easy-solutions®

New Edition SPPU

As per the New Revised Syllabus (2019 course)
of Savitribai Phule Pune University w.e.f. academic year 2020-2021

DATABASE MANAGEMENT SYSTEM

(Code : 214452)

"QUICK READ SERIES"

Semester IV
Information Technology

Chapterwise Solved University Paper Solution
For End Semester Examination

TechKnowledge Publications

SYLLABUS

In-Sem Exam

Unit I : Introduction to DBMS — 06 hrs

Introduction : Basic concepts, Advantages of DBMS over file processing systems, Data abstraction, Database languages, Data models, Data independence, Components of a DBMS, Overall structure of DBMS, Multi-user DBMS architecture, System catalogs, Data Modeling: Basic concepts, Entity, attributes, relationships, constraints, keys.

Unit II : Relational Model — 06 hrs

ER and EER diagrams : Components of ER model, Conventions, Converting ER diagrams into tables Relational Model: Basic concepts, Attributes and Domains, Codd's rules.

Relational Integrity : Nulls, Entity, Referential integrities, Enterprise Constraints, Views, Schema, diagram.

End-Sem Exam

Unit III : Introduction to SQL – PL/SQL — 06 hrs

Introduction to SQL : Characteristics and advantages SQL Data Types, Literals, DDL, DML, SQL Operators Tables: Creating, Modifying, Deleting, Views: Creating, Dropping, Updation using Views, Indexes, Nulls.

SQL DML Queries : SELECT query and clauses, Set operations, Tuple Variables, Set comparison, Ordering of Tuples , Aggregate Functions, Nested Queries, Database Modification using SQL Insert, Update, Delete Queries, Stored Procedure, Triggers, Programmatic SQL : Embedded SQL, Dynamic SQL, ODBC

Unit IV : Database Design & Query Processing — 06 hrs

Relational Databases Design : Purpose of Normalization, Data Redundancy and Update Anomalies, Functional Dependencies. The process of Normalization: 1NF, 2NF, 3NF, BCNF. Introduction to **Query Processing** : Overview, Measures of Query cost, Selection and Join operations, Evaluation of Expressions

Introduction to Query optimization : Estimation, Transformation of Relational Expression

Unit V : Transaction & Concurrency Control — 06 hrs

Transaction Management : Basic concept of a Transaction, Properties of Transactions, Database Architecture, Concept of Schedule, Serial Schedule.

Serializability : Conflict and View, Cascaded aborts Recoverable and Non-recoverable Schedules. Concurrency Control: Need Locking methods Dead locks, Time stamping Methods. Optimistic Techniques, Multi-version Concurrency Control.

Different crash recovery methods : Shadow-Paging, Log-based Recovery: Deferred and Immediate, Check Points

Unit VI : Advanced Databases **06 hrs**

Recent Trends in SE : SCM, Risk Management, Technology evolution, process trends, collaborative development, software reuse, test-driven development, global software development challenges, CASE – taxonomy, tool-kits, workbenches, environments, components of CASE, categories (upper, lower and integrated CASE tools), Introduction to agile tools Jira, Kanban

❑❑❑

Table of Contents

Unit III : Introduction to SQL – PL/SQL **1 to 23**

- **Chapter 4 :** Introduction to SQL
- **Chapter 5 :** Tables
- **Chapter 6 :** Views
- **Chapter 7 :** SQL Queries
- **Chapter 8 :** Stored Procedures
- **Chapter 9 :** Programmatic SQL

Unit IV : Database Design & Query Processing **24 to 32**

- **Chapter 10 :** Relational Database Design
- **Chapter 11 :** Query Processing

Unit V : Transaction & Concurrency Control **33 to 51**

- **Chapter 12 :** Transaction Management
- **Chapter 13 :** Concurrency Control
- **Chapter 14 :** Recovery Methods

Unit VI : Advanced Databases **51 to 64**

- **Chapter 15 :** Database Architecture
- **Chapter 16 :** NoSQL Database

❑❑❑

Savtribai Phule Pune University
Semester - IV (Information Technology)

Database Management System

Unit III

Chapter - 4 : Introduction to SQL

Q.1 Explain role of SQL with example. **(4 Marks)**

Ans. :

Role of SQL

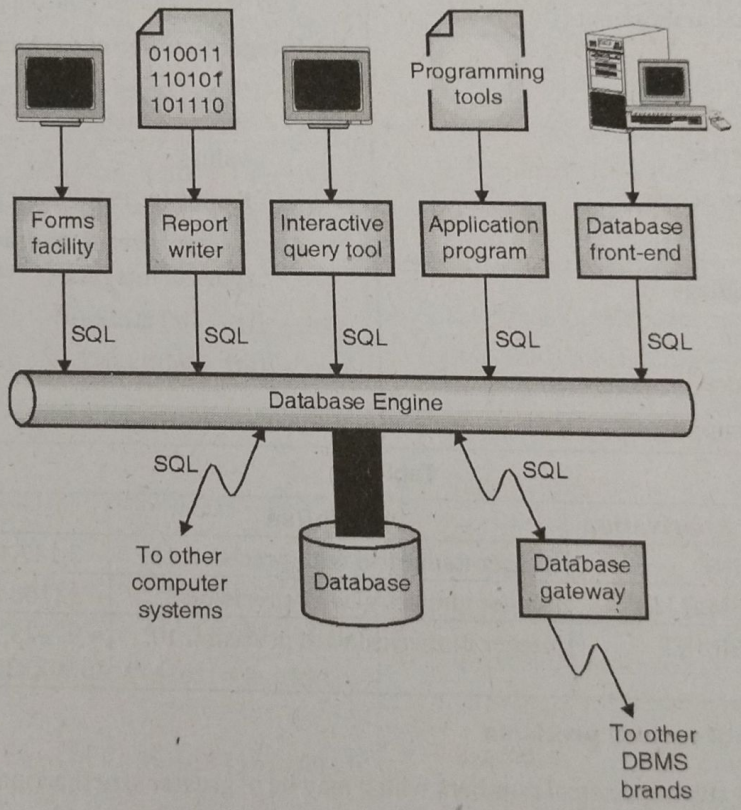


Fig. 4.1 : Role of SQL in DBMS

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

Q.2 What are the characteristics and Advantages of SQL ? **(4 Marks)**

Ans. : Characteristics and Advantages of SQL

SQL is both an easy-to-understand language and a comprehensive tool for managing data. Some of the major features of SQL are listed as follows :

