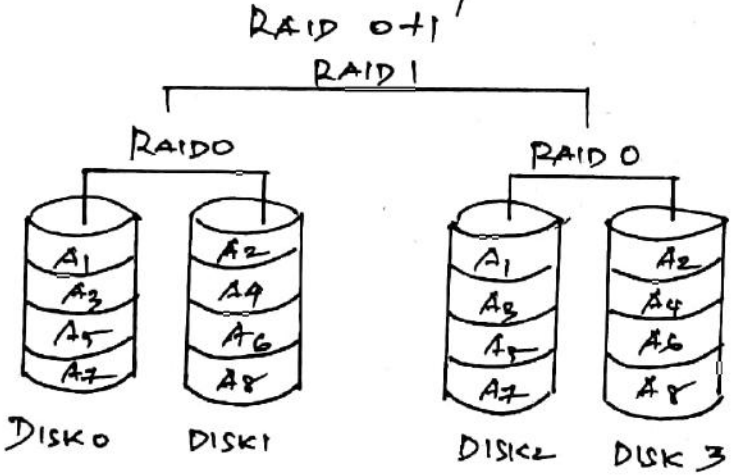
Nested RAID levels

It is also known as Hybrid RAID, which combines two or more of the standard RAID levels to gain performance, additional redundancy or both.

RAID 01 (RAID 0+1)

It is also called RAID 0+1, is a RAID level using a mirror of stripes.

Achieves both replication and striping of data.



Indexing

An index is a small table having only two columns. The first column contains a copy of the primary or candidate key of a table and the second column contains a set of pointers holding the address of the disk block, where that particular key value can be found.

The advantage of using index is that, it makes search operation perform very fast.

Consider, a table which has a row of size 20bytes.

To search record 100, $99 \times 20 = 1980$ bytes of search to be made.

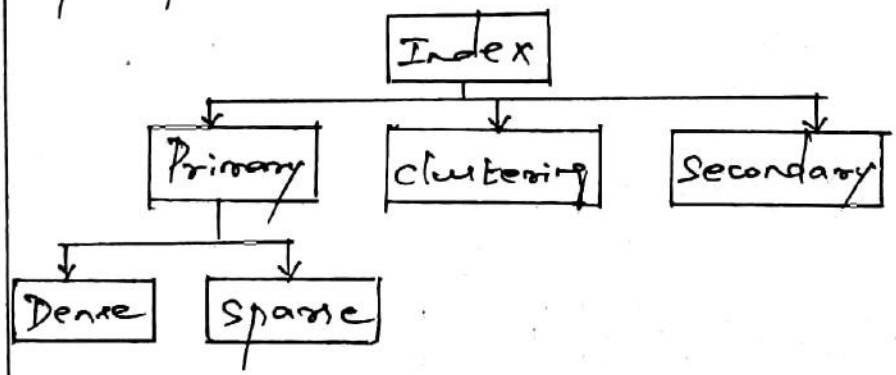
If index is maintained, it will have only two columns, may be each row will be in size of 4 bytes.

To find 100th record, $99 \times 4 = 396$ bytes of data from index to be searched. Then using the address of the address column in index table, the record in actual physical storage can be found.

The result is a much quicker access to the record.

Only minor disadvantage of using index is that it takes up a little more space than the main table.

Types of index

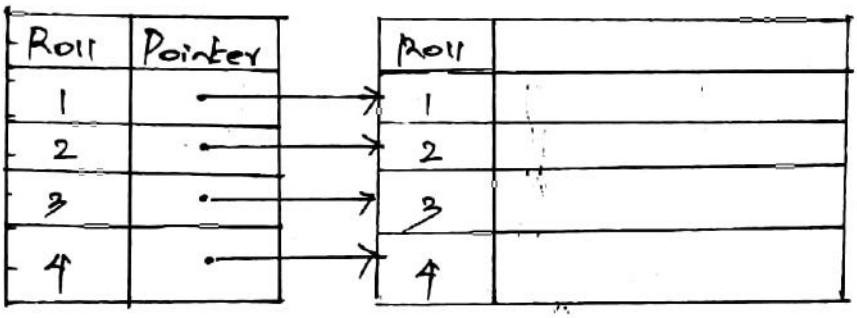


Primary Index

In primary index, there is a one-to-one relationship between the ~~entities~~ entries in the index table and the records in the main table.

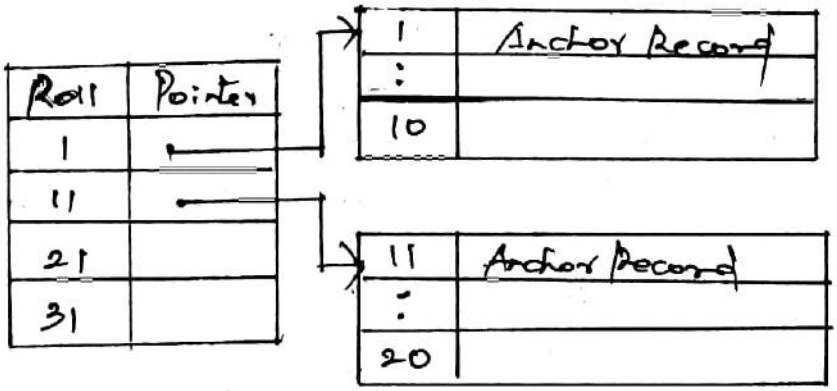
Dense primary index

The number of entries in the index table is the same as the number of entries in the main table. In other words, each and every record in the main table has an entry in the index.



Sparse (or) Non-dense primary index

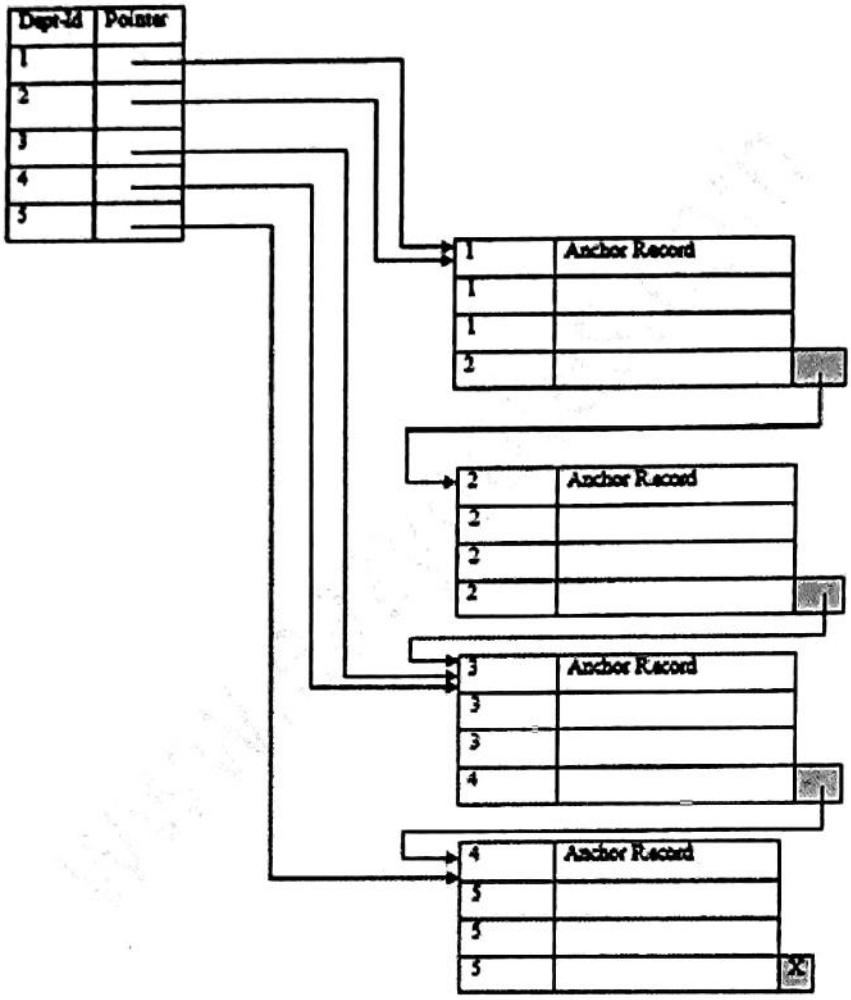
For larger tables, the dense primary index itself begins to grow in size. To keep the size of the index smaller, instead of pointing to each and every record in the main table, the index points to the records in the main table in a gap.



The data blocks have been divided into several blocks, each containing a fixed number of records (Here 10 records / block). The pointer in the index table points to the first record (anchor record) of each block. If 14 is to be searched, the index is searched to find the highest smaller entry which is less than or equal to 14. We have 11. The pointer leads to the record block to find out 14.

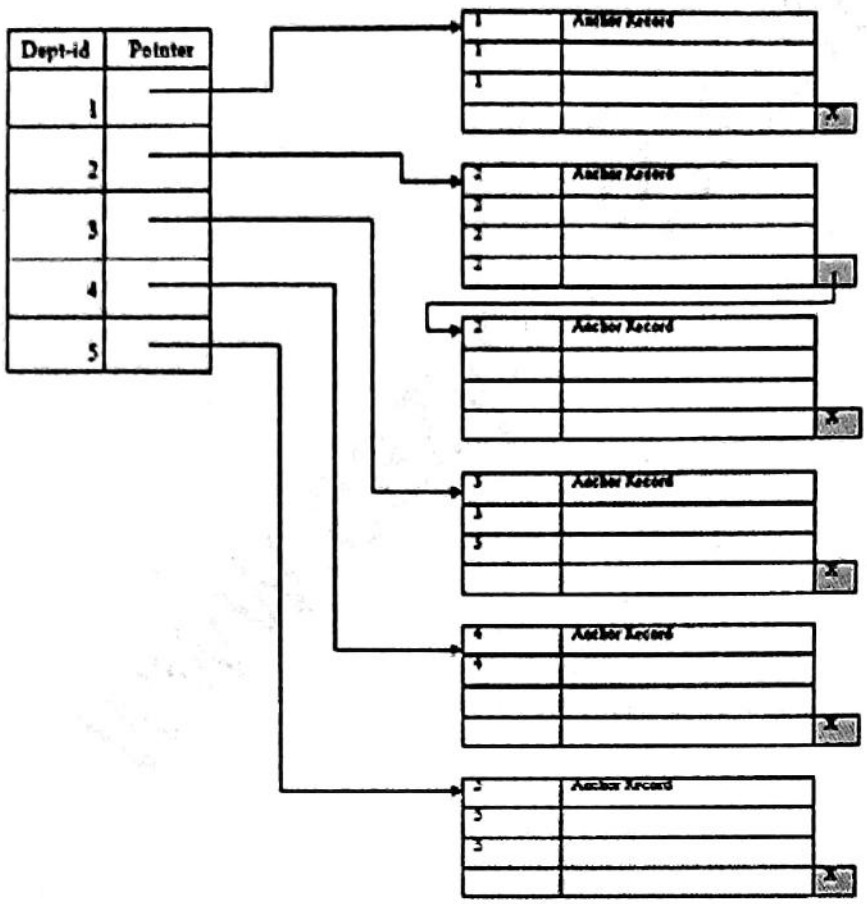
Clustering Index

It may happen sometimes that we are asked to create an Index on a non-unique key, such as Dept-id. There could be several employees in each department. Here we use a clustering index, where all employees belonging to the same Dept-id are considered to be within a single cluster, and the index pointers point to the cluster as a whole.



Let us explain this diagram. The disk blocks contain a fixed number of records (in this case 4 each). The index contains entries for 5 separate departments. The pointers of these entries point to the anchor record of the block where the first of the Dept-id in the cluster can be found. The blocks themselves may point to the anchor record of the next block in case a cluster overflows a block size. This can be done using a special pointer at the end of each block (comparable to the next pointer of the linked list organization).

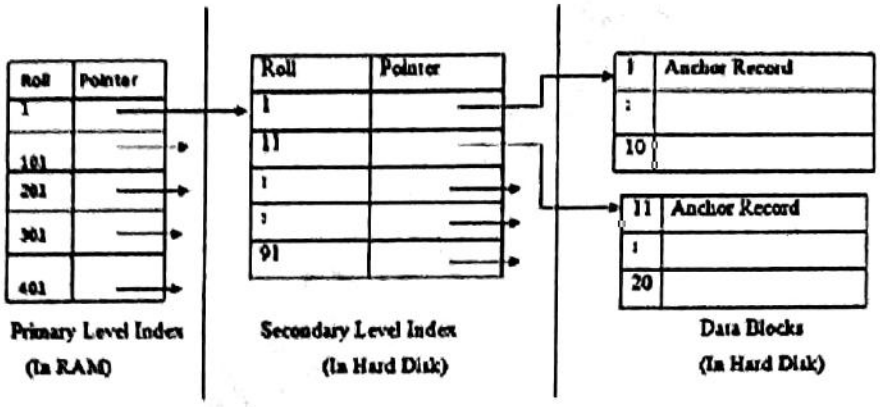
The previous scheme might become a little confusing because one disk block might be shared by records belonging to different cluster. A better scheme could be to use separate disk blocks for separate clusters. This has been explained in the next page.



In this scheme, as you can see, we have used separate disk block for the clusters. The pointers, like before, have pointed to the anchor record of the block where the first of the cluster entries would be found. The block pointers only come into action when a cluster overflows the block size, as for Dept-id 2. This scheme takes more space in the memory and the disk, but the organization in much better and cleaner looking.

Secondary Index

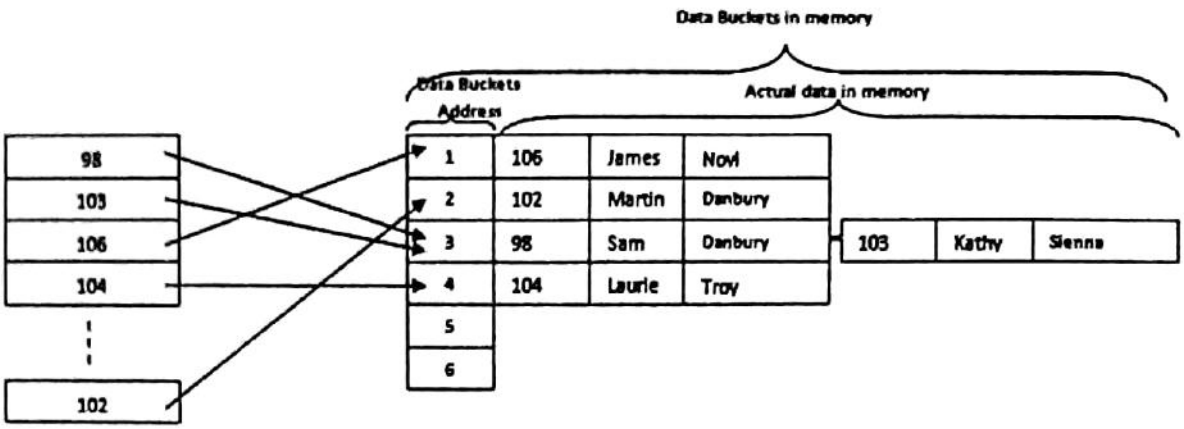
While creating the index, generally the index table is kept in the primary memory (RAM) and the main table, because of its size is kept in the secondary memory (Hard Disk). Theoretically, a table may contain millions of records (like the telephone directory of a large city), for which even a sparse index becomes so large in size that we cannot keep it in the primary memory. And if we cannot keep the index in the primary memory, then we lose the advantage of the speed of access. For very large table, it is better to organize the index in multiple levels. See the following example.



In this scheme, the primary level index, (created with a gap of 100 records, and thereby smaller in size), is kept in the RAM for quick reference. If you need to find out the record of roll 14 now, the index is first searched to find out the highest smaller entry which is less than or equal to 14. We have 1. The adjoining pointer leads us to the anchor record of the corresponding secondary level index, where another similar search is conducted. This finally leads us to the actual data block whose anchor record is roll 11. We now come to roll 11 where a short sequential search is made to find out roll 14.

Multilevel Index

The Multilevel Index is a modification of the secondary level index system. In this system we may use even more number of levels in case the table is even larger.

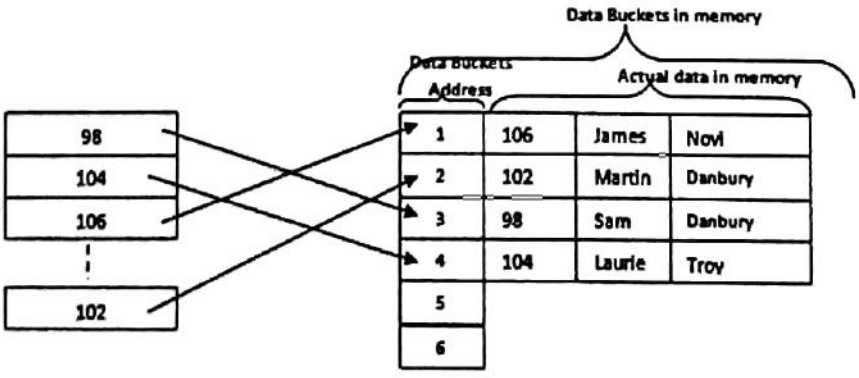


From above two diagrams it now clear how hash function works.

There are two types of hash file organizations – **Static** and **Dynamic** Hashing.

Static Hashing

In this method of hashing, the resultant data bucket address will be always same. That means, if we want to generate address for EMP_ID = 103 using mod (5) hash function, it always result in the same bucket address 3. There will not be any changes to the bucket address here. Hence number of data buckets in the memory for this static hashing remains constant throughout. In our example, we will have five data buckets in the memory used to store the data.



Searching a record

Using the hash function, data bucket address is generated for the hash key. The record is then retrieved from that location. I.e.; if we want to retrieve whole record for ID 104, and if the hash function is mod (5) on ID, the address generated would be 4. Then we will directly got to address 4 and retrieve the whole record for ID 104. Here ID acts as a hash key.

Inserting a record

When a new record needs to be inserted into the table, we will generate a address for the new record based on its hash key. Once the address is generated, the record is stored in that location.

Delete a record

Using the hash function we will first fetch the record which is supposed to be deleted. Then we will remove the records for that address in memory.

Update a record

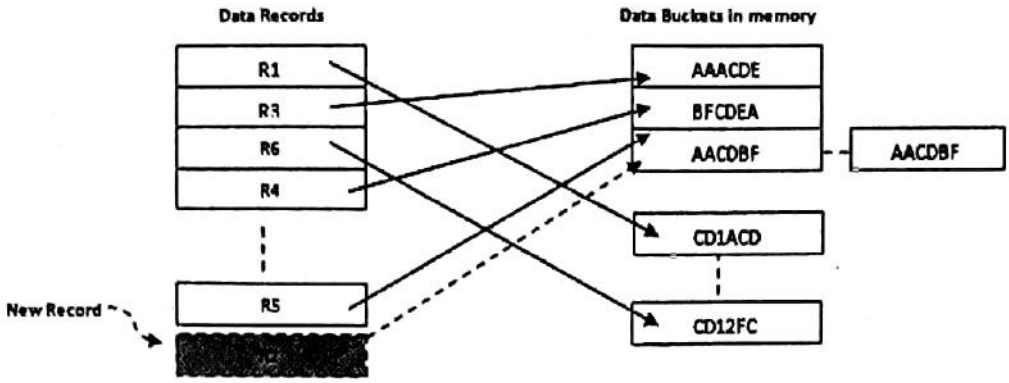
Data record marked for update will be searched using static hash function and then record in that address is updated.

Suppose we have to insert some records into the file. But the data bucket address generated by the hash function is full or the data already exists in that address. How do we insert the data? This situation in the static hashing is called **bucket overflow**. This is one of the critical situations/ drawback in this method. Where will we save the data in this case? We cannot lose the data. There are various methods to overcome this situation. Most commonly used methods are listed below:

Closed hashing

In this method we introduce a new data bucket with same address and link it after the full data bucket. These methods of overcoming the bucket overflow are called closed hashing or **overflow chaining.**

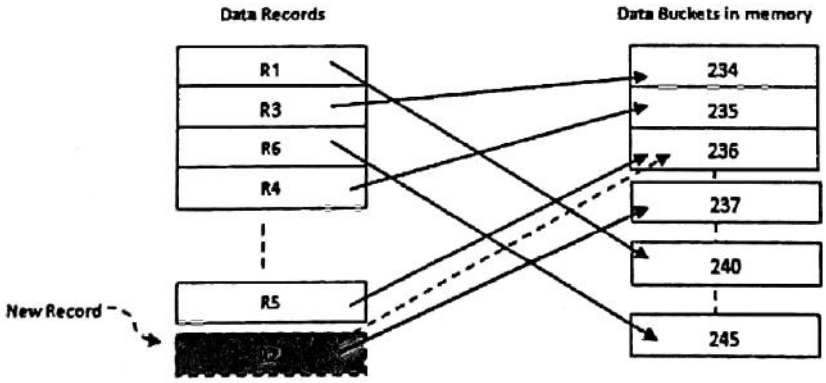
Consider we have to insert a new record R2 into the tables. The static hash function generates the data bucket address as 'AACDBF'. But this bucket is full to store the new data. What is done in this case is a new data bucket is added at the end of 'AACDBF' data bucket and is linked to it. Then new record R2 is inserted into the new bucket. Thus it maintains the static hashing address. It can add any number of new data buckets, when it is full.



Open Hashing

In this method, next available data block is used to enter the new record, instead of overwriting on the older one. This method is called Open Hashing or **linear probing.**

In the below example, R2 is a new record which needs to be inserted. But the hash function generates address as 237. But it is already full. So the system searches next available data bucket, 238 and assigns R2 to it.



In the linear probing, the difference between the older bucket and the new bucket is usually fixed and it will be 1 most of the cases.

Quadratic probing

This is similar to linear probing. But here, the difference between old and new bucket is linear. We use quadratic function to determine the new bucket address.

Double Hashing

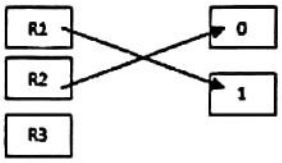
This is also another method of linear probing. Here the difference is fixed like in linear probing, but this fixed difference is calculated by using another hash function. Hence the name is double hashing.

Dynamic Hashing

This hashing method is used to overcome the problems of static hashing – bucket overflow. In this method of hashing, data buckets grows or shrinks as the records increases or decreases. This method of hashing is also known as extendable hashing method. Let us see an example to understand this method.

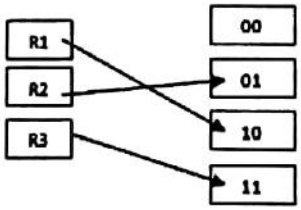
Consider there are three records R1, R2 and R4 are in the table. These records generate addresses 100100, 010110 and 110110 respectively. This method of storing considers only part of this address – especially only first one bit to store the data. So it tries to load three of them at address 0 and 1.

h (R1) →100100
h (R2) →010110
h (R3) →110110



What will happen to R3 here? There is no bucket space for R3. The bucket has to grow dynamically to accommodate R3. So it changes the address have 2 bits rather than 1 bit, and then it updates the existing data to have 2 bit address. Then it tries to accommodate R3.

h (R1) →100100
h (R2) →010110
h (R3) →110110



Now we can see that address of R1 and R2 are changed to reflect the new address and R3 is also inserted. As the size of the data increases, it tries to insert in the existing buckets. If no buckets are available, the number of bits is increased to consider larger address, and hence increasing the buckets. If we delete any record and if the datas can be stored with lesser buckets, it shrinks the bucket size. It does the opposite of what we have seen above. This is how a dynamic hashing works. Initially only partial index/address generated by the hash function is considered to store the data. As the number of data increases and there is a need for more bucket, larger part of the index is consider to store the data.

Advantages of Dynamic hashing

- Performance does not come down as the data grows in the system. It simply increases the memory size to accommodate the data.
- Since it grows and shrinks with the data, memory is well utilized. There will not be any unused memory lying.
- Good for dynamic databases where data grows and shrinks frequently.

Disadvantages of Dynamic hashing

- As the data size increases, the bucket size is also increased. These addresses will be maintained in bucket address tables. This is because, the address of the data will keep changing as buckets grow and shrink. When there is a huge increase in data, maintaining this bucket address table becomes tedious.
- Bucket overflow situation will occur in this case too. But it might take little time to reach this situation than static hashing.

B-Tree

The B-tree is a generalization of a binary search tree in that a node can have more than two children.
B-tree of order m is a tree which satisfies the following properties:
1. The root has atleast two child.
2. Every node has at most m children.
3. Every non-leaf node (except root) has at least [m/2] children.
4. A non-leaf node with k children contains k−1 keys.
5. All leaves appear in the same level, and carry information.

Insertion

Example

key :- 1,12,8,2,25,6,14,28,17,7,52,16,48,68,3,26,29,53,55,45,67.

Order = 5

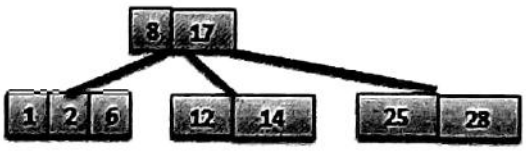
Procedure for adding key in b-tree

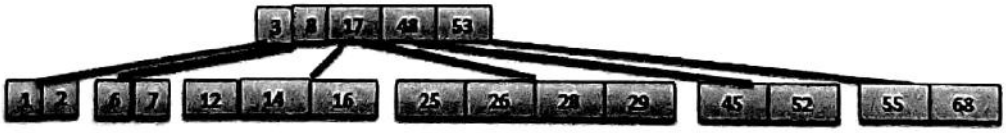
Step1. Add first key as root node.

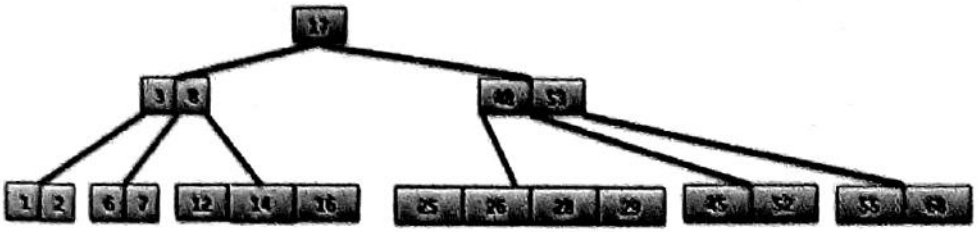
Step2. Add next key at the appropriate place in sorted order.

Step3. Same process applied until root node full. If root node full then spliting process applied.

Some Important steps





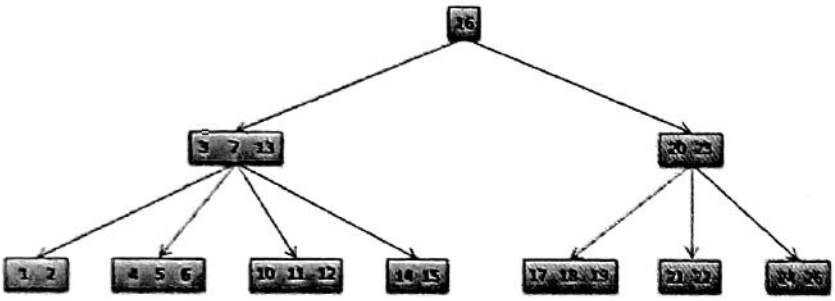


Deletion in B-Tree

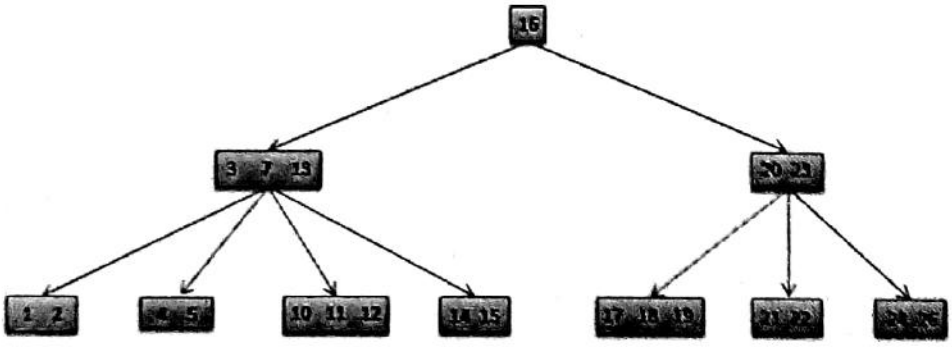
For deletion in b tree we wish to remove from a leaf. There are three possible case for deletion in b tree. Let k be the key to be deleted, x the node containing the key. Then the cases are:

Case-I

If the key is already in a leaf node, and removing it doesn't cause that leaf node to have too few keys, then simply remove the key to be deleted. key k is in node x and x is a leaf, simply delete k from x.



6 deleted



Case-II

If key k is in node x and x is an internal node, there are three cases to consider:

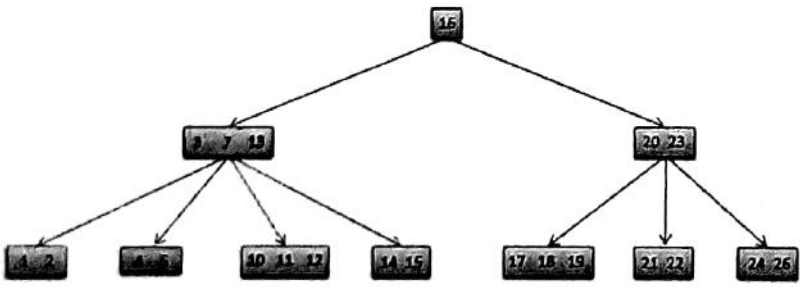
Case-II-a

If the child y that precedes k in node x has at least t keys (more than the minimum), then find the predecessor key k' in the subtree rooted at y. Recursively delete k' and replace k with k' in x

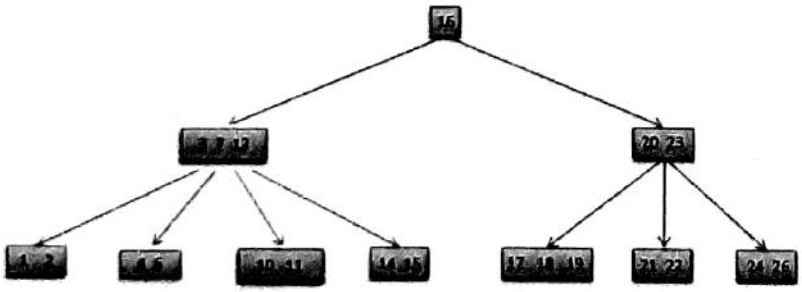
---

Case-II-b

Symmetrically, if the child z that follows k in node x has at least t keys, find the successor k' and delete and replace as before. Note that finding k' and deleting it can be performed in a single downward pass.

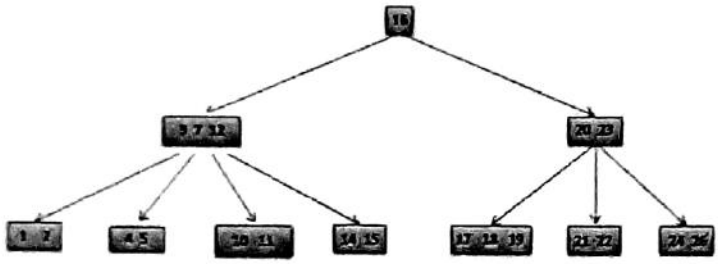


13 deleted

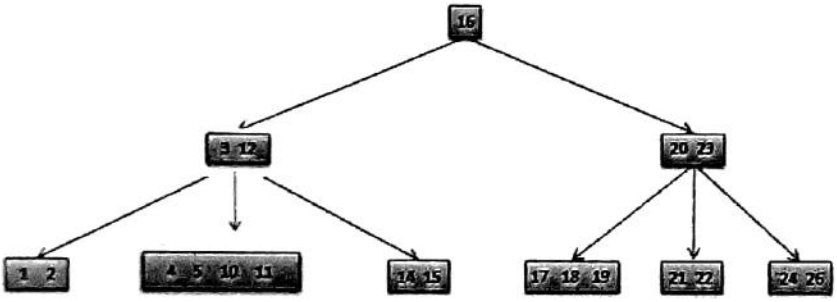


Case-II-c

Otherwise, if both y and z have only t−1 (minimum number) keys, merge k and all of z into y, so that both k and the pointer to z are removed from x. y now contains 2t − 1 keys, and subsequently k is deleted.



7 deleted

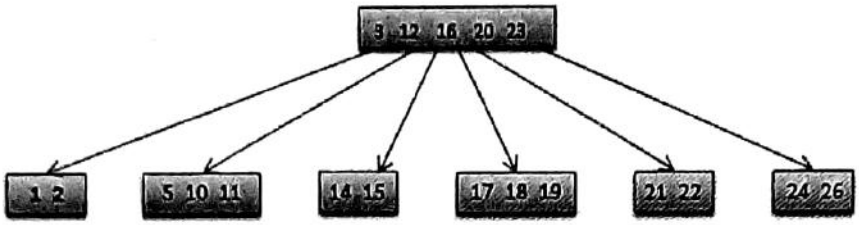


Case-III

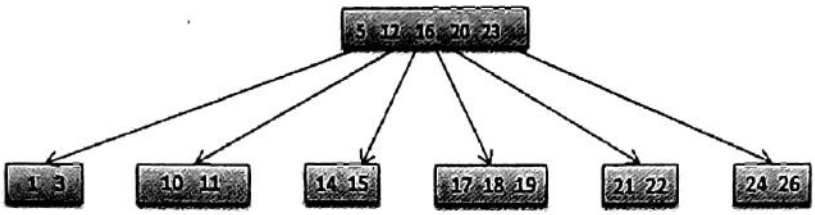
If key k is not present in an internal node x, determine the root of the appropriate subtree that must contain k. If the root has only t − 1 keys, execute either of the following two cases to ensure that we descend to a node containing at least t keys. Finally, recurse to the appropriate child of x.

Case-III-a

If the root has only t−1 keys but has a sibling with t keys, give the root an extra key by moving a key from x to the root, moving a key from the roots immediate left or right sibling up into x, and moving the appropriate child from the sibling to x.

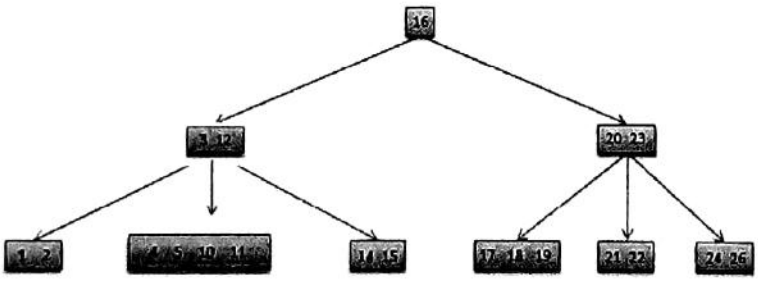


2 deleted

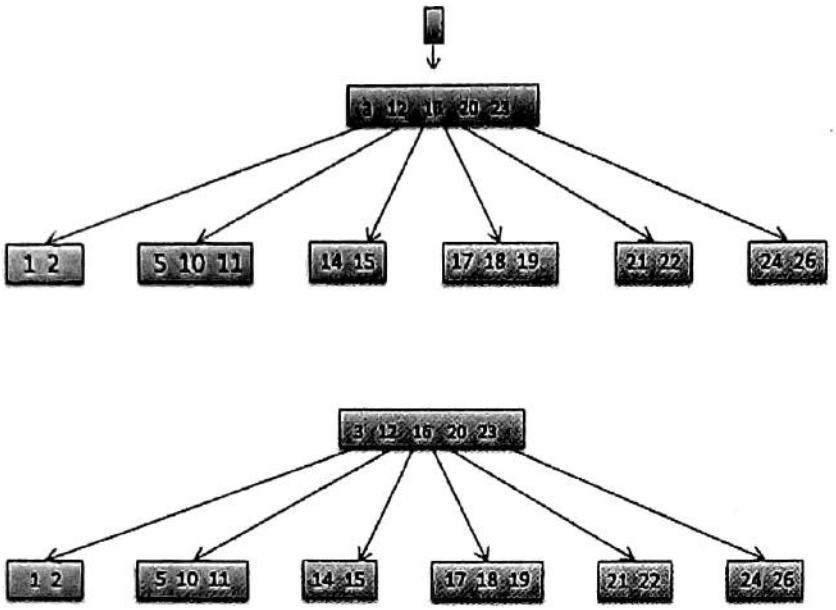


Case-III-b

If the root and all of its siblings have t−1 keys, merge the root with one sibling. This involves moving a key down from x into the new merged node to become the medlan key for that node.



4 deleted



B+ Tree

A B+-tree maintains the following invariants:

- Every node has one more references than it has keys.
- All leaves are at the same distance from the root.
- For every non-leaf node N with k being the number of keys in N: all keys in the first child's subtree are less than N's first key; and all keys in the ith child's subtree ($2 \leq i \leq k$) are between the $(i-1)$th key of n and the ith key of n.
- The root has at least two children.
- Every non-leaf, non-root node has at least $floor(d / 2)$ children.
- Each leaf contains at least $floor(d / 2)$ keys.
- Every key from the table appears in a leaf, in left-to-right sorted order.

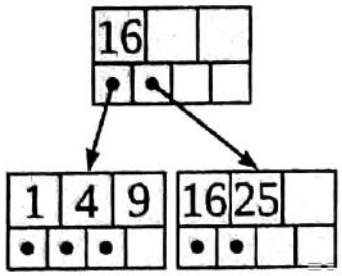
2. Insertion algorithm

Descend to the leaf where the key fits.

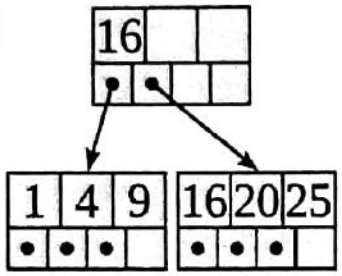
1. If the node has an empty space, insert the key/reference pair into the node.
2. If the node is already full, split it into two nodes, distributing the keys evenly between the two nodes. If the node is a leaf, take a copy of the minimum value in the second of these two nodes and repeat this insertion algorithm to insert it into the parent node. If the node is a non-leaf, exclude the middle value during the split and repeat this insertion algorithm to insert this excluded value into the parent node.

45

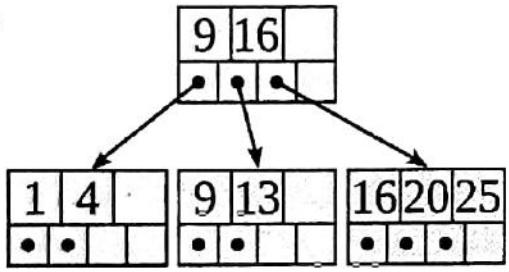
Initial:



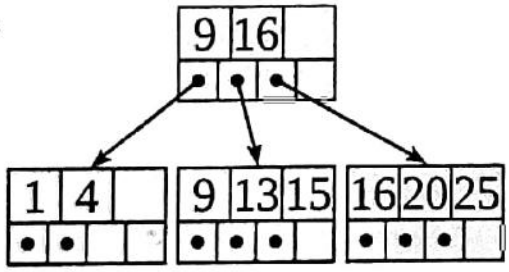
Insert 20:



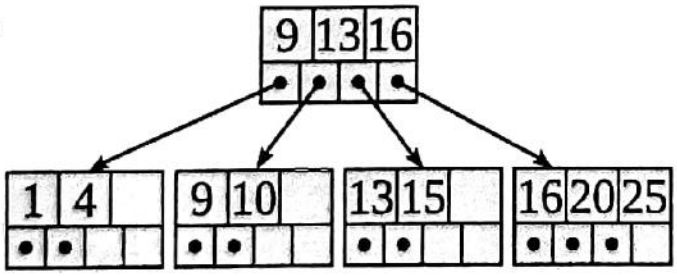
Insert 13:



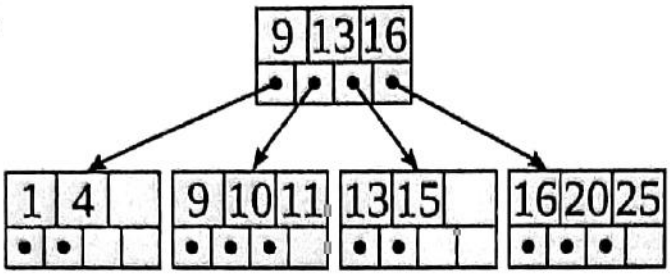
Insert 15:



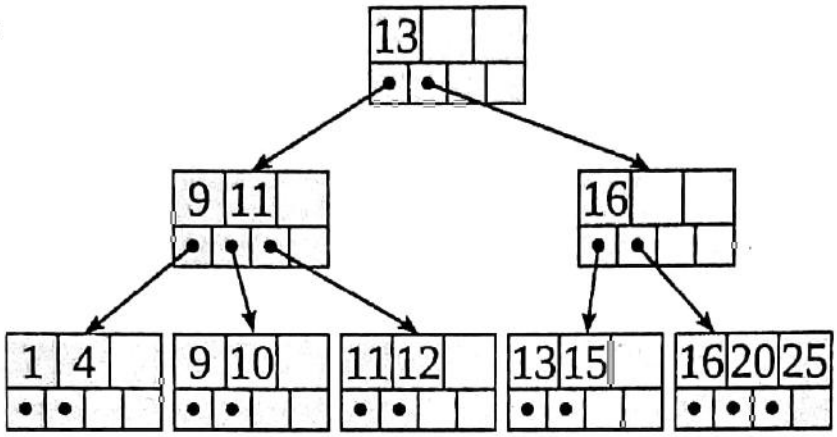
Insert 10:



Insert 11:



Insert 12:

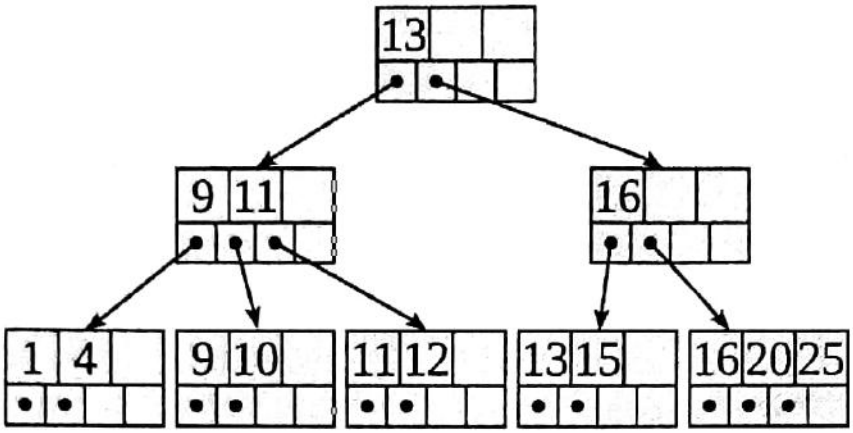


3. Deletion algorithm

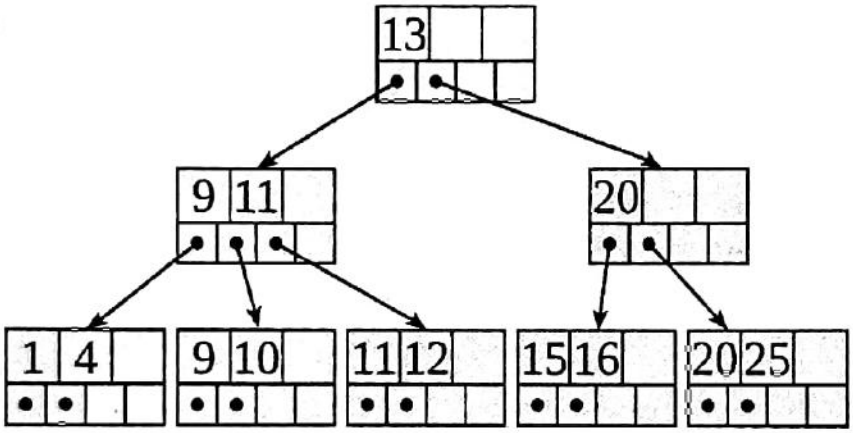
Descend to the leaf where the key exists.

1. Remove the required key and associated reference from the node.
2. If the node still has enough keys and references to satisfy the invariants, stop.
3. If the node has too few keys to satisfy the invariants, but its next oldest or next youngest sibling at the same level has more than necessary, distribute the keys between this node and the neighbor. Repair the keys in the level above to represent that these nodes now have a different "split point" between them; this involves simply changing a key in the levels above, without deletion or insertion.
4. If the node has too few keys to satisfy the invariant, and the next oldest or next youngest sibling is at the minimum for the invariant, then merge the node with its sibling; if the node is a non-leaf, we will need to incorporate the "split key" from the parent into our merging. In either case, we will need to repeat the removal algorithm on the parent node to remove the "split key" that previously separated these merged nodes — unless the parent is the root and we are removing the final key from the root, in which case the merged node becomes the new root (and the tree has become one level shorter than before).

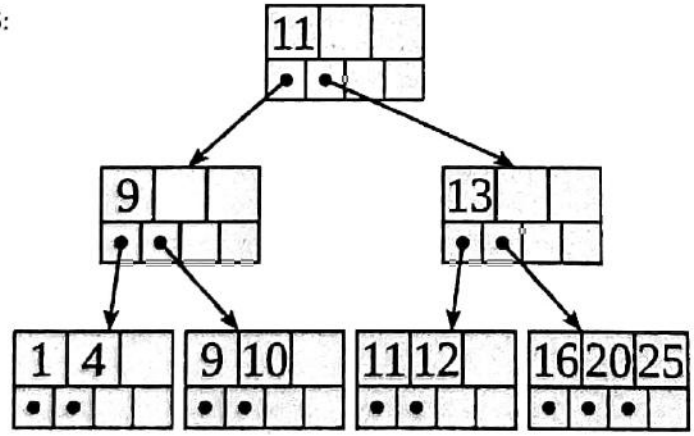
Initial:



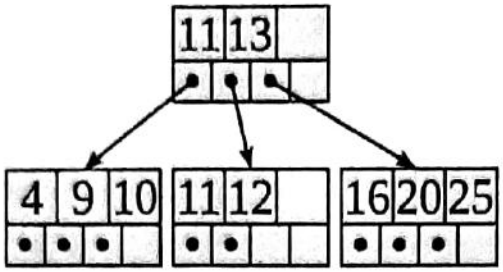
Delete 13:



Delete 15:



Delete 1:



B-Tree

It is an extended version of binary search tree. B-Tree of order 'm' should satisfy the following properties.

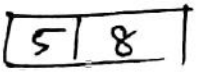
1. Max. no. of children = m
2. Max. no. of keys = m-1
3. Min. no. of children = Ceil (m/2)
4. Min. no. of keys = Ceil (m/2) - 1
5. All leaves appear in the same level.

Eg: Construct a B-Tree of order 3 with the following:
5, 8, 9, 20, 30, 15, 16, 14, 13, 31.

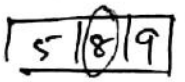
Given: order of B-Tree = m = 3

Then: Max. no. of children = 3
Max. no. of keys = 2
Min. no. of children = 2
Min. no. of keys = 1

Step1:
Add 5, 8

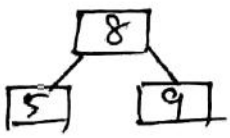


Step 2:
Add 9



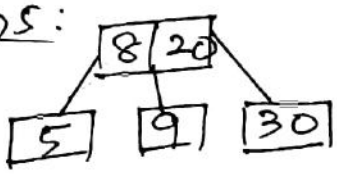
→ Rule violation max key = 2

Step 3:

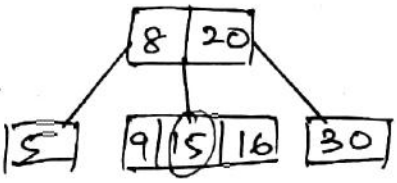


Step 4: Add 20, 30

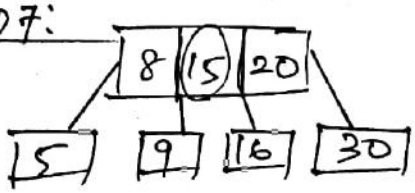
Step 5:



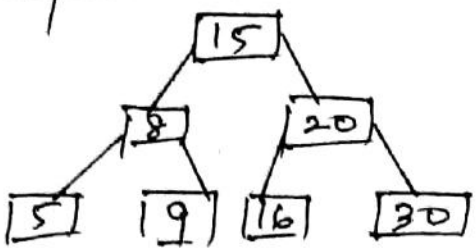
Step 6: Add 15, 16



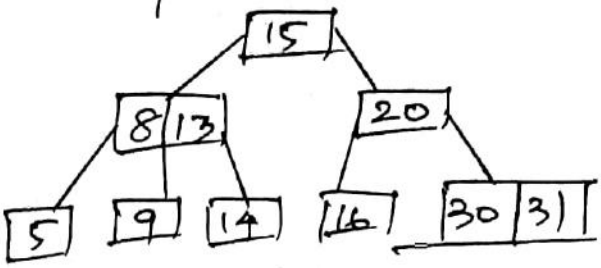
Step 7:



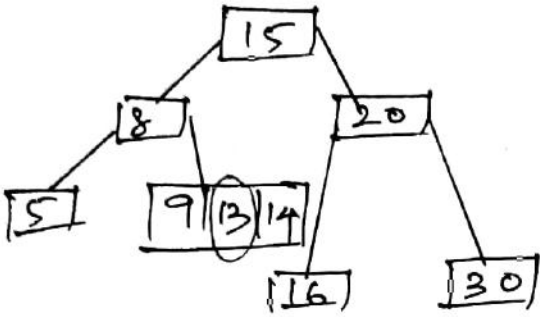
Step 8:



Step 10: Add 31



Step 9: Add 14, 13



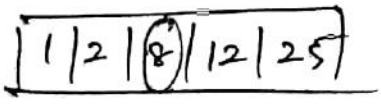
Eg 2: Construct a B-Tree with the following of orders.

1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 67

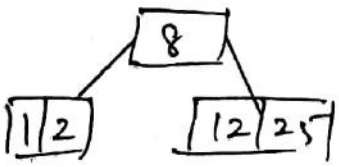
Given: order = m = 5

then: Max. no. of children = 5
Min. no. of children = ceil (m/2) = 3
Max. no. of key = m-1 = 4
Min. no. of key = ceil (m/2) - 1 = 2

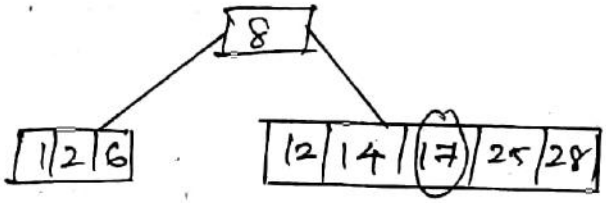
Step 1: Add 1, 12, 8, 2, 25

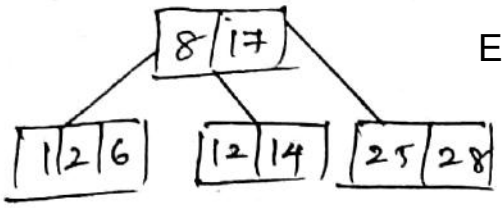


Step 2:

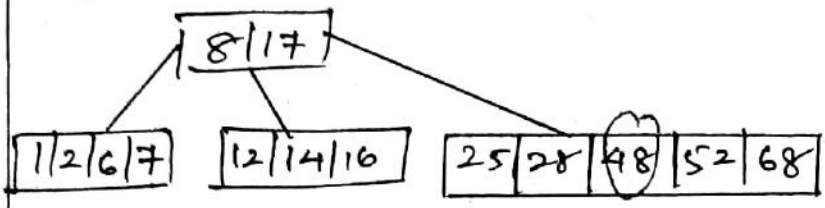


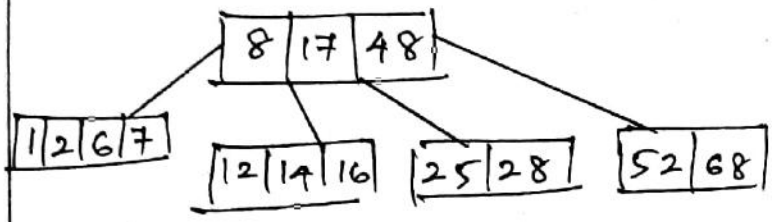
Step 3: Add 6, 14, 28, 17



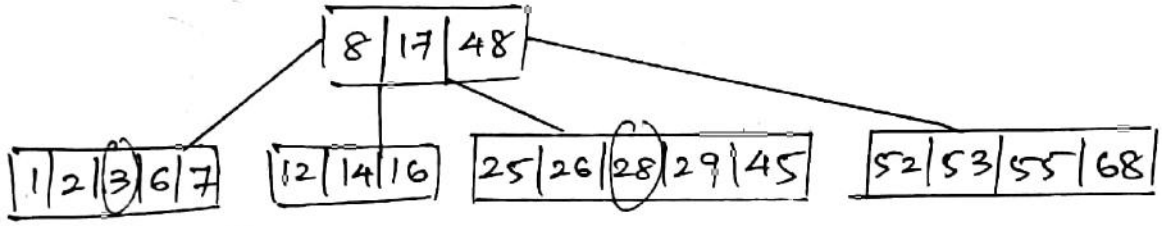


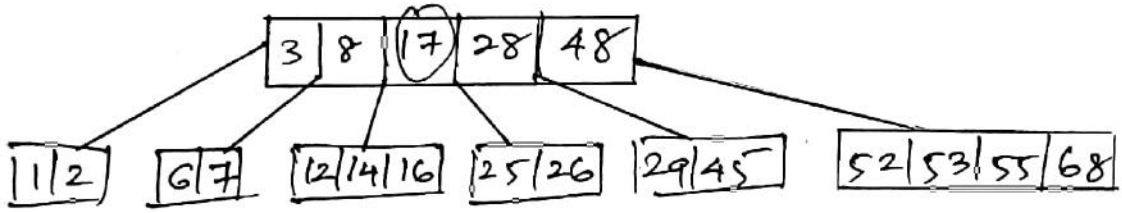
Step 4 : Add 7, 52, 16, 48, 68

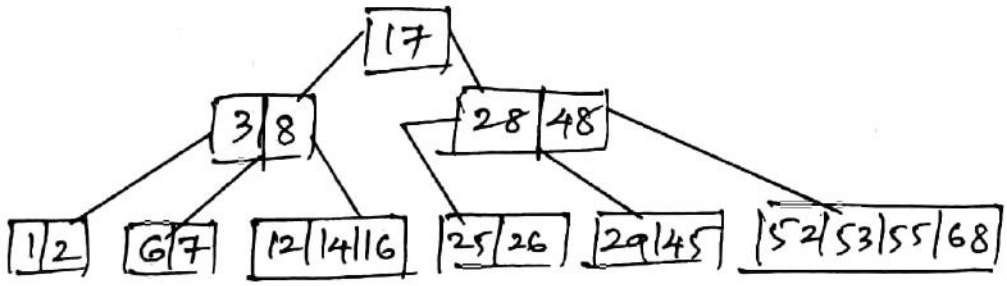




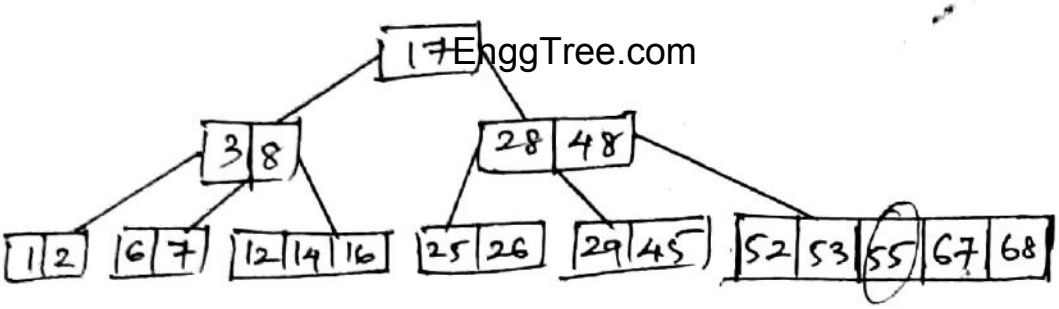
Step 5 : Add 3, 26, 29, 53, 55, 45

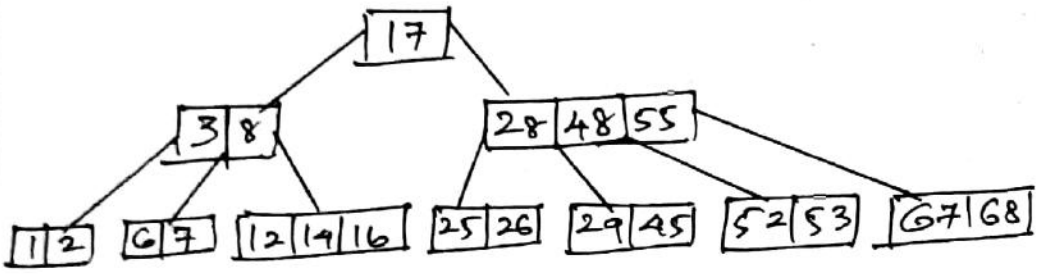






Step 6 : Add 67





Prepared by

Verified by

Approved by

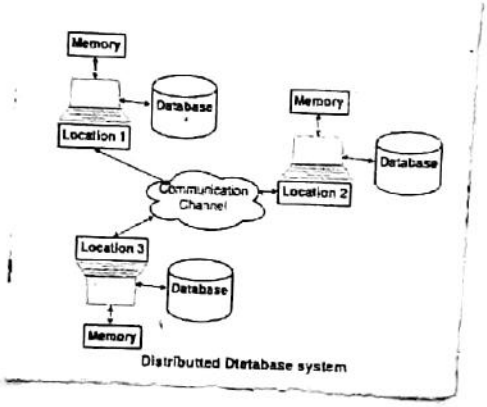
UNIT V ADVANCED TOPICS

Distributed Databases: Architecture, Data Storage, Transaction Processing – Object-based Databases: Object Database Concepts, Object-Relational features, ODMG Object Model, ODL, OQL - XML Databases: XML Hierarchical Model, DTD, XML Schema, XQuery – Information Retrieval: IR Concepts, Retrieval Models, Queries in IR systems.

Distributed databases

Distributed database is a system in which storage devices are not connected to a common processing unit.

Database is controlled by Distributed Database Management System and data may be stored at the same location (or) spread over the interconnected network



Goals of distributed system

Reliability: If one system fails down (or) stops working for some time, another system can complete the task.

Availability: Even if server fails down, another system is available to serve the client request.

Performance: Performance is good, since the databases are available at every

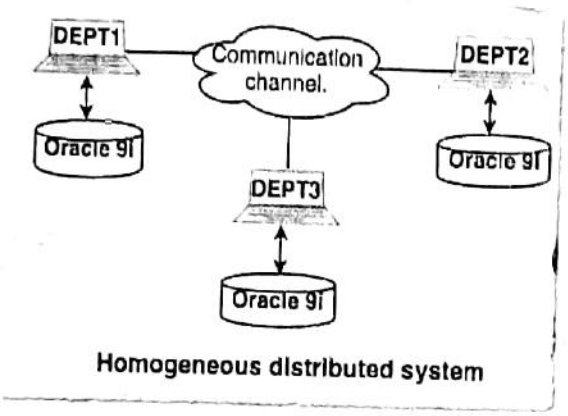
(1)

location which is easy to maintain.

Types of distributed systems

1. Homogeneous distributed databases system

It is a network of two or more databases (with same type of DBMS software) which can be stored on one or more machines. So, in this system data can be accessed and modified simultaneously on several databases in the network.

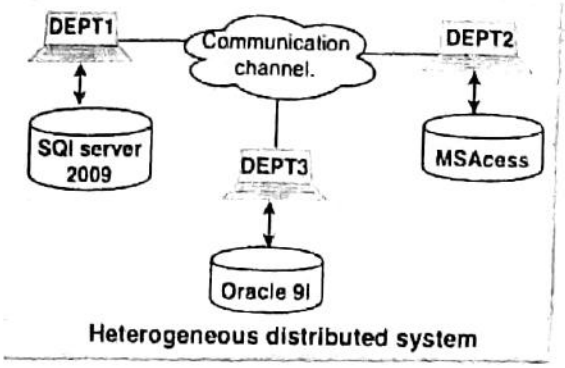


Homogeneous distributed system

2. Heterogeneous distributed database system

It is a network of two or more databases with different types of DBMS software, which can be stored on one or more machines.

In this system, data can be accessible to several databases in the network with the help of generic connectivity (ODBC and JDBC)



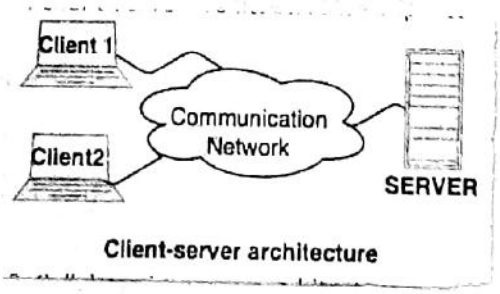
Heterogeneous distributed system

Architecture of distributed DBMS

The basic types of distributed DBMS are as follows:

Client-Server architecture

A client-server architecture has a number of clients and a few servers connected in a network. A client sends a query to one of the servers. The earliest available server solves it and replies.

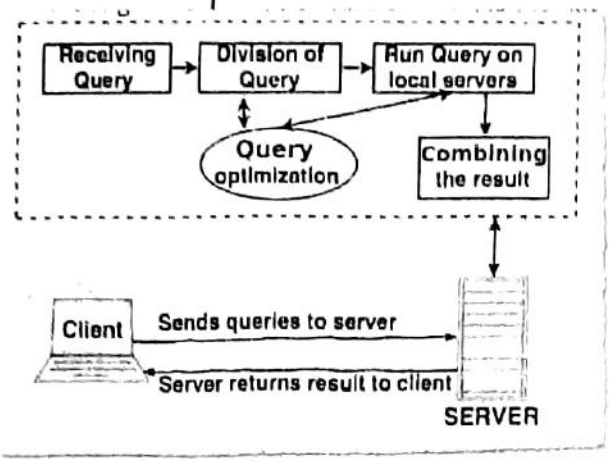


Client-server architecture

Collaborating server architecture

It is designed to run a single query on multiple servers. Servers break single query into multiple small queries and the result is sent to the client. This architecture has a

Collection of database servers. Each server is capable for executing the current transactions across the databases.



Middleware architecture

They are designed in such a way that single query is executed on multiple servers. This system needs only one server which is capable of managing queries and transactions from multiple servers. Middleware architecture uses local servers to handle local queries and transactions.

The softwares are used for execution of queries and transactions across one or more independent database servers, are called as middle ware.

What is fragmentation?

1. The process of dividing the database into a smaller multiple parts is called as fragmentation.
2. These fragments may be stored at different locations.

3. The data fragmentation process should be carried out in such a way that the reconstruction of original database from the fragments is possible.

There are three types of data fragmentation:

Horizontal data fragmentation

It divides a relation (table) horizontally into the group of rows to create subsets of tables.

Vertical fragmentation

It divides a relation vertically into groups of columns to create subsets of tables.

Hybrid fragmentation

It is achieved by performing horizontal and vertical fragmentation together.

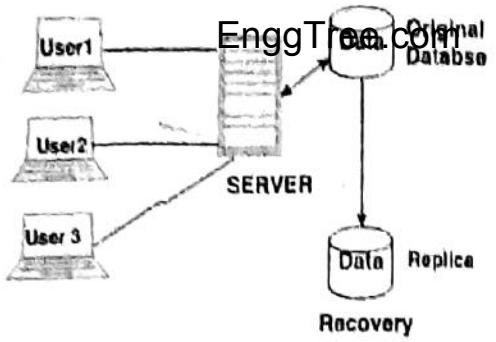
What is data replication?

It is the process in which the data is copied at multiple locations (different computers or servers) to improve the availability of data. Data replication is aimed to increase the availability of data and speed up the query evaluation.

Types of data replication schemes are

Full replication

In this scheme, the database is available to almost every location (or) user in communication n/w.



Full Replication Process In Distributed System

Advantages of full replication

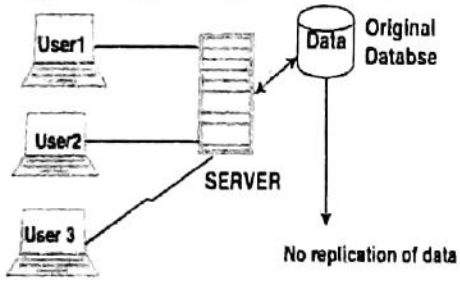
1. High availability of data, as database is available to almost every location.
2. Faster execution of queries.

Disadvantages of full replication

1. Concurrency control is difficult to achieve in full replication.
2. Update operation is slower.

No replication

It means each fragment is stored exactly at one location.



No Replication Process in Distributed Databases

Advantages of no replication

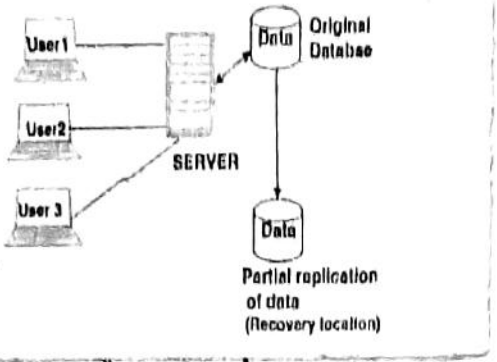
1. Concurrency can be minimized.
2. Each recovery of data.

Disadvantages of no replication

1. Poor availability of data.
2. Slows down the query execution process.

Partial replication

It means only some fragments are replicated from the database.



Advantages of partial replication

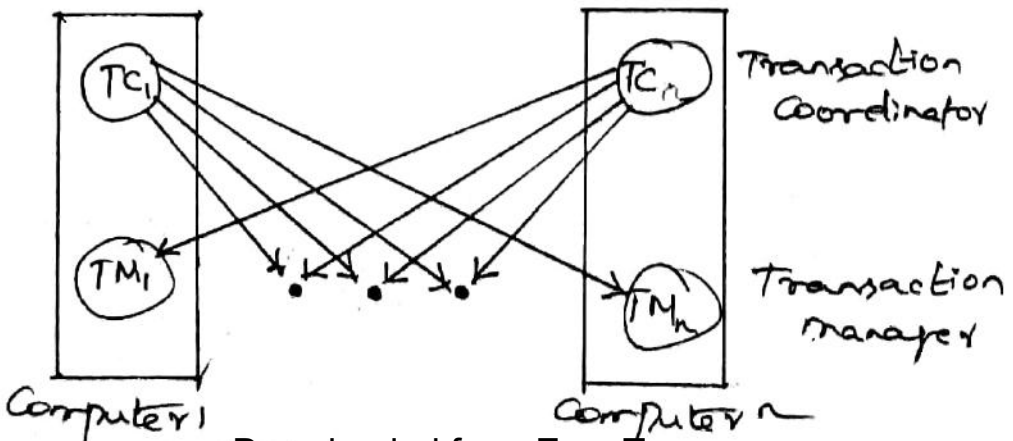
The number of replicas created for fragments depend upon the importance of data in that fragment.

Distributed transactions

Access to various data items in a distributed system is usually accomplished through transactions. There are two types of transactions that we need to consider.

The local transactions are those that access and update data in only one local database, the global transactions are those that access and update data in several local databases.

System architecture



Each site has a local transaction manager, whose function is to ensure the ACID properties of those transactions that execute at that site. The various transaction managers cooperate to execute global transactions.

The transaction manager manages the execution of those transactions that access data stored in a local site.

The transaction coordinator coordinates the execution of various transactions (both local and global) initiated at that site.

Each transaction manager is responsible for

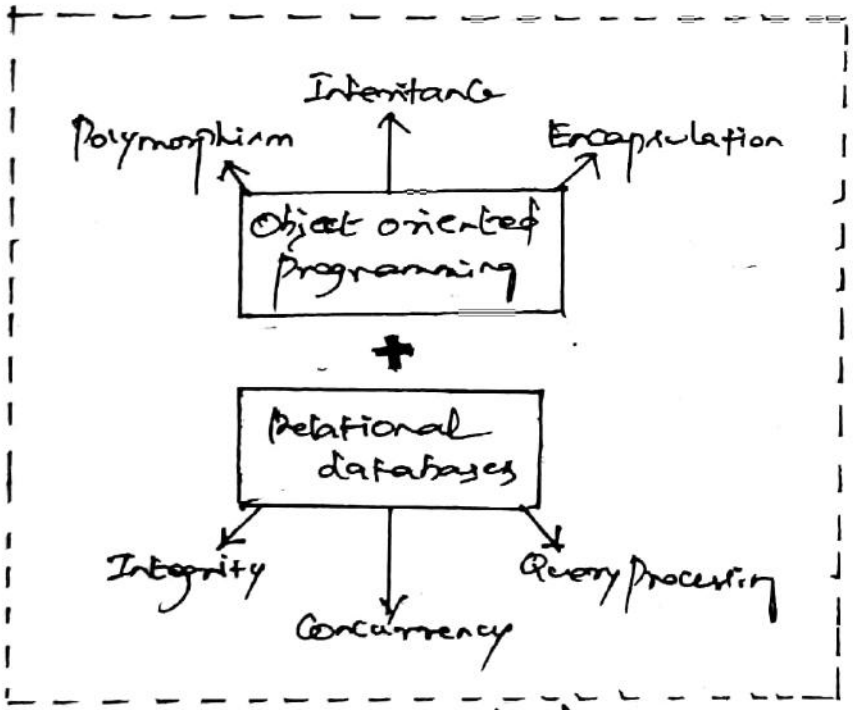
* Maintaining a log for recovery purposes
* Participating in an appropriate concurrency-control scheme to coordinate the concurrent execution of the transaction executing at that site.

The transaction coordinator is responsible for

* starting the execution of transaction
* Breaking the transaction into a number of sub-transactions and distributing these transactions to the appropriate sites for execution.
* Coordinating the termination of the transaction, which may result in the transaction being committed or aborted at all sites.

Introduction to Object oriented databases

* Object oriented database systems are alternative to relational database and other database systems.
* In object oriented database, information is represented in the form of objects.
* Object oriented databases are exactly same as object oriented programming. If we can combine the features of relational model (transaction, concurrency, recovery) to object oriented database, the resultant model is called as object oriented database model.



Object oriented database

Features of OODBMS

In OODBMS, every entity is considered as object and represented in a table. Similar objects

are classified into classes and subclasses and relationship between two objects is maintained. The features are:

1. Complexity

OODBMS has the ability to represent the complex internal structure (of object) with multilevel complexity.

2. Inheritance

Creating a new object from an existing object in such a way that new object inherits all characteristics of an existing object.

3. Encapsulation

It is a data hiding concept in OOP, which binds the data and functions together which can manipulate data and not visible to outside world.

4. Persistence

OODBMS allows to create persistent object (object that remains in memory even after execution). This feature can automatically solve the problem of recovery and concurrency.

Standards for Object-Oriented model

ODMG : Object Data Management Group

* Provide a standard where previous not available.
* Support portability between products.
* Standardize model, querying and programming issues.

Language of specifying the structure of Object database

ODL : Object Definition Language
OQL : Object Query Language

ODL

It is somehow similar to DDL (Data Definition Language in SQL

ODL : classes and attributes

```
class Person{
    String name;
    Date birthdate;
//Methods :
  Person();
  // Constructor : a new person is born
  int age();
  // Returning an atomic type
};
```

```
class Employee : Person{
  //A subclass of person
  float salary;
};
class Student : Person{
  //A subclass of person
  String grade;
};
class Address{
  int number;
  String street;
};
class Building{
  Address address;
  //A complex value embedded in this object
};
class Appartment{
  int number;
};
```

One-to-One relationship

An object refers to another object through a Ref. A Ref behaves as a C++ pointer, but with more semantics. Then, referential integrity can be expressed in the schema and maintained by the system. This is done by declaring the

relationship as system. For instance, we can say that a person lives in an Appartment and that Appartment is used by this person, in the following way:

```
class Person {
  Ref<Appartment> lives_in
  inverse is_used_by;
};
```

If a person moves to another Appartment, the attribute "is_used_by" is automatically reset to NULL until a new person takes this Appartment again. Moreover, if an Appartment object is deleted, the corresponding "lives_in" attribute is automatically reset to NULL.

Collections

The efficient management of very large collections of data is a fundamental database feature. Thus, it introduces a set of predefined generic classes for this purpose:

Set<T>	Set of unsorted unique objects
Bag<T>	a multi-set, allowing duplicates
Varray<T>	Variable size array
List<T>	Variable size and invertable array

A collection is a container of elements of the same class. As usual, polymorphism is obtained through the class hierarchy. For instance a Set<Ref<person>> may contain persons as well as Employees, if the class Employee is a subclass of the class person.

For instance, the following collections can be defined:

```
Set<Ref<person>> persons;
    // the person class extent (tuple (or) record)
Set<Ref<Apartment>> Apartments;
    // the Apartment class extent
Set<Ref<Apartment>> Vacancy;
    // the set of Vacant apartments
List<Ref<Apartment>> Directory;
    // the list of apartments ordered by their no. of rooms.
```

Multiple Relationships

An object is related with more than one object through a relationship.

For example, a Person has two parents and possibly several children; in a building there are many Apartments

```
class Person {
  Set<Ref<Person>> parents
    inverse children; // 2 parents
  List<Ref<Person>> children
    inverse parents; // ordered by birthday
};
class Building {
  List<Ref<Appartment>> appartments
    inverse building;
  // ordered by appartment number
};
class Appartment {
    int number;
    Ref<Building> building
    inverse appartments;
};
```

Object Query Language (OQL)

* OQL is a query language designed to operate on databases described in ODL.

* Tries to bring some concepts from the relational model to the ODBMS

  E.g., the SELECT statement, joins, aggregation, etc.

* References of class properties (attributes, relationships, and methods) using:

  Dot notation (p.a) (or)

  Arrow notation p→a

* In OQL, both notations are equivalent

Path Expressions

For instance, if we have a person "p" and we want to know the name of the street where this person lives, the OQL query is

```
p.lives_in.building.address.street
```

If we want the names of the children of the person p, then the query is

```
Select c.name from c in p.children;
```

The result of this query is a value of type Bag<String>. If we want to get a set, we simply drop duplicates, like in SQL by using "distinct" keyword.

```
Select distinct c.name from c in p.children;
```

If we want the set of addresses of the children of each person of the database, then

```
Select c.lives_in.building.address from p in
    persons, c in p.children;
```

Predicate

The "where" clause can be used to define any predicate which selects only the data matching the predicate.

For instance, if we want the set of people living on Main Street and having atleast two children, the query is:

```
Select c.lives_in.building.address from
  p in persons, c in p.Children where
  p.lives_in.building.address.street =
    "Main Street" and count(p.children)>=2;
```

Join

For instance, to get people living in a street and have the same name as this street, the query is

```
Select p from p in persons, b in (select
  distinct a.building from a in Appartments)
where p.name = b.address.street;
```

XML Database

XML Database is used to store huge amount of information in the XML format. As the use of XML is increasing in every field, it is required to have a secured place to store the XML documents. The data stored in the database can be queried using XQuery, serialized, and exported into a desired format.

XML Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

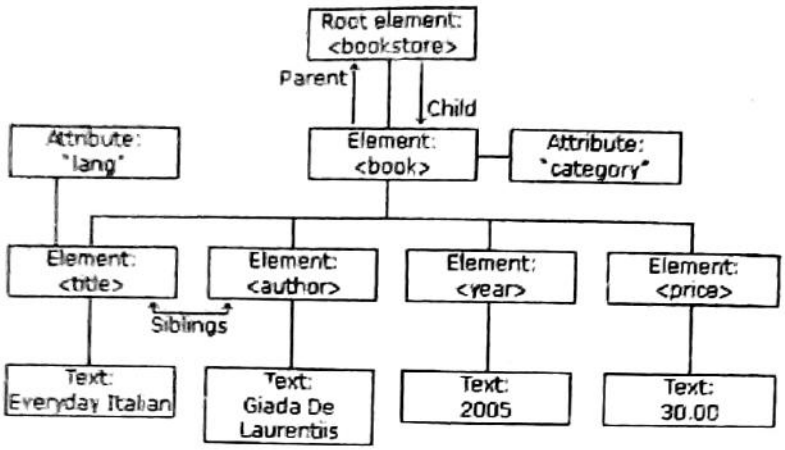
Books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>

  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>

  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>

  </bookstore>
```

XML Tree

XML documents form a tree structure that starts at "the root" and branches to "the leaves".

XML Hierarchial model



What is XQuery?

XQuery is to XML what SQL is to databases.
XQuery was designed to query XML data.

XQuery Example

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

XQuery Expressions

Let us understand each piece of the above XQuery expression.

Use of functions

```
doc("books.xml")
```

doc() is one of the XQuery functions that is used to locate the XML source. Here we've passed "books.xml". Considering the relative path, books.xml should lie in the same path where books.xqy is present.

Use of XPath expressions

```
doc("books.xml")/books/book
```

XQuery uses XPath expressions heavily to locate the required portion of XML on which search is to be made. Here we've chosen all the book nodes available under books node.

Iterate the objects

```
for $x in doc("books.xml")/bookstore/book
```

XQuery treats xml data as objects. In the above example, $x represents the selected node, while the for loop iterates over the collection of nodes.

Apply the condition

```
where $x/price>30
```

As $x represents the selected node, "/" is used to get the value of the required element; "where" clause is used to put a condition on the search results.

Return the result

```
return $x/title
```

As $x represents the selected node, "/" is used to get the value of the required element, price, title; "return" clause is used to return the elements from the search results.

XML DTD

An XML document with correct syntax is called "Well Formed".
An XML document validated against a DTD is both "Well Formed" and "Valid".

Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The DOCTYPE declaration, in the example above, is a reference to an external DTD file. The content of the file is shown in the paragraph below.

XML DTD

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

The DTD above is interpreted like this:

- !DOCTYPE note defines that the root element of the document is note
- !ELEMENT note defines that the note element must contain the elements: "to, from, heading, body"
- !ELEMENT to defines the to element to be of type "#PCDATA"
- !ELEMENT from defines the from element to be of type "#PCDATA"
- !ELEMENT heading defines the heading element to be of type "#PCDATA"
- !ELEMENT body defines the body element to be of type "#PCDATA"

#PCDATA means parse-able text data.

XML Schema

- An XML Schema describes the structure of an XML document, just like a DTD.
- An XML document with correct syntax is called "Well Formed".
- An XML document validated against an XML Schema is both "Well Formed" and "Valid".

XML Schema is an XML-based alternative to DTD:

```
<xs:element name="note">
<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

The Schema above is interpreted like this:

- <xs:element name="note"> defines the element called "note"
- <xs:complexType> the "note" element is a complex type
- <xs:sequence> the complex type is a sequence of elements
- <xs:element name="to" type="xs:string"> the element "to" is of type string (text)
- <xs:element name="from" type="xs:string"> the element "from" is of type string
- <xs:element name="heading" type="xs:string"> the element "heading" is of type string
- <xs:element name="body" type="xs:string"> the element "body" is of type string

XML Schemas are More Powerful than DTD

- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces

Information Retrieval is the discipline that deals with the structure, analysis, organisation, storage, searching and retrieval of information.

Information retrieval mainly deals with unstructured data, and the techniques for indexing, searching & retrieving information from large collection of unstructured documents.

2. average income of p...

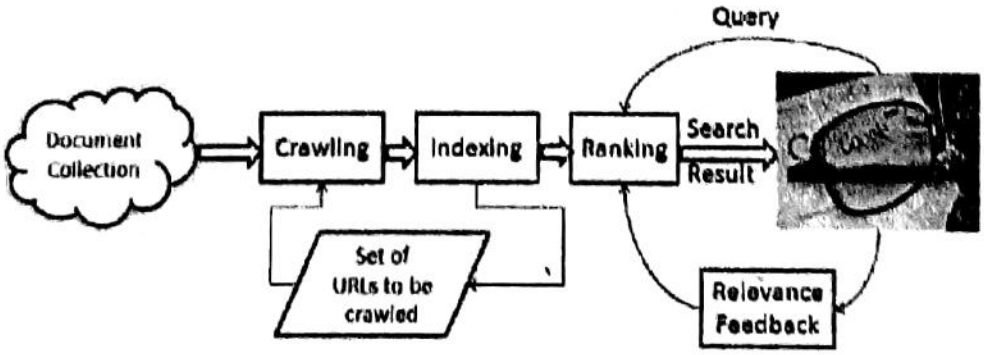
What is Information Retrieval?

Information Retrieval is the art of presentation, storage, organization of and access to information items. The representation and organization of information should be in such a way that the user can access information to meet his information need.

The definition of information retrieval is:

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Another feature of information retrieval is that it does not actually fetch documents. It only informs the user on the existence and whereabouts of documents relating to his query.



Components of an Information Retrieval System

In this section we describe the components of a basic web information retrieval system. A general information retrieval functions in the following steps.

1. The system browses the document collection and fetches documents. - Crawling
2. The system builds an index of the documents - Indexing
3. User gives the query
4. The system retrieves documents that are relevant to the query from the index and displays that to the user - Ranking
5. User may give relevance feedback to the search engine - Relevance Feedback.

The goal of any information retrieval system is to satisfy user's information need. Unfortunately, characterization of user information need is not simple. User's often do not know clearly about the information need. Query is only a vague and incomplete description of the information need. Query operations like query expansion, stop word removal etc. are usually done on the query.

Crawling

The web crawler automatically retrieves documents from the web as per some de ned strategy. The crawler creates a copy of all the documents it crawls to be processed by the search engine. The crawler starts from a list of URLs (documents) called seed. The crawler visits the URLs, identifies the outgoing hyperlinks there and adds them to the list of URLs (documents) to be visited. This way the crawler traverses the web graph following hyperlinks. It saves a copy of each document it visits.

Indexing

The documents crawled by the search engine are stored in an index for efficient retrieval. The documents are first parsed, and then tokenized, stop-word removed and stemmed. After thatthey are stored in an inverted index. The process is discussed below.

Tokenization

This system extracts word tokens (index terms) from running text. For example, given a piece of text: "Places to be visited in Delhi" it outputs [places, to, be, visited, in, Delhi].

Stop-word eliminator

Stop-words are those words that do not have any disambiguation power. Common examples ofstop words are articles, prepositions etc.

In this step, stop words are removed from the list of tokens. For example, given the list of token generated by tokenizer, it strips it down to: [places, visited, Delhi].

Stemmer

The remaining tokens are then stemmed to the root form (e.g. visited->visit). For example, after stemming the list of tokens becomes this: [place, visit, Delhi].

Inverted index

The ordinary index would contain for each document, the index terms within it. But the inverted index stores for each term the list of documents where they appear. The benefit of using an inverted index comes from the fact that in IR we are interested in finding the documents that contain the index terms in the query. So, if we have an inverted index, we do not have to scan through all the documents in collection in search of the term. Often a hash-table is associated with the inverted index so that searching happens in O(1) time.

Inverted Index may contain additional information like how many times the term appears in the document, the o set of the term within the document etc.

Example Say there are three documents.

Doc1 Milk is nutritious

Doc2 Bread and milk tastes good

Doc1 Brown bread is better

After stop-word elimination and stemming, the inverted index looks like:

milk	1,2
nutritious	1
bread	2, 3
taste	2
good	2
brown	3
better	3

Ranking

When the user gives a query, the index is consulted to get the documents most relevant to the query. The relevant documents are then ranked according to their degree of relevance, importance etc.

Relevance Feedback

Relevance feedback is one of the classical ways of refining search engine rankings. It works in the following way:

Search engine first generate an initial set of rankings. Users select the relevant documents within this ranking. Based on the information in these documents, a more appropriate ranking is presented (for example, the query may be expanded using the terms contained in the first set of relevant documents).

Sometimes users do not enough domain knowledge to form good queries. But they can select relevant documents from a list of documents once the documents are shown to him.

For example,

When the user fires a query 'matrix', initially documents on both the topics (movie and maths) are retrieved. Then say, the user selects the maths documents as relevant.

This feedback can be used to refine the search and retrieve more documents from mathematics domain.

Types of relevance feedback

Explicit - User gives feedback to help system to improve.

Implicit - User doesn't know he is helping

e.g. "similar pages" features in Google.

Pseudo - User doesn't do anything! Top 'k' judgments are taken as relevant. Being fully automated it has always this risk that results may drift completely away from the intended document set.

Types of Queries in IR Systems

Different keywords are associated with the document set during the process of indexing. These keywords generally consist of words, phrases, and other characterizations of documents such as date created, author names, and type of document. They are used by an IR system to build an inverted index, which is then consulted during the search. The queries formulated by users are compared to the set of index keywords. Most IR systems also allow the use of Boolean and other operators to build a complex query. The query language with these operators enriches the expressiveness of a user's information need.

1. Keyword Queries

Keyword-based queries are the simplest and most commonly used forms of IR queries: the user just enters keyword combinations to retrieve documents. The query keyword terms are implicitly connected by a logical AND operator. A query such as 'database concepts' retrieves documents that contain both the words 'data-base' and 'concepts' at the top of the retrieved results. In addition, most systems also retrieve documents that contain only 'database' or only 'concepts' in their text. Some systems remove most commonly occurring words (such as *a*, *the*, *of*, and so on, called **stopwords**) as a preprocessing step before sending the filtered query key-words to the IR engine. Most IR systems do not pay attention to the ordering of these words in the query. All retrieval models provide support for keyword queries.

2. Boolean Queries

Some IR systems allow using the AND, OR, NOT, ( ), + , and – Boolean operators in combinations of keyword formulations. AND requires that both terms be found. OR lets either term be found. NOT means any record containing the second term will be excluded. '( )' means the Boolean operators can be nested using parentheses. '+' is equivalent to AND, requiring the term; the '+' should be placed directly in front of the search term. '–' is equivalent to AND NOT and means to exclude the term; the '–' should be placed directly in front of the search term not wanted. Complex Boolean queries can be built out of these operators and their combinations, and they are evaluated according to the classical rules of Boolean algebra. No ranking is possible, because a document either satisfies such a query (is "relevant") or does not satisfy it (is "nonrelevant"). A document is retrieved for a Boolean query if the query is logically true as an exact match in the document. Users generally do not use combinations of these complex Boolean operators, and IR systems support a restricted version of these set operators. Boolean retrieval models can directly sup-port different Boolean operator implementations for these kinds of queries.

3. Phrase Queries

When documents are represented using an inverted keyword index for searching, the relative order of the terms in the document is lost. In order to perform exact phrase retrieval, these phrases should be encoded in the inverted index or implemented differently (with relative positions of word occurrences in documents). A phrase query consists of a sequence of words that makes up a phrase. The phrase is generally enclosed within double quotes. Each retrieved document must contain at least one instance of the exact phrase. Phrase searching is a more restricted and specific version of proximity searching that we mention below. For example, a phrase searching query could be 'conceptual database design'. If phrases are indexed by the retrieval model, any retrieval model can be used for these query types. A phrase thesaurus may also be used in semantic models for fast dictionary searching for phrases.

4. Proximity Queries

Proximity search refers to a search that accounts for how close within a record multiple terms should be to each other. The most commonly used proximity search option is a phrase search that requires terms to be in the exact order. Other proximity operators can specify how close terms should be to each other.

Some will also specify the order of the search terms. Each search engine can define proximity operators differently, and the search engines use various operator names such as NEAR, ADJ(adjacent), or AFTER. In some cases, a sequence of single words is given, together with a maximum allowed distance between them. Vector space models that also maintain information about positions and offsets of tokens (words) have robust implementations for this query type. However, providing support for complex proximity operators becomes computationally expensive because it requires the time-consuming preprocessing of documents, and is thus suitable for smaller document collections rather than for the Web.

5. Wildcard Queries

Wildcard searching is generally meant to support regular expressions and pattern matching-based searching in text. In IR systems, certain kinds of wildcard search support may be implemented—usually words with any trailing characters (for example, 'data*' would retrieve *data, database, datapoint, dataset*, and so on). Providing support for wildcard searches in IR systems involves preprocessing overhead and is not considered worth the cost by many Web search engines today. Retrieval models do not directly provide support for this query type.

6. Natural Language Queries

There are a few natural language search engines that aim to understand the structure and meaning of queries written in natural language text, generally as a question or narrative. This is an active area of research that employs techniques like shallow semantic parsing of text, or query reformulations based on natural language under-standing. The system tries to formulate answers for such queries from retrieved results. Some search systems are starting to provide natural language interfaces to provide answers to specific types of questions, such as definition and factoid questions, which ask for definitions of technical terms or common facts that can be retrieved from specialized databases. Such questions are usually easier to answer because there are strong linguistic patterns giving clues to specific types of sentences—for example, 'defined as' or 'refers to'. Semantic models can provide support for this query type.

Prepared by Verified by Approved by 26

UNIT I RELATIONAL DATABASES

Purpose of Database System – Views of data – Data Models – Database System Architecture – Introduction to relational databases – Relational Model – Keys – Relational Algebra – SQL fundamentals – Advanced SQL features – Embedded SQL– Dynamic SQL

UNIT-I/PART-A

1.	**Define database management system?** Database management system (DBMS) is a collection of interrelated data and a set of programs to access those data.
2.	**List any five applications of DBMS.** Banking, Airlines, Universities, Credit card transactions, Tele communication, Finance, Sales, Manufacturing, Human resources.
3.	**What is the purpose of Database Management System? *(Nov/Dec 14)*** Data redundancy and inconsistency , Difficulty in accessing data, Data isolation, Integrity problems, Atomicity problems and Concurrent access anomalies
4.	**Define instance and schema?** **Instance:** Collection of data stored in the data base at a particular moment is called an Instance of the database **Schema:** The overall design of the data base is called the data base schema.
5.	**Define the terms 1) physical schema 2) logical schema.** **Physical schema:** The physical schema describes the database design at the physical level, which is the lowest level of abstraction describing how the data are actually stored. **Logical schema:** The logical schema describes the database design at the logical level, which describes what data are stored in the database and what relationship exists among the data.
6.	**What is a data model? List the types of data models used?** A data model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.
7.	**Define- relational algebra.** The relational algebra is a procedural query language. It consists of a set of operations that take one or two relation as input and produce a new relation as output.
8.	**What is a data dictionary?** A data dictionary is a data structure which stores meta data about the structure of the database i.e. the schema of the database.
9.	**List out the operations of the relational algebra** The Six basic operators Select, project, union, set difference, Cartesian product and Rename.
10.	**Define relational data model** Relational model use a collection of tables to represent both data and the relationships among those data. Each table has a multiple columns and each column has unique name.
11.	**Explain Semi structured data model** ✓ Specification of data where individual data item of same type may have different sets of attributes ✓ Sometimes called schema less or self-describing ✓ XML is widely used to represent this data model
12.	**Define Object based data model** Object based data model can be seen as extension of the E-R model with notion of encapsulation, methods and object identify.
13.	**Explain Hierarchical data model** ✓ The Hierarchical data model organizes data in a tree structure. There is hierarchy of parent and child data segments. ✓ This model uses parent child relationship. ✓ 1:M Mapping between record type

14.	**Define Network Model** ✓ Some data were more naturally modeled with more than one parent per child. ✓ This model permitted the modeling of M:N relationship
15.	**Write the characteristics that distinguish the Database approach with the File-based approach.** ***(Apr/May 15)(Nov/Dec 16)*** **File-based System.** 1. Separation and isolation of data 2. Duplication of data 3. Incompatible file formats 4. Data dependence **Database Approach:** 1. Control of data redundancy 2. Data consistency 3. Sharing of data 4. Improved data integrity 5. Improved security
16.	**What are the disadvantages of file processing system?*(May/June 16)*** The file processing system has the following major disadvantages: ✓ Data redundancy and inconsistency ✓ Integrity Problems ✓ Security Problems ✓ Difficulty in accessing data ✓ Data isolation.
17.	**Define query language?** A query is a statement requesting the retrieval of information. The portion of DML that involves information retrieval is called a query language.
18.	**List the string operations supported by SQL?** 1) Pattern matching Operation 2) Concatenation 3) Extracting character strings 4) Converting between uppercase and lower case letters.
19.	**List out some date functions.** ✓ To_date ✓ To_char(sysdate,'fmt') ✓ d,dd,ddd,mon,dy,day,y,yy,yyy,yyyy,year,month,mm
20.	**What is the use of sub queries?** A sub query is a select-from-where expression that is nested with in another query. A common use of sub queries is to perform tests for set membership, make set comparisons, and determine set cardinality.
21.	**Name the categories of SQL command? *(May/June 16)*** SQL commands are divided in to the following categories: 1. Data - definition language 2. Data manipulation language 3. Data Query language 4. Data control language 5. Data administration statements 6. Transaction control statements
22.	**List the SQL domain Types?** SQL supports the following domain types. Char (n) , varchar (n), int , numeric (p,d) , float(n) , date.

23. What are aggregate functions? And list the aggregate functions supported by SQL?

Aggregate functions are functions that take a collection of values as input and return a single value. Aggregate functions supported by SQL are

- ✓ Average: avg
- ✓ Minimum: min
- ✓ Maximum: max
- ✓ Total: sum Count: count

24. What is the difference between char and varchar2 data type?

- ✓ Char and varchar2 are data types which are used to store character values.
- ✓ Char is static memory allocation; varchar2 is dynamic memory allocation.

25. How to add primary key to a table with suitable query?

Alter table <table name> add primary key(column);

26. Differentiate static and dynamic SQL. *(Nov/Dec 14,15,16) (Apr/May 15)*

Static SQL	Static SQL
The SQL statements do not change each time the program is run is called Static SQL.	The SQL statements do not change each time the program is run is called Static SQL.
Static SQL is compiled and optimized prior to its execution	Static SQL is compiled and optimized prior to its execution
The statement is prepared before the program is executed and the operational form of the statement persists beyond the execution of the program.	The statement is prepared before the program is executed and the operational form of the statement persists beyond the execution of the program.

27. Why does SQL allow duplicate tuples in a table or in a query result? *(Nov/Dec 15)*

If key constraint is not set on a relation every result in a relation will be considered as a tuple and hence SQL allows duplicate tuples in a table. Distinct keyword is used to avoid duplicate tuples in the result.

28. Define: DDL, DML, DCL and TCL. *(Nov/Dec 14,16)(Apr/May 15)*

DDL COMMANDS:

- Create
- Alter
  - ✓ Add
  - ✓ Modify
  - ✓ Drop
- Rename
- Drop

DML Commands:

- Insert
- Select
- Update
- Delete

DCL commands

- Grant - Provide access privilege to user
- Revoke - Get back access privilege from user

TCL commands

- Commit
- Rollback
- Save point

29. What is the use of Union and intersection operation?

Union: The result of this operation includes all tuples that are either in r1 or in r2 or in both r1 and r2.Duplicate tuples are automatically eliminated.

Intersection: The result of this relation includes all tuples that are in both r1 and r2.

30.	**What is embedded SQL? What are its advantages?** The SQL standard defines embedded of SQL in a variety of programming languages such as C, Java, and Cobol. A language to which SQL queries are embedded is referred to as a host language, and the SQL structures permitted in the host language comprise embedded SQL. The basic form of these languages follows that of the System R embedding of SQL into PL/I. EXEC SQL statement is used to identify embedded SQL request to the preprocessor EXEC SQL <embedded SQL statement > END_EXEC
	UNIT-I / PART-B
1.	Explain the purpose and components of DBMS in detail.
2.	List out the disadvantages of File system over DB & explain it in detail.
3.	List out the operations of the relational algebra and explain with suitable examples.*(Nov/Dec 16)*
4.	i) With the help of a neat block diagram explain the basic architecture of a database management system.*(Nov/Dec 15)(May/June 16)* ii) What are the advantages of having a centralized control of data? Illustrate your answer with suitable example. *(Nov/Dec 15)*
5.	Briefly explain about views of data.*(May/June 16)*
6.	Discuss about (i) Data Models (ii) Mapping cardinalities. *(Nov/Dec 14)*
7.	Explain about data definition language and data manipulation language in SQL with examples.*(Nov/Dec 14)(May/June 16)*
9.	Explain about data control language and TCL in SQL with examples.
10.	Design an employee detail relation and explain referential integrity using SQL queries.
11.	Consider a student registration database comprising of the below given table schema. Student File Student Number | Student Name | Address | Telephone Course File Course Number | Description | Hours | Professor Number Professor File Professor Number | Name | Office Registration file Student Number | Course Number | Date Consider a suitable sample of tuples/records for the above mentioned tables and write DML statements (SQL) to answer for the queries listed below. 1. Which courses does a specific professor teach? 2. What courses does specific professors? 3. Who teaches a specific course and where is his/her office? 4. For a specific student number, in which courses is the student registered and what is his/her name? 5. Who are the professors for a specific student? 6. Who is the student registered in a specific course? *(Apr/May 15)*
12	Explain about SQL Fundamentals.*(May/June 16)*
13	Describe the six clauses in the syntax of an SQL query, and show what type of constructs can be specified in each of the six clauses. Which of the six clauses are required and which are optional? *(Nov/Dec 15)*

14.	Assume the following table. Degree(degcode, name, subject) Candidate(seatno, degcode, semester, month, year, result) Marks(seatno, degcode, semester, month, year, papcode, marks) Degcode-degree code, Name-name of the degree (MSc, MCOM) Subject-subject of the course. E.g. Phy, Papcode- Paper code E.g. A1 Solve the following queries using SQL: (i) Write a SELECT statement to display all the degree codes which are there in the candidate table but not present in degree table in the order of degcode. (ii) Write a SELECT statement to display the name of all the candidates who have got less than 40 marks in exactly 2 subjects. (iii)Write a SELECT statement to display the name, subject and number of candidates for all degrees in which there are less than 5 candidates. (iv) Write a SELECT statement to display the names of all the candidates who have got highest total marks in MSc.,(Maths) ***(Nov/Dec 15)***

UNIT II DATABASE DESIGN

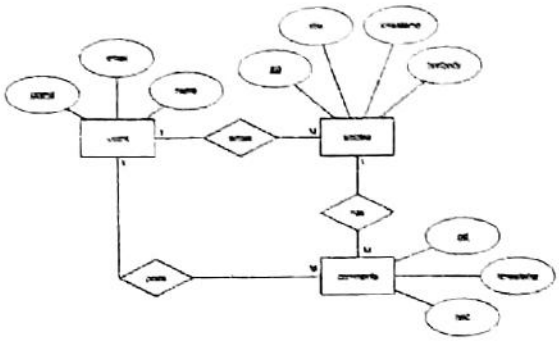
Entity-Relationship model – E-R Diagrams – Enhanced-ER Model – ER-to-Relational Mapping – Functional Dependencies – Non-loss Decomposition – First, Second, Third Normal Forms, Dependency Preservation – Boyce/Codd Normal Form – Multi-valued Dependencies and Fourth Normal Form – Join Dependencies and Fifth Normal Form

UNIT-II / PART-A

1.	**Explain entity relationship model?*(May/June 16)*** The entity relationship model is a collection of basic objects called entities and relationship among those objects. An entity is a thing or object in the real world that is distinguishable from other objects.
2.	**What is relationship? Give examples** A relationship is an association among several entities. Example: A depositor relationship associates a customer with each account that he/she has.
3.	**What are stored and derived attributes?** Stored attributes: The attributes stored in a data base are called stored attributes. Derived attributes: The attributes that are derived from the stored attributes are called derived attributes
4.	**What are composite attributes?** Composite attributes can be divided in to sub parts. The degree of relationship type is the number of participating entity types.
5.	**What is a weak entity? Give example. *(Nov/Dec 16)*** It is an entity that cannot be identified uniquely without considering some primary key attributes of another identifying owner entity. An example is including Dependent information for employees for insurance purposes.
6.	**What are attributes? Give examples.** An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set. Example: possible attributes of customer entity are customer name, customer id, Customer Street, customer city.
7.	**Mention the 2 forms of integrity constraints in ER model?** ✓ Key declarations ✓ Form of a relationship

8.	**What is the use of integrity constraints?** Integrity constraints ensure that changes made to the database by authorized users do not result in a loss of data consistency. Thus integrity constraints guard against accidental damage to the database
9.	**List some security violations (or) name any forms of malicious access.** 1) Unauthorized reading of data 2) Unauthorized modification of data 3) Unauthorized destruction of data.
10.	**What is a primary key?** Primary key is a set of one or more attributes that can uniquely identify record from the relation; it will not accept null values and redundant values. A relation can have only one primary key.
11.	**What is called query processing?** Query processing refers to the range of activities involved in extracting data from a database.
12.	**What is called a query evaluation plan?** A sequence of primitive operations that can be used to evaluate be query is a query evaluation plan or a query execution plan.
13.	**What is called as an N-way merge?** The merge operation is a generalization of the two-way merge used by the standard in-memory sort-merge algorithm. It merges N runs, so it is called an N-way merge.
14.	**What is a super key?** A super key is a set of one or more attributes that collectively allows us to identify uniquely an entity in the entity set.
15.	**What is foreign key?** A relation schema r1 derived from an ER schema may include among its attributes the primary key of another relation schema r2.this attribute is called a foreign key from r1 referencing r2.
16.	**What is the difference between unique and primary key?** Unique and primary key are keys which are used to uniquely identify record from the relation. But unique key accepts null values; primary key does not accept null values.
17.	**What does the cardinality ratio specify?** Mapping cardinalities or cardinality ratios express the number of entities to which another entity can be associated. Mapping cardinalities must be one of the following: One to one, One to many, Many to one and Many to many.
18.	**Explain the two types of participation constraint.** ✓ Total: The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R. ✓ Partial: if only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be partial.
19.	**Define Tuple variable?** Tuple variables are used for comparing two tuples in the same relation. The tuple variables are defined in the from clause by way of the as clause.
20.	**What is first normal form?** The domain of attribute must include only atomic (simple, indivisible) values.
21.	**What is 2NF?** Relation schema R is in 2NF if it is in 1NF and every non-prime attribute An in R is fully functionally dependent on primary key.
22.	**What is meant by domain key normal form?** Domain/key normal form (DKNF) is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.

23.	**Define Functional dependency.** ***(Apr/May 15)*** In a given relation R, X and Y are attributes. **Attribute Y is functionally dependent on attribute X if each value of X determines EXACTLY ONE value of Y**, which is represented as X → Y (X can be composite in nature). We say here "x determines y" *or "y is functionally dependent on x" Empid → Ename*
24.	**Define full functional dependency.** The removal of any attribute A from X means that the dependency does not hold any more.
25.	**Explain about partial functional dependency?** X and Y are attributes. Attribute Y is partially dependent on the attribute X only if it is dependent on a sub-set of attribute X.
26.	**What you meant by transitive functional dependency?** **Transitive dependency** is a functional dependency which holds by virtue of transitivity. A transitive dependency can occur only in a relation that has three or more attributes. Let A, B, and C designates three distinct attributes (or distinct collections of attributes) in the relation. Suppose all three of the following conditions hold: 1.A→ B 2. It is not the case that B→A 3. B→C Then the functional dependency A→C (Which follows from 1 and 3 by the axiom of transitivity) is a transitive dependency.
	UNIT-II / PART-B
1.	Explain ER model by taking Hospital management/Banking System/University Database as case study***(Nov/Dec 14)***
2.	Explain the various components of ER diagram with examples.
3.	Discuss about (i) Data Models (ii) Mapping cardinalities. ***(Nov/Dec 14)***
4.	Explain functional dependency in database design with its properties.
5.	Design an E-R diagram for keeping track of the exploits of your favourite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modelled as derived attributes.
6.	Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Each payment is for a particular period of time set of customers, and the date when the payment was received. ***(Nov/Dec 16)***
7.	A car rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number, license number, manufacturer, model, date of purchase, and color. Special data are included for certain types of vehicles. • Trucks: cargo capacity. • Sports cars: horsepower, renter age requirement. • Vans: number of passengers. • Off-road vehicles: ground clearance, drivetrain (four- or two-wheel drive). Construct an ER model for the car rental company database. ***(Nov/Dec 15)***

8.	State the need for Normalization of a Database and Explain the various Normal Forms (1st, 2nd, 3rd, BCNF, 4th, 5th and Domain- Key) with suitable examples. ***(Apr/May 15)(Nov/Dec 14,16)***
9.	Draw E – R Diagram for the "Restaurant Menu Ordering System", which will facilitate the food items ordering and services within a restaurant. The entire restaurant scenario is detailed as follows. The Customer is able to view the food items menu, call the waiter, place orders and obtain the final bill through the computer kept in their table. The waiters through their wireless tablet PC are able to initialize a table for customers, control the table functions to assist customers, orders, send orders to food preparation staff (chef) and finalize the customer's bill. The food preparation staffs (Chefs), with their touch-display interface to the system, are able to view orders sent to the kitchen by waiters. During preparation, they are able to let the waiter know the status of each item, and can send notification when items are completed. The system should have full accountability and logging facilities, and should support supervisor actions to account for exceptional circumstances, such as a meal being refunded or walked out on. ***(Apr/May 15)***
10.	For the ER diagram given below explain ER to Relational mapping procedures. 

UNIT III TRANSACTIONS

Transaction Concepts – ACID Properties – Schedules – Serializability – Concurrency Control – Need for Concurrency – Locking Protocols – Two Phase Locking – Deadlock – Transaction Recovery – Save Points – Isolation Levels – SQL Facilities for Concurrency and Recovery.

UNIT-III/ PART-A

1.	**Give the reasons for allowing concurrency?** The reasons for allowing concurrency is if the transactions run serially, a short transaction may have to wait for a preceding long transaction to complete, which can lead to unpredictable delays in running a transaction. So concurrent execution reduces the unpredictable delays in running transactions.
2.	**What is average response time?** The average response time is that the average time for a transaction to be completed after it has been submitted.
3.	**What are the two types of serializability?** The two types of serializability is Conflict serializability, View serializability.
4.	**Differentiate strict two phase locking protocol and rigorous two phase locking protocol.*(May/June 16)*** ✓ In strict two phases locking protocol all exclusive mode locks taken by a transaction is held until that transaction commits. ✓ Rigorous two phase locking protocol requires that all locks be held until the Transaction commits.

5.	**How the time stamps are implemented** ✓ Use the value of the system clock as the time stamp. That is a transaction's time stamp is equal to the value of the clock when the transaction enters the system. ✓ Use a logical counter that is incremented after a new timestamp has been assigned; that is the time stamp is equal to the value of the counter.
6.	**What are the different modes of lock?** The modes of lock are: ✓ Shared ✓ Exclusive
7.	**What are the time stamps associated with each data item?** ✓ W-timestamp (Q) denotes the largest time stamp if any transaction that executed WRITE (Q) successfully. ✓ R-timestamp (Q) denotes the largest time stamp if any transaction that executed READ (Q) successfully.
8.	**Define blocks?** The database system resides permanently on nonvolatile storage, and is partitioned into fixed-length storage units called blocks.
9.	**Define deadlock?** Neither of the transaction can ever proceed with its normal execution. This situation is called deadlock
10.	**Define the phases of two phase locking protocol** Growing phase: a transaction may obtain locks but not release any lock. Shrinking phase: a transaction may release locks but may not obtain any new locks.
11.	**Define upgrade and downgrade?** It provides a mechanism for conversion from shared lock to exclusive lock is known as upgrade. It provides a mechanism for conversion from exclusive lock to shared lock is known as downgrade.
12.	**What is a database graph?** The partial ordering implies that the set D may now be viewed as a directed acyclic graph, called a database graph.
13.	**What are uncommitted modifications?** The immediate-modification technique allows database modifications to be output to the database while the transaction is still in the active state. Data modifications written by active transactions are called uncommitted modifications.
14.	**What is meant by buffer blocks?** The blocks residing temporarily in main memory are referred to as buffer blocks.
15.	**Define shadow paging.** An alternative to log-based crash recovery technique is shadow paging. This technique needs fewer disk accesses than do the log-based methods.
16.	**Define page.** The database is partitioned into some number of fixed-length blocks, which are referred to as pages.
17.	**Explain current page table and shadow page table.** The key idea behind the shadow paging technique is to maintain two page tables during the life of the transaction: the current page table and the shadow page table. Both the page tables are identical when the transaction starts. The current page table may be changed when a transaction performs a write operation.

18.	**What is transaction?** Collections of operations that form a single logical unit of work are called transactions.
19.	**What are the drawbacks of shadow-paging technique?** ✓ Commit Overhead ✓ Data fragmentation ✓ Garbage collection
20.	**What is meant by garbage collection.*(May/June 16)*** Garbage may be created also as a side effect of crashes. Periodically, it is necessary to find all the garbage pages and to add them to the list of free pages. This process is called garbage collection.
21.	**What are the properties of transaction? Or Write the ACID properties of Transaction. *(Nov/Dec 14) (Apr/May 15)(May/June 16)*** Atomicity , Consistency, Isolation and Durability
22.	**What is recovery management component?** Ensuring durability is the responsibility of a software component of the base system called the recovery management component.
23.	**When is a transaction rolled back?** Any changes that the aborted transaction made to the database must be undone. Once the changes caused by an aborted transaction have been undone, then the transaction has been rolled back.
24.	**Give an example of two phase commit protocol. *(Nov/Dec 15)*** Client want all or nothing transactions and Transfer either happens or nothing at all.
25.	**What are the states of transaction?** The states of transaction are ✓ Active ✓ Partially committed ✓ Failed ✓ Aborted ✓ Committed ✓ Terminated
26.	**What is a shadow copy scheme?** It is simple, but efficient, scheme called the shadow copy schemes. It is based on making copies of the database called shadow copies that one transaction is active at a time. The scheme also assumes that the database is simply a file on disk.
27.	**What is serializability? How it is tested? *(Nov/Dec 14,16)*** A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule. Precedence graph is used to test the serializability
28.	**Mention the approaches of deadlock recovery** The common solution is to roll back one or more transactions to break the deadlock ✓ Selection of victim ✓ Rollback ✓ Partial ✓ Total and Starvation.

29.	**What is meant by concurrency control? *(Nov/Dec 15)*** A transaction is a particular execution of the program. When multiple transactions are trying to access the same shareable resource, many problems arise if the access control is not done properly. Mechanisms to which access control can be maintained is called Concurrency control.
30.	**List the four conditions for deadlock. *(Nov/Dec 16)*** ✓ mutual exclusion ✓ hold and wait or partial allocation ✓ no pre-emption ✓ resource waiting or circular wait
31.	**What are different isolation levels in database?** ✓ Serializable. ✓ Repeatable reads. ✓ Read committed. ✓ Read uncommitted. ✓ Dirty reads. ✓ Non-repeatable reads. ✓ Phantom reads. ✓ Isolation levels vs read phenomena.
32.	**What is isolation in database?** Transaction isolation is an important part of any transactional system. It deals with consistency and completeness of data retrieved by queries unaffecting a user data by other user actions. A database acquires locks on data to maintain a high level of isolation.
	UNIT-III / PART-B
1.	Discuss view serializability and conflict serializability *(Nov/Dec 15)*
2.	Write short notes on Transaction State and discuss the properties of transaction.
3.	Briefly describe two phase locking in concurrency control techniques. *(Nov/Dec 14,16)*
4.	Explain the concepts of concurrent execution in Transaction processing system.*(Nov/Dec 14)*
5.	Explain Transaction concept with an example. *(Nov/Dec 14)*
6.	Explain about dead lock recovery algorithm with an example.
7.	Illustrate Granularity locking method in concurrency control.
8.	Describe Database Recovery concepts.
9.	What is concurrency control? How is it implemented in DBMS? Illustrate with a suitable example. *(Nov/Dec 14)*
10.	Briefly explain about Two phase commit and three phase commit protocols. *(Apr/May 15) (May/June 16) (Nov/Dec 14)*
11.	What is deadlock? How does it occur? How transactions be written to (i) Avoid deadlock (ii) guarantee correct execution. Illustrate with suitable example. *(Nov/Dec 14,15,16)*

12.	Explain about Locking Protocols. ***(May/June 16)***
13.	Consider the following extension to the tree-locking protocol, which allows both shared and exclusive locks: • A transaction can be either a read-only transaction, in which case it can request only shared locks, or an update transaction, in which case it can request only exclusive locks. • Each transaction must follow the rules of the tree protocol. Read-only transactions may lock any data item first, whereas update transactions must lock the root first. Show that the protocol ensures serializability and deadlock freedom. ***(Nov/Dec 16)***
14.	Consider the following schedules. The actions are listed in the order they are schedule, and prefixed with transaction name. S1: T1: R(X), T2: R(x), T1: W(Y), T2: W(Y), T1: R(Y), T2: R(Y) S2:T3: R(X), T1: R(X), T1: W(Y), T2: R (Z), T2: W (Z), T3: R (Z) For each of the schedules, answer the following questions: i. What is the precedence graph for the schedule? ii. Is the schedule conflict-serializable? If so, what are all the conflict equivalent serial schedules? iii. Is the schedule view-serializable? If so, what are all the view equivalent serial schedules? ***(Apr/May 15)***

15.	Consider the following two transactions:
	T_1: read(A);
	read(B);
	if $A = 0$ **then** $B := B + 1$;
	write(B).
	T_2: read(B);
	read(A);
	if $B = 0$ **then** $A := A + 1$;
	write(A).
	Add lock and unlock instructions to transactions T31 and T32, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock? ***(Nov/Dec 16)***

UNIT IV IMPLEMENTATION TECHNIQUES

RAID – File Organization – Organization of Records in Files – Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing – Query Processing Overview – Algorithms for SELECT and JOIN operations – Query optimization using Heuristics and Cost Estimation.

UNIT-IV / PART-A

1.	**What is B-Tree?** • A B-tree eliminates the redundant storage of search-key values. • It allows search key values to appear only once.
2.	**What is a B+-Tree index?** A B+-Tree index takes the form of a balanced tree in which every path from the root of the root of the root of the tree to a leaf of the tree is of the same length.

3.	**What is a hash index?** A hash index organizes the search keys, with their associated pointers, into a hash file structure
4.	**Define seek time.** The time for repositioning the arm is called the seek time and it increases with the distance that the arm is called the seek time.
5.	**Define rotational latency time.** The time spent waiting for the sector to be accessed to appear under the head is called the rotational latency time.
6.	**What is called mirroring?** The simplest approach to introducing redundancy is to duplicate every disk. This technique is called mirroring or shadowing.
7.	**What are the two main goals of parallelism?** • Load -balance multiple small accesses, so that the throughput of such accesses increases. • Parallelize large accesses so that the response time of large accesses is reduced.
8.	**What is an index?** An index is a structure that helps to locate desired records of a relation quickly, without examining all records
9.	**What are the factors to be taken into account when choosing a RAID level?** • Monetary cost of extra disk storage requirements. • Performance requirements in terms of number of I/O operations • Performance when a disk has failed and Performances during rebuild.
10.	**What are the types of storage devices?** Primary storage, Secondary storage, Tertiary storage, Volatile storage, Nonvolatile storage
11.	**What is called remapping of bad sectors?** If the controller detects that a sector is damaged when the disk is initially formatted, or when an attempt is made to write the sector, it can logically map the sector to a different physical location.
12.	**Define software and hardware RAID systems?*(May/June 16)*** RAID can be implemented with no change at the hardware level, using only software modification. Such RAID implementations are called software RAID systems and the systems with special hardware support are called hardware RAID systems.
13.	**Define hot swapping?** Hot swapping permits the removal of faulty disks and replaces it by new ones without turning power off. Hot swapping reduces the mean time to repair.
14.	**What are the ways in which the variable-length records arise in database systems?** Storage of multiple record types in a file, Record types that allow variable lengths for one or more fields, Record types that allow repeating fields.
15.	**What are the two types of blocks in the fixed -length representation? Define them.** ✓ Anchor block: Contains the first record of a chain. ✓ Overflow block: Contains the records other than those that are the first record of a chain.

16. **What is hashing file organization?**
In the hashing file organization, a hash function is computed on some attribute of each record. The result of the hash function specifies in which block of the file the record should be placed.

17. **What are called index-sequential files?**
The files that are ordered sequentially with a primary index on the search key are called index-sequential files.

18. **Define Primary Index and Secondary Index**
It is in a sequentially ordered file, the index whose search key specifies the sequential order of the file. Also called clustering index. The search key of a primary index is usually but not necessarily the primary key. It is an index whose search key specifies an order different from the sequential order of the file. Also called non clustering index.

19. **Give an example of a join that is not a simple equi-join for which partitioned parallelism can be used.** *(Nov/Dec 15)*
$r\ join\ (r.A = s.B) \wedge (r.A < s.c)$

20. **Differentiate static and dynamic hashing.** *(Apr/May 15) (Nov/Dec 14,15)*

Static Hashing	Dynamic Hashing
In static hashing, when a search-key value is provided, the hash function always computes the same address.	Hash function, in dynamic hashing, is made to produce a large number of values and only a few are used initially.
The number of buckets provided remains unchanged at all times i.e. fixed	Dynamic hashing provides a mechanism in which data buckets are added and removed dynamically and on-demand .i.e. no. of buckets not fixed.
Space and overhead is more	Minimum space and less overhead
As file grows performance decreases	Performance do not degrade as file grows

21. **List out the mechanisms to avoid collision during hashing.** *(Nov/Dec 16)*
- ✓ In overflow chaining, the overflow buckets of a given bucket are chained together in a linked list.
- ✓ Above scheme is called closed hashing. An alternative, called open hashing, which does not use overflow buckets, is not suitable for database applications.

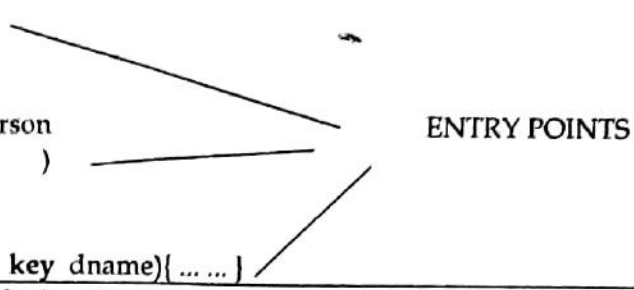
22. **What are the disadvantages of B-Tree over B+ Tree?** *(Nov/Dec 16)*
- ✓ Only small fraction of all search-key values are found early
- ✓ Non-leaf nodes are larger. Thus, B-Trees typically have greater depth than corresponding B^+-Tree
- ✓ Insertion and deletion more complicated than in B^+-Trees
- ✓ Implementation is harder than B^+-Trees.
- ✓ Not possible to sequentially scan a table by just looking at leafs.

23. **What is called query processing?**
Query processing refers to the range of activities involved in extracting data from a database.

24. **What is called a query evaluation plan?**
A sequence of primitive operations that can be used to evaluate be query is a query evaluation plan or a query execution plan.

25.	**Explain "Query optimization"?(May/June 16)** Query optimization refers to the process of finding the lowest cost method of evaluating a given query.
26.	**State the need for Query Optimization. (Apr/May 15)** The query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans.
	UNIT-IV / PART-B
1.	Describe File Organization.
2.	Define RAID and Briefly Explain RAID techniques.***(Nov/Dec 14, 15, 16) (Apr/May 15,16)***
3.	Explain Secondary storage devices.
4.	Explain about static and dynamic hashing with an example
5.	Explain about Multidimensional and parallel with an example
6.	Explain about ordered indices with an example
7.	Explain about B+ trees indexing concepts with an example ***(Nov/Dec 14)(May/June 16)***
8.	Explain about B trees indexing concepts with an example ***(Nov/Dec 14)***
9.	Illustrate indexing and hashing techniques with suitable examples. ***(Nov/Dec 15)***
10.	Explain about Query optimization with neat Diagram. ***(Nov/Dec 14,16)***
11.	Give a detailed description about Query processing and Optimization.Explain the cost estimation of Query Optimization ***(Nov/Dec 14).***
12.	Discuss about join order optimization and heuristic optimization algorithm. ***(Apr/May 15)***
13.	Briefly explain about Query Processing***(May/June 16)***
	UNIT V ADVANCED TOPICS Distributed Databases: Architecture, Data Storage, Transaction Processing – Object-based Databases: Object Database Concepts, Object-Relational features, ODMG Object Model, ODL, OQL – XML Databases: XML Hierarchical Model, DTD, XML Schema, XQuery – Information Retrieval: IR Concepts, Retrieval Models, Queries in IR systems.
	UNIT-V / PART-A
1.	**What is homogeneous distributed database and heterogeneous distributed database** A homogeneous distributed database has identical software and hardware running all databases instances, and may appear through a single interface as if it were a single database. A heterogeneous distributed database may have different hardware, operating systems, database management systems, and even data models for different databases.
2.	**Define Distributed Database Systems. *(Nov/Dec 16)*** Database spread over multiple machines (also referred to as sites or nodes).Network interconnects the machines. Database shared by users on multiple machines is called Distributed Database Systems
3.	**What are the types of Distributed Database** ✓ Homogenous distributed DB ✓ Heterogeneous distributed DB
4.	**Define fragmentation in Distributed Database** The system partitions the relation into several fragment and stores each fragment at different sites Two approaches : ✓ Horizontal fragmentation ✓ Vertical fragmentation

5.	**Define Database replication.** Database replication can be used on many database management systems, usually with a master/slave relationship between the original and the copies. The master logs the updates, which then ripple through to the slaves. The slave outputs a message stating that it has received the update successfully, thus allowing the sending of subsequent updates.
6.	**What is the advantage of OODB?** An integrated repository of information that is shared by multiple users, multiple products, multiple applications on multiple platforms.
7.	**What is Object database System?** An object database is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object-relational databases are a hybrid of both approaches.
8.	**What are the advantages of OODB?** An integrated repository of information that is shared by multiple users, multiple products, multiple applications on multiple platforms. It also solves the following problems: 1. The semantic gap: The real world and the Conceptual model is very similar. 2. Impedance mismatch: Programming languages and database systems must be interfaced to solve application problems. But the language style, data structures, of a programming language (such as C) and the DBMS (such as Oracle) are different. The OODB supports general purpose programming in the OODB framework. 3. New application requirements: Especially in OA, CAD, CAM, CASE, object-orientation is the most natural and most convenient.
9.	**How do you define types in object relational feature in oracle?** Oracle allows us to define types similar to the types of SQL. The syntax is CREATE TYPE t AS OBJECT (list of attributes and methods);
10.	**Define ODMG Object model?** The ODMG object model is the data model upon which the object definition langauge (ODL) and object query language (OQL) are based.
11.	**Define ODL.** ODL langauge is used to create object specifications: • classes and interfaces - Using the specific langauge bindings to specify how ODL • constructs can be mapped to constructs in specific programming • language, such as C++, SMALLTALK, and JAVA.
12.	**Define Information Retrieval.** It is an activity of obtaining information resources relevant to an information need from a collection of information resources.
13.	**Define Relevance Ranking.** ***(Nov/Dec 14)*** A system in which the search engine tries to determine the theme of a site that a link is coming from.
14.	**Can we have more than one constructor in a class? If yes, explain the need for such a situation.** ***(Nov/Dec 15)*** Yes, default constructor and constructor with parameter
15.	**Define XML Database.** An XML database is a data persistence software system that allows data to be stored in XML format. These data can then be queried, exported and serialized into the desired format. XML databases are usually associated with document-oriented databases.

16.	**Define OQL with syntax.** • Entry point to the database: needed for each query which can • be any named *persistent object*: ▪ the name of the extent of a class **class** Person (**extent** persons **key** ssn) {... ... } **class** Faculty **extends** Person (**extent** faculy) { ... ... } **class** Department (**extent** departmet **key** dname){ ... ... } 
17.	**Define Crawling and indexing the web.** ***(Nov/Dec 14)*** Web Crawling is the process of search engines combing through web pages in order to properly index them. These "web crawlers" systematically crawl pages and look at the keywords contained on the page, the kind of content, all the links on the page, and then returns that information to the search engine's server for indexing. Then they follow all the hyperlinks on the website to get to other websites. When a search engine user enters a query, the search engine will go to its index and return the most relevant search results based on the keywords in the search term. Web crawling is an automated process and provides quick, up to date data.
18.	**How does the concept of an object in the object-oriented model differ from the concept of an entity in the entity-relationship model?*(Nov/Dec 16)*** An entity is simply a collection of variables or data items. An object is an encapsulation of data as well as the methods (code) to operate on the data. The data members of an object are directly visible only to its methods. The outside world can gain access to the object's data only by passing pre-defined messages to it and these messages are implemented by the methods.
19.	**Is XML Hierarchical?** XML documents have a hierarchical structure and can conceptually be interpreted as a tree structure, called an XML tree. XML documents must contain a root element (one that is the parent of all other elements). All elements in an XML document can contain sub elements, text and attributes.
20.	**What is DTD?** A document type definition (DTD) contains a set of rules that can be used to validate an XML file. After you have created a DTD, you can edit it manually, adding declarations that define elements, attributes, entities, and notations, and how they can be used for any XML files that reference the DTD file.
21.	**What is the use of XML Schema?** XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces.
22.	**What is Xpath and Xquery?** XPath can be used to navigate through elements and attributes in an XML document. XPath is a syntax for defining parts of an XML document. XPath uses path expressions to navigate in XML documents. XPath contains a library of standard functions. XPath is a major element in XSLT and in XQuery.
23.	**Define Keyword Queries.** Keyword-based queries are the simplest and most commonly used forms of IR queries: the user just enters keyword combinations to retrieve documents.

24.	**What are the Types of Queries in IR Systems** • Keyword Queries, Boolean Queries • Phrase Queries • Proximity Queries • Wildcard Queries • Natural Language Queries.
25.	**State the steps to create DTD.** Create a new DTD, complete the following steps: 1. Create a project to contain the DTD if needed. 2. In the workbench, click File > New > Other and select XML > DTD. Click Next. 3. Select the project or folder that will contain the DTD. 4. In the File name field, type the name of the DTD, for example MyDTD.dtd. The name of your DTD file must end with the extension .dtd 5. Click Next. 6. Optional: You can use a DTD template as the basis for your new DTD file. To do so, click the Use DTD Template check box, and select the template you want to use. 7. Click Finish.

UNIT-V / PART-B

1.	Explain about Object Oriented Databases and XML Databases.
2.	Explain in detail (i) Information Retrieval (iii) Transaction processing ***(Nov/Dec 14)***
3.	Write short notes on Distributed Transactions. ***(Nov/Dec 14)***
4.	Explain in detail the Client - Server Architecture for DDBMS
5.	Suppose an Object Oriented database had an object A, which references object B, which in turn references object C. Assume all objects are on disk initially? Suppose a program first dereferences A, then dereferences B by following the reference from A, and then finally dereferences C. Show the objects that are represented in memory after each dereference, along with their state. ***(Nov/Dec 15)***
6.	Suppose that you have been hired as a consultant to choose a database system for your client's application. For each of the following applications, state what type of database system (relational, persistent programming language-based OODB, object relational; do not specify a commercial product) you would recommend. Justify your recommendation. (i)A computer-aided design system for a manufacturer of airplanes. (ii)A system to track contributions made to candidates for public office. (iii)An information system to support the making of movies. ***(Nov/Dec 16)***
7.	Give the DTD for an XML representation of the following nested-relational schema *Emp = (ename, ChildrenSet* setof*(Children), SkillsSet* setof*(Skills))* *Children = (name, Birthday)* *Birthday = (day, month, year)* *Skills = (type, ExamsSet* setof*(Exams)).* *Exams = (year, city)* ***(Nov/Dec 16)***
8.	Explain XML Schema with an example.
9.	Explain various queries in IR Systems with an example.
10.	Explain ODL and OQL with an example.
11.	Explain ODMG - Object Model in detail.

SCENARIO BASED SQL QUESTIONS

1. Consider a student registration database comprising of the below given table schema.
Student_File(student_number, student_name, address, telephone)
Course_File(course_number, description, hours, professor_number)
Professor_File(Professor_number, name, office)
Registration_File(student_number, course_number, date) **A/M 15** **(16)**

Consider a suitable sample of tuples/records for the above mentioned tables and write DML statements (SQL) to answer for each queries listed below

Student File

Student Number	Student Name	Address	Telephone

Course File

Course Number	Description	Hours	Professor Number

Professor File

Professor Number	Name	Office

Registration file

Student Number	Course Number	Date

(i) Which courses does a specific professor teach?
selectcoursenumber,description,professor_name from coursefile,professorfile where coursefile.professornumber=professorfile.professornumber

(ii) What courses are taught by two specific professors?
select description from course where professor number=102 and professor number=103;

(iii) Who teaches a specific course and where is his/her office?
selectcoursenumber,description,professor_name,office from coursefile,professorfile where coursefile.professornumber=professorfile.professornumber

(iv) For a specific student number, in which courses is the student registered and what is his/her name?
selectstudentname,coursename from studentfile.coursefile,registrationfile where studentfile.studentnumber=registration.studentnumber and coursefile.coursenumber=regitrationfile.coursenumber;

(v) Who are the professors for a specific student?
selectstudentname,professorname from studentfile,coursefile,professorfile,registrationfile where studentfile.studentnumber=registration.studentnumber andcoursefile.coursenumber=regitrationfile.coursenumberand coursefile.professornumber=professorfile.professornumber

(vi) Who is the student registered in a specific course?
selectstudentname,coursename from studentfile.coursefile,registrationfile wherestudentfile.studentnumber=registration.studentnumber and coursefile.coursenumber=regitrationfile.coursenumber;

2. Assume the following table:
Degree (degcode, name, subject)

candidate (seatno, degcode, name, semester, month, year, result)
Marks (seatno, degcode,semester, month, year, papcode, marks)
Degcode-degree, Name -name of the degree (MSc. MCOM)
Subject — subject of the courses Eg.Phy, Pap code — paper code eg. A1. N/D 15 (16)
Solve the following queries using SQL:

(i)Write a SELECT statement to display all the degree codes which are there in the candidate table but not present in degree table in the order of degcode. (4)
Select degcode from candidates where degcode not in(select degcode from degree order by degcode;

(Or)

Select c.degcode from candidate c,degree d where c.degcode<>d.degcode order by degcode;

(ii) Write a SELECT statement to display the name of all the dates who have got less than 40 marks in exactly 2 subjects (4)
Select name from candidate where degcode in (select degcode from marks group by degcode having count(marks<40)=2);

(iii) Write SELECT statement to display the name, subject and number of candidates for all degrees in which there are less than 5 candidates. (4)
Select name,subject,count(name) from degree where degcode in (select degcode from candidate where count(seatno)<5)

(iv)Write a SELECT statement to display the names of all the candidates who have got highest total marks in MSc., (Maths).(4)
Select name from candidate where degcodein(select degcode from marks where marks=(select max(marks) from marks) and papcode=(select papcode from subject where subject=''maths');

3.Consider the following relational schema:
Employee(empno, office, age)
Books(isbn, title, authors, publisher)
Loan(empno, isbn, date)
Write the following queries in relational algebra: N/D 18 (16)
i) Find the names of the employees who have borrowed a book published by XYZ limited. (4)
ii) Find the names of the employees who have borrowed all books published by XYZ limited. (4)
iii) Find the names of the employees who have borrowed more than 5 different books published by XYZ limited. (4)
iv) For each publisher, find the names of the employees who have borrowed more than 5 different books of that publisher. (4)

a. select name from employee e, books b, loan l where e.empno = l.empno and l.isbn = b.isbn and b.publisher = 'XYZ limited'

b. select name from employee e join loan l on e.empno=l.empno join (select isbn from books where publisher = 'XYZ limited') x on l.isbn=x.isbngroup by e.empno,namehaving count(*)=(select count(*) from books where publisher='XYZ limited')

c. select name from employee,loan where employee.empno=loan.empno and isbn in (select distinct isbn from books where publisher=XYZ limited') group by employee.empno,name having count(isbn) >=5

d. select name from employee,loan,books where employee.empno=loan.empno and books.isbn=loan.isbn group by employee.empno,name,books.publisher having count(loan.isbn) >=5

i. π name (employee ⋈ (πempno (Loan ⋈ (π isbn(б publisher=' XYZ limited' (books)))))

ii. π name, isbn(employee ⋈ πempno (loan ⋈books) ÷ π isbn(б publisher=' XYZ limited' (books))

iii. π name (employee ⋈ πempno loan ⋈books (π isbn(б publisher=' XYZ limited' ^count(isbn)>5(books))

iv. π name (employee ⋈ πempno loan ⋈books (π isbn(бpublisher=' XYZ limited' ^count(isbn)>5(books))

4. Consider the database schema
Emp(emp-name, type, birthday, set of(exam-names), set of(skills))
Children(Emp-name, Ch-name, birthday)
Skills(type, set of(Exam-names))
Exams(Exam-name, year, city)
Write SQL statements for the following queries. N/D 13 (16)

(i) Find the names of all employees who have birthday in March as their children. (2)
selectenamefrom emp as e, e.ChildrenSet as cwhere 'March' in
(select birthday. Monthfrom c)

(ii) Find those employees who took an examination for the skill type "typing" in the city "Chennai". (2)
select e.enamefrom emp as e, e.SkillSet as s, s.ExamSet as xwhere s.type = 'typing' and x.city = 'Chennai'

(iii) List all exam names under specific skill type for the given employee other than his exam names. (2)
select distinct s.typefrom emp as e, e.SkillSet as s

PART B

UNIT 1

1. Draw an ER diagram for the "Restaurant Menu Ordering System", which will facilitate the food items ordering and services within a restaurant. The entire restaurant scenario is detailed as follows. The customer is able to view the food items menu, call the waiter, place orders and obtain the final bill through the computer kept in their table. The waiters through their wireless tablet PC are able to initialize a table for customers, control the table functions to assist customers, orders, send orders to food preparation staff (chef) and finalize the customer's bill. The food preparation staffs (chefs), with their touch-display interfaces to the system, are able to view orders sent to the kitchen by waiters. During preparation, they are able to let the waiter know the status of each item, and can send notifications when items are completed. The system should have full accountability and logging facilities, and should support supervisor actions to account for exceptional circumstances, such as a meal being refunded or walked out on. **(SCENARIO)**

2. E-R. Diagram for Banking System. **(SCENARIO)**

3. A Car rental company maintains a database for all vehicles in its current fleet for all vehicles, it includes the vehicle identification-number,
License- number, manufacturer, model, date of purchase and color. Special data are included for certain types of vehicles.
Trucks: Cargo capacity
Sports Cars: horsepower. renter age requirement
Vans: number of passengers
Off-road vehicles: ground clearance, drivetrain(four-or two -wheel drive)
Construct an ER model for the car rental company database.**(SCENARIO)**

4. Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Eachpayment is for a particular period of time and has an associated due date, and the date when the payment was received.**(SCENARIO)**

5. State and explainthe architecture of DBMS. Draw the ER diagram forbanking systems. (Home loan applications). **(SCENARIO)**

6. Discuss the correspondence between the ER model construct and the relational model constructs. Show how each ER model construct can be mapped to the relational model. Discuss the option for mapping EER Model construct.**(INDIRECT)**

7. State the need for normalization of a database and explain the various normal forms (1st,2nd, 3rd, BCNF, 4^{th}, 5^{th}, and domain-key) with suitable examples **(DIRECT)**
8. What are Normal forms? Explain the types of Normal form with an example.**(INDIRECT)**

9. Explain the various operations in relational algebra with examples. **(INDIRECT)**
10. Explain select , project and Cartesian product operations in relational algebra with an example. **(DIRECT)**
11. List the operations of relational algebra and the purpose of each with example. **(DIRECT)**
12. Discuss the main characteristics of the database approach and how does it differ from traditional file system. **(INDIRECT)**

13. What are the three levels of abstraction in DBMS? **(INDIRECT)**
14. Briefly explain about views of data.**(INDIRECT)**

15. With help of a nest block diagram explain the basic architecture of a database management system. **(INDIRECT)**
16. Briefly explain about database system architecture.**(DIRECT)**
17. Explain the overall architecture of the database system in detail. **(DIRECT)**

18. Suppose that we have the following three tuples in a legal instance of a relation schema S with three attributes ABC (listed in order): (1,2,3) , (4,2,3) and (5,3,3)
(i) Which of the following dependencies can you infer does not hold over, schema S?
(1) A→ B (2) BC → A (3) B → C
(ii) Can you identify any dependencies that hold over S?**(TWISTED)**

19. Given: VAR Exam_Marks BASE RELATION { Student_ID SID, Course_ID CID, Mark INTEGER} KEY {Student ID, Course ID};
Write down the relational algebra expression to give, for each pair of students sitting in the same exam, the absolute value of difference between the marks. Assume you can write ABS (x) to obtain the absolute value of x. **(TWISTED)**

UNIT 2

1. Consider a student registration database comprising of the below given table schema.
Student_File(student_number, student_name, address, telephone)
Course_File(course_number, description, hours, professor_number)
Professor_File(Professor_number, name, office)
Registration_File(student_number, course_number, date)
Consider a suitable sample of tuples/records for the above mentioned tables and write DML statements (SQL) to answer for each queries listed below.
(i) Which courses does a specific professor teach?
(ii) What courses are taught by two specific professors?
(iii) Who teaches a specific course and where is his/her office?
(iv) For a specific student number, in which courses is the student registered and what is his/her name?
(v) Who are the professors for a specific student?
(vi) Who are the students registered in a specific course?**(SCENARIO)**

2. Assume the following table
Degree (degcode, name, subject)
candidate (seatno, degcode, name, semester, month, year, result)
Marks (seatno, degcode,semester, month, year, papcode, marks)
Degcode-degree, Name -name of the degree (MSc. MCOM)
Subject — subject of the courses Eg.Phy, Pap code — paper code eg. A1.
Solve the following queries using SQL:
(i)Write a SELECT statement to display all the degree codes which are there in the candidate table but not present in degree table in the order of degcode. (4)
(ii) Write a SELECT statement to display the name of all the dates who have got less than 40 marks in exactly 2 subjects (4)
(iii) Write SELECT statement to display the name, subject and number of candidates for all degreesin which there are less than 5 candidates. (4)
(iv)Write a SELECT statement to display the names of all the candidates who have got highest total marks in MSc., (Maths).(4)**(SCENARIO)**

3. Consider the database schema
   Emp(emp-name, type, birthday, set of(exam-names), set of(skills))
   Children(Emp-name, Ch-name, birthday)
   Skills(type, set of(Exam-names))
   Exams(Exam-name, year, city)
   Write SQL statements for the following queries. (2)
   (i) Find the names of all employees who have birthday in March as their children. (2)
   (ii) Find those employees who took an examination for the skill type "typing" in the city "Chennai". (2)
   (iii) List all exam names under specific skill type for the given employee other than his exam names. (2)
   (iv) Find the names of the city and year where the examination is going to held for the given skill type. (2)
   (v) Explain referential integrity with an example. (8)**(SCENARIO)**

.4. Consider the relation schema given in Figure 1.Design and draw an ER diagram that capture the information of this schema.
   Employee (empno, name, office, age)
   Books (isbn, title, authors, publisher)
   Loan (empno, isbn, date)
   Write the following queries in SQL (a) Find the name of employees who have borrowed a book published by McGraw-Hill. (b) Find the name of employees who have borrowed all book published by McGraw-Hill. (16) **(SCENARIO)**

5. Consider the following relational schema:
   Employee (empno, name, office, age)
   Books (isbn, title, authors, publisher)
   Loan (empnoisbn, date)
   Write the following queries in relational algebra.
   (i) Find the names of employees Who have borrowed a book Published by XYZ Ltd.,
   (ii) Find the names of employees who have borrowed all books Published by XYZ Ltd.,
   (iii) Find the names of employees who have borrowed more than five different books published by XYZ Ltd.,
   (iv) For each Publisher, find the names of employees who have borrowed more than five books of that Publisher. **(SCENARIO)**

6. Write the DDL, DML, DCL commands for the student's database, which contains student details: name, id, DOB, branch, DOJ.
   Course details : Course name, Course id, Stud. id, Faculty name, id, marks. **(SCENARIO)**

7. Let relations r_1(A,B,C)and r_2(C,D,E) have the following properties: r_1 has 20,000 tuples, r_2 has 45,000 tuples, 25 tuples of r_1 fit on one block and 30 tuples of r_2 fit on one block. Estimate the number of block transfers and seeks required, using each of the following join strategies for $r_1 \infty$ r2:(i) Nested_loop join, (ii) Block nested-loop join, (iii) Merge join, (iv) Hash join.**(SCENARIO)**

8. Explain the following with examples: **(DIRECT)**
   (i) DDL
   (ii) DML

9. Explain about SQL Fundamentals. **(DIRECT)**
10. Explain about data definition language. **(DIRECT)**

11. State and explain the command DDL, DML, DCL with suitable example. **(DIRECT)**
12. Describe the six clauses in the syntax of an/SQL query, and show what type of constructs can be specified in each of the six clauses. Which of the six clauses are required and which are optional? **(INDIRECT)**

13. Give a detailed description about Query Processing and Optimization. Explain the cost estimation of Query Optimization. **(DIRECT)**
14. Discuss about the join order optimization and heuristic optimization algorithms. **(DIRECT)**
15. Explain query optimization with an example. **(DIRECT)**
16. Briefly explain about Query Processing. **(DIRECT)**
17. What is meant by semantic query optimization? How does it differ from other query optimization technique? Give example. **(INDIRECT)**
18. Explain the catalog information for cost estimation for selection and sorting operation in database.**(INDIRECT)**
19. How does a DBMS represent a relational query evaluation plan?**(INDIRECT)**
20. Since indices speed query processing why might they not be kept on several search keys? List as many reasons as Possible.**(TWISTED)**

21. Justify the need of embedded SQL. Consider the relation student(RegNo,name and grade).Write embedded dynamic SQL program in C language to retrieve all students records whose mark is more than 90. **(INDIRECT)**
22. What is embedded SQL? Give example.**(DIRECT)**

UNIT 3

1.Explain the Two-phase and Three-phase commit protocols. **(DIRECT)**
2.Consider the following schedules. The actions are listed in the order they are scheduled, and prefixed with the transaction name. (APR/MAY 2015)

S1: T1:R(X), T2:R(X), T1:W(Y), T2:W(Y), T1:R(Y), T2:R(Y)
S2: T3:W(X), T1:R(X), T1:W(Y), T2:R(Z), T2:W(Z), T3:R(Z)
For each of the schedules, answer the following questions;
(i) What is the precedence graph for the schedule? (2)
(ii) Is the schedule conflict-serializable? If so, what are all the conflict equivalent serial schedules? (7)
(iii) Is the schedule view-serializable? If so, what are all the view equivalent serial schedules? (7)**(SCENARIO)**

3. Consider the following two transactions:

T_1 read(A);
read(B);
if A=0 then B:=B+1;
write(B);
T_2: read(B);
read(A);
if B=0 then A:=A+1;
write(A);

4. Add lock and unlock instructions to transactions T_1 and T2, so that they observe the two-phase locking protocol. Can the execution of these transactions result in deadlock? **(SCENARIO)**

5. Consider the following extension to the three-locking protocol, which allows both shared and exclusive locks:
   - A transaction can be either a read-only transaction, in which case it can request only shared locks, or an update transaction, in which case it can request only exclusive locks.
   - Each transaction must follow the rules of the tree protocol. Read-only transactions may lock any data item first, whereas update transactions must lock the root first. Show that the protocol ensures serializability and deadlock freedom. **(SCENARIO)**

6. What is Concurrency? Explain it in terms of locking mechanism and two phase Commit Protocol.**(DIRECT)**
7. Write short notes on:
   (i) Transaction concept. **(DIRECT)**
   (ii) Deadlock. **(DIRECT)**
8. (i)What is concurrency control? How is it implemented in DBMS? Illustrate with a suitable example. **(DIRECT)**
   (ii) Discuss view serializability and conflict serializability. **(DIRECT)**
9. What is deadlock? How does it occur? How transactions be written to
   (i)Avoid deadlock **(DIRECT)**
   (ii)Guarantee correct execution. **(INDIRECT)**
   Illustrate with suitable example.
10. Explain the following:
(i) Different locking mechanism used in lock based concurrency control. **(DIRECT)**
(ii) Validation based protocol with an example. **(INDIRECT)**

10. (i) Illustrate two phase locking protocol with an example. **(DIRECT)**
    (ii) Outline deadlock handling mechanisms. **(DIRECT)**
11. Briefly explain about two phase commit protocol. **(DIRECT)**
12. Explain about Locking Protocols. **(DIRECT)**
13. Discuss the violations caused by each of the following: dirty read, on repeatable read and phantoms with suitable example. **(TWISTED)**
14. Explain why timestamp based concurrency control allows schedules that are not recoverable. Describe how it can be modified through buffering to disallow such schedules.**(TWISTED)**
15. State and explain the lock based concurrency control with suitable example.
When does deadlock occur? Explain two-phase commit protocol with example. **(DIRECT)**
16. Explain the methods used to handle Deadlock.**(DIRECT)**
17. (i) Differentiate strict two phase locking protocol and rigorous two phase locking protocol.
    (ii) How the time stamps are implemented? Explain. **(DIRECT)**
18. Explain the concept of Deadlock avoidance and prevention in detail.**(DIRECT)**

UNIT 4

1. With suitable diagrams, discuss about the RAID levels (level 0, level 1, level 0+1, level 3, level 4 and level 5). (16) (APR/MAY 2015) **(DIRECT)**
2. (i) Explain in detail RAID technology. **(DIRECT)**
3. Explain in detail about (i) B+ tree index (ii) B tree index Files. **(DIRECT)**
4. (i) What is RAID? List the different levels in RAID technology and explain its features. **(DIRECT)**
   (ii)Illustrate indexing and hashing techniques with suitable examples. **(DIRECT)**

5. What is the need for building distributed database? Explain important issues in building distributed database with an example.
Explain how distributed database is used in client/server environment. **(DIRECT)**
6.What is RAID? Briefly explain different levels of RAID. Discuss the factors to be considered in choosing a RAID level. **(DIRECT)**
7. (i) Explain the architecture of a distributed database systems. **(DIRECT)**
8. (ii) Explain the concept of RAID. **(DIRECT)**
9. (i) Explain the architecture of a distributed database system. **(DIRECT)**
10. Suppose that you have been hired as a consultant to choose a database system system for your client's application. For each of the following applications, state what type of database system(relational, persistent programming language-based OODB, object relational, do not specify a commercial product) you would recommend. Justify your recommendation. **(TWISTED)**

(i) A computer-aided design system for a manufacturer of airplanes.
(ii) A system to track contributions made to candidates for public office.
(iii) An information system to support the making of movies.

11. Briefly Explain RAID and RAID levels. **(DIRECT)**
12. Briefly explain about B+ tree index file with example. **(DIRECT)**
13. Compare and contrast the distributed databases and the centralized database systems. **(INDIRECT)**
14. Explain what a RAID system is. How does it improve performances and reliability? Discuss the level 3 and level 4 of RAID. **(INDIRECT)**
15. i) What are the various feature of distributed database versus centralized database system? **(INDIRECT)**
ii) Explain the B+ tree indexes on multiple keys with a suitable example. **(INDIRECT)**
16. Explain the distinction between static and dynamic hashing. Discuss the relative merits of each technique in database applications. **(INDIRECT)**
17. Explain why allocations of records to blocks affect database system performance significantly.**(SCENARIO)**
18. (i) Explain how reliability can be improved through redundancy? **(INDIRECT)**
(ii). How the records are represented and organized in files. Explain with suitable example. **(INDIRECT)**

UNIT 5

1.Write short notes on Distributed Transactions. **(DIRECT)**
2. Suppose an Object oriented database had an object A, which references object B, which in turn references object C. Assume all objects are on disk initially? Suppose a program first & references A, then dereferences B by following the reference from A, and then finally dereferences C. Show the objects that are represented in memory after each dereference, along with their state. **(SCENARIO)**
3. Give the DTD or XML schema for an XML representation of the following nested- relational schema:
Emp=(ename, ChildrenSetsetoff(children), SkillsSet setoff(Skills)
Children = (name, Birthday)
Birthday = (day, month, year)
Skills = (type, ExamSetsetoff(Exams))
Exams = (year, city)**(SCENARIO)**
4. Consider the following bitmap technique for tracking free spaces in a file. For each block in the file, two bits are maintained in the bitmap. If the block is between, 0 and 30 percent full, the bits are 00, between 30 and 60 percent the bits are 00, between 30 and 60 percent the bits are 01,

between 60 and 90 percent the bits are 10, and above 90 percent the bits are 11. Such bitmaps can be kept in memory even for quite large files. **(SCENARIO)**

(i) Describe how to keep the bitmap up to date on record insertions and deletions.

(ii) Outline the benefit of the bitmap techniques over free lists in searching for free space andin updating free space information.

5. Explain about Distributed Databases and their characteristics, functions and advantages and disadvantages. **(DIRECT)**

6. What are the basic crawling operations? Explain the processing steps in crawling procedure with example. **(DIRECT)**

7. Explain the process of querying XML data with an example. **(INDIRECT)**

8. State the necessity for crawling and indexing the web. Explain the procedure for it. **(INDIRECT)**

9. (i) Compare and contrast between object oriented and XML databases. **(INDIRECT)**

(ii) Give XML representation of bank management system and also explain about Document Type Definition and XML schema.**(SCENARIO)**

EXERCISE QUESTIONS

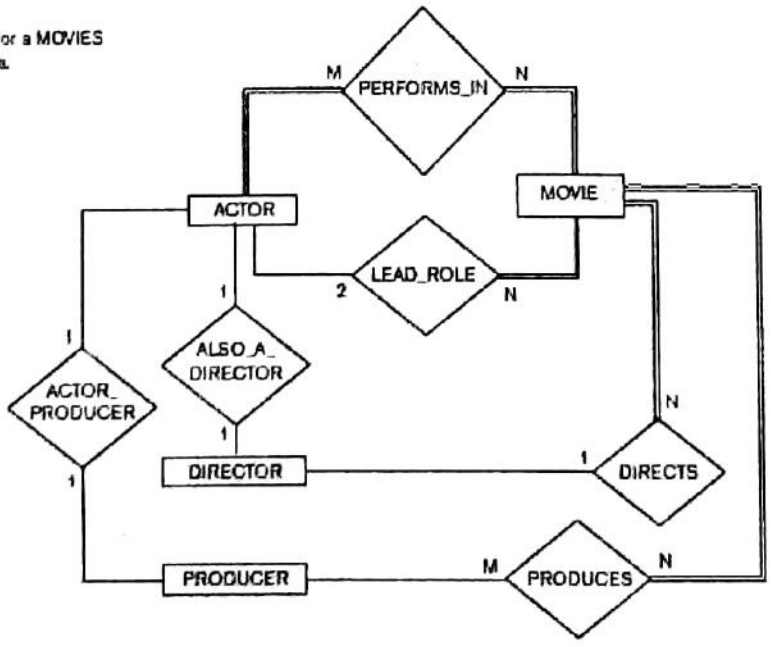
UNIT 1

1. Describe the three-schema architecture.Why do we need mappings betweenschema levels? How do different schema definition languages support thisarchitecture?

UNIT 2

2. Draw an ER diagram for the MOVIES .Drawinstances of each entity type: MOVIES, ACTORS, PRODUCERS, DIRECTORSinvolved; make up instances of the relationships as they exist in reality forthose movies.

Figure 7.24
An ER diagram for a MOVIES database schema.



3. Draw an ER diagram shown for a BANK database.Each bank can have multiple branches, and each branch can have multipleaccounts and loans.

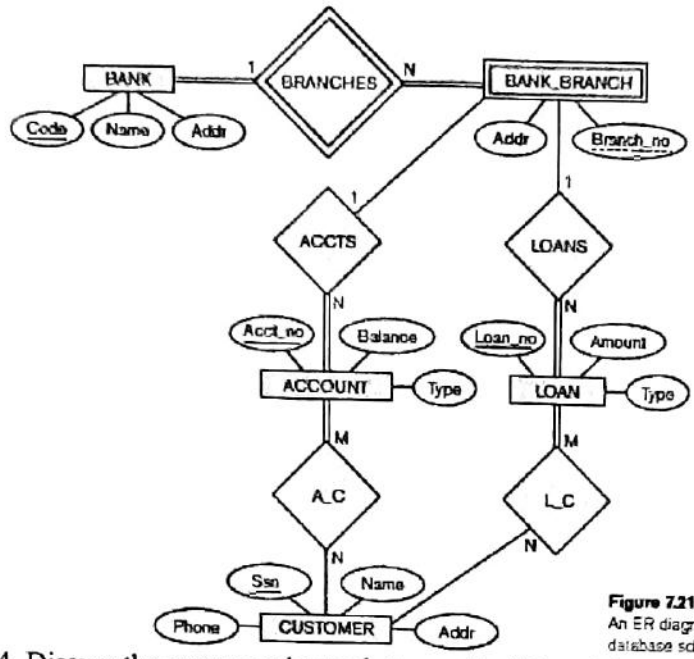
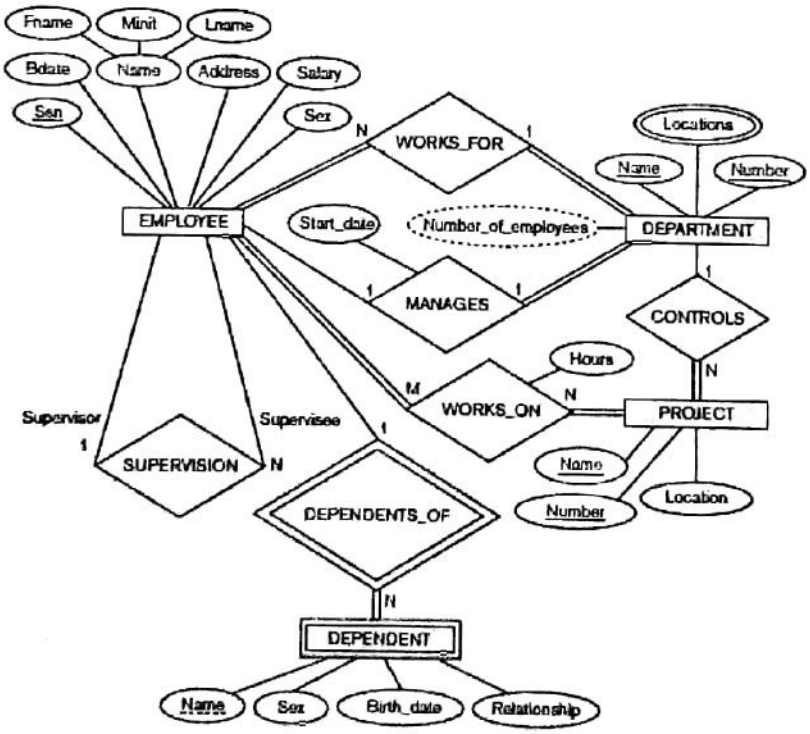


Figure 7.21
An ER diagram for a BANK database schema.

4. Discuss the correspondences between the ER model constructs and the relationalmodel constructs. Show how each ER model construct can be mappedto the relational model and discuss any alternative mappings.

Figure 9.1
The ER conceptual schema diagram for the COMPANY database.



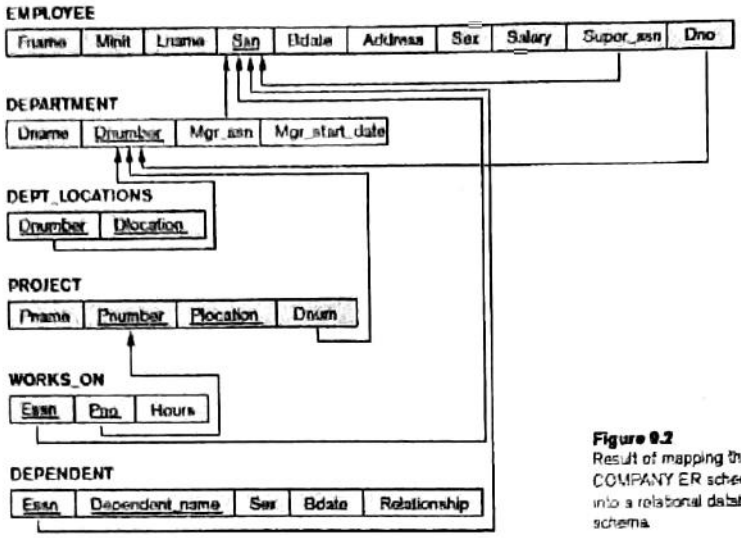


Figure 8.2
Result of mapping the COMPANY ER schema into a relational database schema.

5. Discuss insertion, deletion, and modification anomalies. Why are they consideredbad? Illustrate with examples.

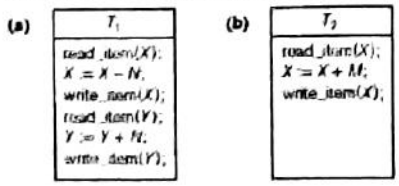
6. Define Boyce-Codd normal form. How does it differ from 3NF? Why is itconsidered a stronger form of 3NF?

UNIT 3

7. Which of the following schedules is (conflict) serializable? For each serializableschedule, determine the equivalent serial schedules.
a. r1(X); r3(X); w1(X); r2(X); w3(X);
b. r1(X); r3(X); w3(X); w1(X); r2(X);
c. r3(X); r2(X); w3(X); r1(X); w1(X);
d. r3(X); r2(X); r1(X); w3(X); w1(X);

8. Consider the three transactions T1, T2, and T3, and the schedules S1 and S2given below. Draw the serializability (precedence) graphs for S1 and S2, andstate whether each schedule is serializable or not. If a schedule is serializable,write down the equivalent serial schedule(s).
T1: r1 (X); r1 (Z); w1 (X);
T2: r2 (Z); r2 (Y); w2 (Z); w2 (Y);
T3: r3 (X); r3 (Y); w3 (Y);
S1: r1 (X); r2 (Z); r1 (Z); r3 (X); r3 (Y); w1 (X); w3 (Y); r2 (Y); w2 (Z); w2 (Y);
S2: r1 (X); r2 (Z); r3 (X); r1 (Z); r2 (Y); r3 (Y); w1 (X); w2 (Z); w3 (Y); w2 (Y);

9. List all possible schedules for transactions T1 and T2 in given figure, anddetermine which are conflictserializable (correct) and which are not.



10. Prove that the wait-die and wound-wait protocols avoid deadlock and starvation.
11. Prove that cautious waiting avoids deadlock.

UNIT 4

12. Discuss the reasons for converting SQL queries into relational algebraqueries before optimization is done.
13. Discuss the different algorithms for implementing each of the followingrelational operators and the circumstances under which each algorithm canbe used: SELECT, JOIN, PROJECT, UNION, INTERSECT, SET DIFFERENCE,CARTESIAN PRODUCT.
14. What is a query execution plan?
15. What is meant by the term heuristic optimization? Discuss the main heuristicsthat are applied during query optimization.
16. How does a query tree represent a relational algebra expression? What ismeant by an execution of a query tree? Discuss the rules for transformationof query trees and identify when each rule should be applied during optimization.
17. What is meant by semantic query optimization? How does it differ fromother query optimization techniques?
18. What is meant by cost-based query optimization?

19. Consider a disk with block size B = 512 bytes. A block pointer is P = 6 byteslong, and a record pointer is PR = 7 bytes long. A file has r = 30,000EMPLOYEE records of fixed length. Each record has the following fields: Name(30 bytes), Ssn (9 bytes), Department_code (9 bytes), Address (40 bytes),Phone (10 bytes), Birth_date (8 bytes), Sex (1 byte), Job_code (4 bytes), and Salary (4 bytes, real number).An additional byte is used as a deletion marker.
a. Calculate the record size R in bytes.
b. Calculate the blocking factor bfr and the number of file blocks b, assumingan unspanned organization.
20. How does disk mirroring help improve reliability? Give a quantitativeexample.
21. What characterizes the levels in RAID organization?
22. What are the highlights of the popular RAID levels 0, 1, and 5?

UNIT 5

23. Discuss what is meant by the following terms: degree of homogeneity of aDDBMS, degree of local autonomy of a DDBMS, federated DBMS, distributiontransparency, fragmentation transparency, replication transparency,multidatabase system.
24. Discuss the architecture of a DDBMS. Within the context of a centralizedDBMS, briefly explain new components introduced by the distribution ofdata.
25. Give a general definition of information retrieval (IR).What does informationretrieval involve when we consider information on the Web?
26. Discuss the types of data and the types of users in today's informationretrieval systems.
27. What is meant by navigational, informational, and transformational search?
28. What are the two main modes of interaction with an IR system? Describewith examples.

CS8492 DATABASE MANAGEMENT SYSTEMS
ANNA UNIVERSITY QUESTION BANK

PART A

UNIT -1

1. Write the characteristics that distinguish the database approach with the file-based approach. (APR/MAY 2015)
2. Define: Functional dependency. (APR/MAY 2015)
3. Why 4NF in normal form is more desirable than BCNF? (NOV/DEC 2014)
4. What is the purpose of Database Management System? (NOV/DEC 2014)
5. State the anomalies of 1NF. (NOV/DEC 2015)
6. Is it possible for several attributes to have the same domain? Illustrate your answer with suitable examples. (NOV/DEC 2015)
7. What do you mean by simple and composite attribute? (NOV/DEC 2013)
8. Define trivial functional dependency. (NOV/DEC 2013)
9. Define functional dependency. (NOV/DEC 2013)
10. What is an entity relationship model? (APR/ MAY 2011)
11. Define single valued and multi valued attributes. (NOV/DEC 2011)
12. What are the disadvantages of file processing system? (MAY/JUNE 16)
13. Explain entity relationship model. (MAY/JUNE 16)
14. What are the desirable properties of decomposition? (APR/MAY 2017)
15. Distinguish between key and super key. (APR/MAY 2017)
16. State the levels of abstraction in a DBMS. (NOV/DEC 2017)
17. What are the problems caused by redundancy? (NOV/DEC 2017)
18. Mention some of the major responsibilities of a database administrator.(NOV/DEC 2018)
19. Give an example for one to one and one to many relationships.(NOV/DEC 2018)

UNIT -2

1. State the need for query optimization. (APR/MAY 2015)
2. What is the difference between static and dynamic SQL?(APR/MAY 2015)
3. Differentiate between Dynamic SQL and Static SQL.(NOV/DEC 2014, 2016)
4. Give a brief description on DCL command. (NOV/DEC 2014)
5. Differentiate static and dynamic SQL. (NOV/DEC 2015)
6. Why does SQL allow duplicate tuples in a table or in a query result? (NOV/DEC 2015)
7. Define query. (NOV/DEC 2013)
8. What is transaction? (APR/ MAY 2011)
9. What is meant by Cost Estimation? (NOV/DEC 2011)
10. With an example explain referential integrity. (NOV/DEC 2013)
11. What is data definition language? Give example. (NOV/DEC 2016)
12. Name the categories of SQL command. (MAY/JUNE 16)
13. Explain "Query optimization". (MAY/JUNE 16)
14. What is query execution plan? (APR/MAY 2017)
15. Which cost components are used most often as the basis for cost function? (APR/MAY 2017)
16. What is static SQL and how is it different from dynamic SQL? (NOV/DEC 2017)
17. State the steps in query processing. (NOV/DEC 2017)
18. What are aggregate functions? And list the aggregate functions supported by SQL?(NOV/DEC 2018)
19. Write a SQL statement to find the names and loan numbers of all customers who have a loan at XYZ branch.(NOV/DEC 2018)

UNIT -3

1. Write the ACID properties of Transaction. (APR/MAY 2015)
2. Define: DDL, DML, DCL, and TCL. (APR/MAY 2015)
3. Define the properties of Transaction.(NOV/DEC 2014)
4. What is Serializability? How it is tested? (NOV/DEC 2014, 2016)
5. What is meant by concurrency control? (NOV/DEC 2015)
6. Give an example of two phase commit protocol. (NOV/DEC 2015)
7. Define deadlock. (NOV/DEC 2011)
8. What are two pitfalls (problems) of lock-based protocols? (NOV/DEC 2013)
9. Define deadlock. (NOV/DEC 2013)
10. What are the states of transaction? (NOV/DEC 2011)
11. List the four conditions for deadlock. (NOV/DEC 2016)
12. What are the properties of transaction? (MAY/JUNE 16)
13. Differentiate strict two phase locking protocol and rigorous two phase locking protocol. (MAY/JUNE 16)
14. What is serializable schedule? (APR/MAY 2017)
15. What type of locking needed for insert and delete operations? (APR/MAY 2017)
16. State need for concurrency. (NOV/DEC 2017)
17. Define ACID properties. (NOV/DEC 2017)
18. Highlight the role of a recovery management component.(NOV/DEC 2018)
19. Give the drawbacks of shadow-paging technique.(NOV/DEC 2018)

UNIT 4

1. How dynamic hashing differ from static hashing? (APR/MAY 2015)
2. Write about four types (Star, snowflake, and galaxy and fact constellation) of data warehouse schemas. (APR/MAY 2015)
3. Differentiate between Static and Dynamic Hashing.(NOV/DEC 2014)
4. Define Data Mining and Data Warehousing.(NOV/DEC 2014)
5. Differentiate static and dynamic hashing. (NOV/DEC 2015)
6. Give an example of a join that is not a simple equi-join for which partitioned parallelism can be used. (NOV/DEC 2015)
7. Describe flash memory. (NOV/DEC 2013)
8. How does B-tree differ from a B+ – tree? Why is a B+ – tree usually preferred as an access structure to a data file? (NOV/DEC 2013)
9. State the advantages of distributed systems. (NOV/DEC 2011)
10. What is called mirroring? (NOV/DEC 2011)
11. List out the mechanisms to avoid collision during hashing. (NOV/DEC 2016)
12. What are the advantages of B Tree over B+ Tree? (NOV/DEC 2016)
13. What is meant by garbage collection? (MAY/JUNE 16)
14. Define software and hardware RAID systems. (MAY/JUNE 16)
15. Define replication transparency. (APR/MAY 2017)
16. What are data fragmentations? State the various fragmentations with example. (NOV/DEC 2017)
17. Define ordered indices with example. (NOV/DEC 2017)
18. Why is a B+- tree usually preferred as an access structure to a data file? (NOV/DEC 2018)
19. What are the ways in which the variable-length records represented in database systems? (NOV/DEC 2018)

UNIT -5

1. What is Crawling and Indexing the web?(NOV/DEC 2014)
2. What is Relevance Ranking?(NOV/DEC 2014)
3. List the types of privileges used in database access control. (NOV/DEC 2015)
4. Define distributed database management systems. (NOV/DEC 2016)
5. State the function of XML schema. (NOV/DEC 2017)
6. How are transactions performed in Object oriented database?(NOV/DEC 2018)

PART B

UNIT 1

1. Draw an ER diagram for the "Restaurant Menu Ordering System", which will facilitate the food items ordering and services within a restaurant. The entire restaurant scenario is detailed as follows. The customer is able to view the food items menu, call the waiter, place orders and obtain the final bill through the computer kept in their table. The waiters through their wireless tablet PC are able to initialize a table for customers, control the table functions to assist customers, orders, send orders to food preparation staff (chef) and finalize the customer's bill. The food preparation staffs (chefs), with their touch-display interfaces to the system, are able to view orders sent to the kitchen by waiters. During preparation, they are able to let the waiter know the status of each item, and can send notifications when items are completed. The system should have full accountability and logging facilities, and should support supervisor actions to account for exceptional circumstances, such as a meal being refunded or walked out on. (16) (APR/MAY 2015) **(REPEATED)**
2. State the need for normalization of a database and explain the various normal forms (1st,2nd, 3rd, BCNF, 4^{th}, 5^{th}, and domain-key) with suitable examples (16) (APR/MAY 2015) **(REPEATED)**
3. Write Short Notes on; (16) (NOV/DEC 2014)
   (i) Data Model and its Types.
   (ii) E-R. Diagram for Banking System. **(REPEATED)**
4. What are Normal forms? Explain the types of Normal form with an example. (16)(NOV/DEC 2014) **(REPEATED)**
5. Explain the various operations in relational algebra with examples. (16) (MAY/JUNE 2014)
6. (i) Discuss the main characteristics of the database approach and how does it differ from traditional file system. (NOV/DEC 2013)
   (ii) What are the three levels of abstraction in DBMS? (8 + 8)
7. (i)With help of a nest block diagram explain the basic architecture of a database management system.(8) (NOV/DEC 2015)
   (ii)What are the advantages of having a centralized control of data? Illustrate your answer with suitable example. (8)
8. A Car rental company maintains a database for all vehicles in its current fleet for all vehicles, it includes the vehicle identification-number,
   License- number, manufacturer, model, date of purchase and color. Special data are included for certain types of vehicles. (16) (NOV/DEC 2015)
   Trucks: Cargo capacity
   Sports Cars: horsepower. renter age requirement
   Vans: number of passengers
   Off-road vehicles: ground clearance, drivetrain(four-or two -wheel drive)
   Construct an ER model for the car rental company database.
9. (i) Explain select , project and Cartesian product operations in relational algebra with an example. (6) (NOV/DEC 2016)
   (ii) Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded

accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Eachpayment is for a particular period of time and has an associated due date, and the date when the payment was received. (7) (NOV/DEC 2016)(NOV/DEC 2018) **(REPEATED)**

10. Explain first normal form, second normal form, third normal form and BCNF with an example. (NOV/DEC 2016)
11. Briefly explain about database system architecture. (MAY/JUNE 16)
12. Briefly explain about views of data. (MAY/JUNE 16)
13. Discuss the correspondence between the ER model construct and the relational model constructs. Show how each ER model construct can be mapped to the relational model. Discuss the option for mapping EER Model construct. (APR/MAY 2017)
14. Explain the overall architecture of the database system in detail. (APR/MAY 2017)
15. List the operations of relational algebra and the purpose of each with example. (APR/MAY 2017)
16. i) Differentiate between foreign key constraints and referential integrityconstraints with suitable example.
ii) Distinguish between lossless-join decomposition and dependency preserving decomposition. (NOV/DEC 2017)
17. State and explainthe architecture of DBMS. Draw the ER diagram forbanking systems. (Home loan applications). (NOV/DEC 2017)
18. Suppose that we have the following three tuples in a legal instance of a relation schema S with three attributes ABC (listed in order): (1,2,3) , (4,2,3) and (5,3,3)
(i) Which of the following dependencies can you infer does not hold over, schema S?
(1) A→ B (2) BC → A (3) B → C
(ii) Can you identify any dependencies that hold over S?(NOV/DEC 2018)

UNIT 2

1. Consider a student registration database comprising of the below given table schema. (16) (APR/MAY 2015)

Student_File(student_number, student_name, address, telephone)
Course_File(course_number, description, hours, professor_number)
Professor_File(Professor_number, name, office)
Registration_File(student_number, course_number, date)

Consider a suitable sample of tuples/records for the above mentioned tables and write DML statements (SQL) to answer for each queries listed below.

(i) Which courses does a specific professor teach?
(ii) What courses are taught by two specific professors?
(iii) Who teaches a specific course and where is his/her office?
(iv) For a specific student number, in which courses is the student registered and what is his/her name?
(v) Who are the professors for a specific student?
(vi) Who are the students registered in a specific course?

2. Discuss about the join order optimization and heuristic optimization algorithms. (16) (APR/MAY 2015)
3. Explain the following with examples: (NOV/DEC 2014)
(i) DDL (4)
(ii) DML (4)
(iii) Embedded SQL (8)
4. Give a detailed description about Query Processing and Optimization. Explain the cost

estimation of Query Optimization. (16) (NOV/DEC 2014)

5. Describe the six clauses in the syntax of an SQL query, and show what type of constructs can be specified in each of the six clauses.
Which of the six clauses are required and which are optional? (16) (NOV/DEC 2015)

6. Assume the following table (NOV/DEC 2015)

Degree (degcode, name, subject)
candidate (seatno, degcode, name, semester, month, year, result)
Marks (seatno, degcode,semester, month, year, papcode, marks)
Degcode-degree, Name -name of the degree (MSc. MCOM)
Subject — subject of the courses Eg.Phy, Pap code — paper code eg. A1.
Solve the following queries using SQL:
(i)Write a SELECT statement to display all the degree codes which are there in the candidate table but not present in degree table in the order of degcode. (4)
(ii) Write a SELECT statement to display the name of all the dates who have got less than 40 marks in exactly 2 subjects (4)
(iii) Write SELECT statement to display the name, subject and number of candidates for all degreesin which there are less than 5 candidates. (4)
(iv)Write a SELECT statement to display the names of all the candidates who have got highest total marks in MSc., (Maths).(4)

7. Consider the database schema (NOV/DEC 2013)

Emp(emp-name, type, birthday, set of(exam-names), set of(skills))
Children(Emp=name, Ch-name, birthday)
Skills(type, set of(Exam-names))
Exams(Exam-name, year, city)
Write SQL statements for the following queries. (2)
(i) Find the names of all employees who have birthday in March as their children. (2)
(ii) Find those employees who took an examination for the skill type "typing" in the city "Chennai". (2)
(iii) List all exam names under specific skill type for the given employee other than his exam names. (2)
(iv) Find the names of the city and year where the examination is going to held for the given skill type. (2)
(v) Explain referential integrity with an example. (8)

8. Let relations r_1(A,B,C)and r_2(C,D,E) have the following properties: r_1 has 20,000 tuples, r_2 has 45,000 tuples, 25 tuples of r_1 fit on one block and 30 tuples of r_2 fit on one block. Estimate the number of block transfers and seeks required, using each of the following] join strategies for $r_1 \infty r2$:
(i) Nested_loop join, (ii) Block nested-loop join, (iii) Merge join, (iv) Hash join. (13) (NOV/DEC 2016)

9. (i) Explain query optimization with an example. (8) (NOV/DEC 2016)
(ii) What is embedded SQL? Give example. (5) (NOV/DEC 2016)

10.Explain about SQL Fundamentals. (MAY/JUNE 16)

11.Explain about data definition language. (MAY/JUNE 16)

12. Briefly explain about Query Processing. (MAY/JUNE 16)

13.What is meant by semantic query optimization? How does it differ from other query optimization technique? Give example. (APR/MAY 2017)

14. Justify the need of embedded SQL. Consider the relation student(RegNo,name and grade).Write embedded dynamic SQL program in C language to retrieve all students records whose mark is more than 90. (APR/MAY 2017)

15.Consider the relation schema given in Figure 1.Design and draw an ER diagram that capture the information of this schema.

Employee (empno, name, office, age)
Books (isbn, title, authors, publisher)
Loan (empno, isbn, date)
Write the following queries in SQL (a) Find the name of employees who have borrowed a book published by McGraw-Hill. (b) Find the name of employees who have borrowed all book published by McGraw-Hill (16) (APR/MAY 2017)

16. i) State and explain the command DDL, DML, DCL with suitable example.
ii) Justify the need of embedded SQL. Consider the relation student (studentno, name, mark and grade) Write embedded dynamic SQL statements in Clanguage to retrieve all the students' records whose mark is more than 90. (NOV/DEC 2017)

17. Explain the catalog information for cost estimation for selection and sorting operationin database (NOV/DEC 2017)

18. Write the DDL, DML, DCL commands for the student's database, which contains student details: name, id, DOB, branch, DOJ.
Course details : Course name, Course id, Stud. id, Faculty name, id, marks. (15) (NOV/DEC 2017)

19. Consider the following relational schema:
Employee (empno, name, office, age)
Books (isbn, title, authors, publisher)
Loan (empnoisbn, date)
Write the following queries in relational algebra.
(i) Find the names of employees Who have borrowed a book Published by XYZ Ltd.,
(ii) Find the names of employees who have borrowed all books Published by XYZ Ltd.,
(iii) Find the names of employees who have borrowed more than five different books published by XYZ Ltd.,
(iv) For each Publisher, find the names of employees who have borrowed more than five books of that Publisher.(NOV/DEC 2018)

20. (i) Since indices speed query processing why might they not be kept on several search keys? List as many reasons as Possible.
(ii) How does a DBMS represent a relational query evaluation plan?(NOV/DEC 2018)

21. Given: VAR Exam_Marks BASE RELATION { Student_ID SID,Course_ID CID, Mark INTEGER} KEY {Student ID, Course ID);
Write down the relational algebra expression to give, for each pair of students sitting in the same exam, the absolute value of difference between the marks. Assume you can write ABS (x) to obtain the absolute value of x.

UNIT 3

1.Explain the Two-phase and Three-phase commit protocols. (16) (APR/MAY 2015)

2.Consider the following schedules. The actions are listed in the order they are scheduled, and prefixed with the transaction name. (APR/MAY 2015)

S1: T1:R(X), T2:R(X), T1:W(Y), T2:W(Y), T1:R(Y), T2:R(Y)
S2: T3:W(X), T1:R(X), T1:W(Y), T2:R(Z), T2:W(Z), T3:R(Z)
For each of the schedules, answer the following questions;
(i) What is the precedence graph for the schedule? (2)
(ii) Is the schedule conflict-serializable? If so, what are all the conflict equivalent serial schedules? (7)
(iii) Is the schedule view-serializable? If so, what are all the view equivalent serial schedules? (7)

3. What is Concurrency? Explain it in terms of locking mechanism and two phase Commit Protocol. (16) (NOV/DEC 2014) **(REPEATED)**

4. Write short notes on: (NOV/DEC 2014)
   (i) Transaction concept. (8)
   (ii) Deadlock. (8) **(REPEATED)**

5. (i)What is concurrency control? How is it implemented in DBMS? Illustrate with a suitable example. (8) (NOV/DEC 2015)
   (ii) Discuss view serializability and conflict serializability. (8)

6. What is deadlock? How does it occur? How transactions be written to
   (i)Avoid deadlock (8) **(REPEATED)**
   (ii)Guarantee correct execution. (8)
   Illustrate with suitable example. (NOV/DEC 2015)

7. Explain the following: (NOV/DEC 2015)
(i) Different locking mechanism used in lock based concurrency control. **(10) (REPEATED)**
(ii) Validation based protocol with an example. **(6)**

8. (i) Consider the following two transactions:

T_1 read(A);
read(B);
if A=0 then B:=B+1;
write(B);
T_2: read(B);
read(A);
if B=0 then A:=A+1;
write(A);

Add lock and unlock instructions to transactions T_1 and T2, so that they observe the two-phase locking protocol. Can the execution of these transactions result in deadlock? (6) (NOV/DEC 2016)

9. Consider the following extension to the three-locking protocol, which allows both shared and exclusive locks:

- A transaction can be either a read-only transaction, in which case it can request only shared locks, or an update transaction, in which case it can request only exclusive locks.
- Each transaction must follow the rules of the tree protocol. Read-only transactions may lock any data item first, whereas update transactions must lock the root first.

Show that the protocol ensures serializability and deadlock freedom. (7) (NOV/DEC 2016)

10. (i) Illustrate two phase locking protocol with an example. (6) (NOV/DEC 2016)
    (ii) Outline deadlock handling mechanisms. (7) (NOV/DEC 2016)
11. Briefly explain about two phase commit protocol. (MAY/JUNE 16)
12. Explain about Locking Protocols. (MAY/JUNE 16)
13. Discuss the violations caused by each of the following: dirty read, on repeatable read and phantoms with suitable example. (APR/MAY 2017)
14. Explain why timestamp based concurrency control allows schedules that are not recoverable. Describe how it can be modified through buffering to disallow such schedules.(APR/MAY 2017)
15. State and explain the lock based concurrency control with suitable example. (NOV/DEC 2017)
When does deadlock occur? Explain two-phase commit protocol with example. (NOV/DEC 2017)
16. Explain the methods used to handle Deadlock. (NOV/DEC 2018)
17. (i) Differentiate strict two phase locking protocol and rigorous twophase locking protocol.(6)
(ii) How the time stamps are implemented? Explain.(7)(NOV/DEC 2018)
18. Explain the concept of Deadlock avoidance and prevention in detail.(NOV/DEC 2018)

UNIT 4

1. With suitable diagrams, discuss about the RAID levels (level 0, level 1, level 0+1, level 3, level 4 and level 5). (16) (APR/MAY 2015) **(REPEATED)**
2. (i) Explain in detail RAID technology. (8) (NOV/DEC 2014)
3. Explain in detail about (i) B+ tree index (ii) B tree index Files. (16) (NOV/DEC 2014)
4. (i) What is RAID? List the different levels in RAID technology and explain its features. (8) (NOV/DEC 2015) **(REPEATED)**
   (ii)Illustrate indexing and hashing techniques with suitable examples. (8)
5. What is the need for building distributed database? Explain important issues in building distributed database with an example.
Explain how distributed database is used in client/server environment. **(16)** (NOV/DEC 2013)
6. What is RAID? Briefly explain different levels of RAID. Discuss the factors to be considered in choosing a RAID level. **(16)** (NOV/DEC 2015) **(REPEATED)**
7. (i) Explain the architecture of a distributed database systems. (7) (NOV/DEC 2016)
   (ii) Explain the concept of RAID. (7) (NOV/DEC 2016)
8. (ii) Explain the concept of RAID. (7)(NOV/DEC 2016)
9. (i) Explain the architecture of a distributed database system. (7) (NOV/DEC 2016)
10. Suppose that you have been hired as a consultant to choose a database system system for your client's application. For each of the following applications, state what type of database system(relational, persistent programming language-based OODB, object relational, do not specify a commercial product) you would recommend. Justify your recommendation. (13) (NOV/DEC 2016)
    (i) A computer-aided design system for a manufacturer of airplanes.
    (ii) A system to track contributions made to candidates for public office.
    (iii) An information system to support the making of movies.
11. Briefly Explain RAID and RAID levels. (MAY/JUNE 16)
12. Briefly explain about B+ tree index file with example. (MAY/JUNE 16)
13. Compare and contrast the distributed databases and the centralized database systems. (APR/MAY 2017)
14. Explain what a RAID system is. How does it improve performances and reliability? Discuss the level 3 and level 4 of RAID. (APR/MAY 2017)
15. i) What are the various feature of distributed database versus centralized database system? (6)
    ii) Explain the B+ tree indexes on multiple keys with a suitable example. (NOV/DEC 2017)
16. Explain the distinction between static and dynamic hashing. Discuss the relative merits of each technique in database applications. (NOV/DEC 2017)
17. Explain why allocations of records to blocks affect database system performance significantly. (NOV/DEC 2018)
18. (i) Explain how reliability can be improved through redundancy? (6)
(ii). How the records are represented and organized in files. Explain with suitable example. (7) (NOV/DEC 2018)

UNIT 5

1. Write short notes on Distributed Transactions. (8) (NOV/DEC 2014)
2. Suppose an Object oriented database had an object A, which references object B, which in turn references object C. Assume all objects are on disk initially? Suppose a program first & references A, then dereferences B by following the reference from A, and then finally dereferences C. Show the objects that are represented in memory after each dereference, along with their state. (NOV/DEC 2015).
3. Give the DTD or XML schema for an XML representation of the following nested- relational

schema:
Emp=(ename, ChildrenSetsetof(children), SkillsSet setoff(Skills) (13) (NOV/DEC 2016)
Children = (name, Birthday)
Birthday = (day, month, year)
Skills = (type, ExamSetsetoff(Exams))
Exams = (year, city)

4. Consider the following bitmap technique for tracking free spaces in a file. For each block in the file, two bits are maintained in the bitmap. If the block is between, 0 and 30 percent full, the bits are 00, between 30 and 60 percent the bits are 00, between 30 and 60 percent the bits are 01, between 60 and 90 percent the bits are 10, and above 90 percent the bits are 11. Such bitmaps can be kept in memory even for quite large files. (13) (NOV/DEC 2016)

(i) Describe how to keep the bitmap up to date on record insertions and deletions.

(ii) Outline the benefit of the bitmap techniques over free lists in searching for free space and in updating free space information.

5. Explain about Distributed Databases and their characteristics, functions and advantages and disadvantages. (MAY/JUNE 16)

6. What are the basic crawling operations? Explain the processing steps in crawling procedure with example. (APR/MAY 2017)

7. Explain the process of querying XML data with an example. (APR/MAY 2017)

8. State the necessity for crawling and indexing the web. Explain the procedure for it. (NOV/DEC 2017)

9. (i) Compare and contrast between object oriented and XML databases.(7)
(ii) Give XML representation of bank management system and also explain about Document Type Definition and XML schema. (6)(NOV/DEC 2018)