1. Microsoft commitment (SQL Server, ODBC, and ADO)
2. Internet database access
3. Java integration (JDBC)
4. Vendor independence
5. Portability across computer systems
6. SQL standards
7. IBM endorsement and commitment (DB2)
8. Relational foundation
9. High-level, English-like structure
10. Interactive, ad hoc queries
11. Programmatic database access
12. Multiple views of data
13. Complete database language
14. Dynamic data definition
15. Client/server architecture
16. Enterprise application support
17. Extensibility and object technology
18. Industry infrastructure

Q.3 Discuss various MYSQL data types. **SPPU - Dec. 17, 6 Marks**

Ans. : MySQL Data Types

1. Built-in Data Types

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

1. Numeric Data Types

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

(a) Integer number of various size

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

  Example: 111, 23 etc.

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

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

  (iii) SMALLINT (iv) BIGINT

Table 4.1

Sr. No.	Data type	Abbrivation	Description	Range
1.	INTEGER	INT	Integer numerical with precision 10.	– 2,147,483,648 to 2,147,483,647
2.	SMALL INTEGER	SMALLINT	Integer numerical with precision 5.	– 32768 to 32767
3.	BIG INTEGER	BIGINT	Integer numerical with precision 19.	– 9, 223, 372, 036, 854, 775, 808 to 9, 223, 372, 036, 854, 775, 807

(b) Floating point numbers of various precision

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

  Example : 11.2, 12.3 etc.

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

  (i) FLOAT or REAL (ii) DOUBLE PRECISION

Table 4.2

Sr.No.	Data Type	Abbrivation	Description	Range
1.	FLOAT (p)	FLOAT (p)	Approximate numerical with mantissa precision p.	$1 \le p \le 45$ Zero or absolute value 10^{-999} to 10^{+999}
2.	REAL	REAL	Approximate numerical with mantissa precision 7.	Zero or absolute value 10^{-38} to 10^{+38}

(c) Formatted numbers

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

Example :

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

(i) DECIMAL or DEC(i, j)

(ii) NUMERIC(i, j)

Where i = Precision = Total number of digits after decimal point

j = Scale = Total number of digits after decimal point.

Table 4.3 : (default value is 0)

Data Type	Description	Range
DECIMAL(p, s)	Exact numerical with precision p and scale s.	$1 \le p \le 45$ $0 \le s \le p$
NUMERIC(p, s)	Exact numerical with precision p and scale s.	$1 \le p \le 45$ $0 \le s \le p$

2. Character String Data Type

- This data type is used to store a character string which is combination of some alpha bates and enclosed in single quotation marks.

  Example : 'Maheṣh','abc' etc.

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

Example : If 'abc' is stored in char (10) will be stored as 'abc '. (abc padded with 7 blank spaces)

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

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

Table 4.4

Data Type	Description	Range
CHAR(n)	Character string of fixed length n.	$1 \le n \le 15000$
VARCHAR(n)	Variable length character string of maximum length n.	$1 \le n \le 15000$

3. Date Time Data Type

Quick Read

(a) Date

- The DATE data type has ten positions, and its components are YEAR, MONTH and DAY in form YYYY-MM-DD.
- The length is 10.
- Generally not supported by SQL server.(Supported by DB2)
- **Example :** Date '2009-01-01' (as 'YYYY-MM-DD')

(b) Time

- The TIME data type has at least eight positions, and its components are HOUR, MINUTES and SECOND in form HH:MM:SS [.sF] where, F is the fractional part of the SECOND value.
- Generally not supported by SQL server.
- If a second's precision is not specified, s defaults to 0. The length is 8 (or 9 + s, if $s > 0$).
- **Example :** Time '11:16:59' (as 'HH:MM:SS')

(c) Timestamp / datetime

- The TIMESTAMP data type includes both date and time fields, plus a minimum of six positions and for decimal fractions of second and optional with TIMEZONE Qualifier.
- Represented using the fields YEAR, MONTH, DAY, HOUR, MINUTE and SECOND in the format YYYY-MM-DD HH:MM:SS[.sF] where, F is the fractional part of the SECOND value.
- If a second precision is not specified, s defaults to 6. The length is 26 (or 19, if s = 0 or 20+s, if $s > 0$).
- **Example :** Time Stamp '2009:01:01 11:16:59 648302'

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

  CurrentTimeStamp : Local date and time without time zone.

(d) Interval

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

INTERVAL YEAR TO MONTH Datatype.

Example 1 :

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

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

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

INTERVAL DAY TO SECOND Datatype.

Example 2:

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

Generally not supported by SQL server but supported by Oracle.

2. Literals

- Literals are explicit representations of data in for of some expressions.
- Literals are used in SQL statements to represent data that does not change.

1. **Character Literal :** The character data stored in character type of data is character literal.

   Char chr='test';

2. **Numeric Literal :** The numeric data stored in character type of data is character literal.

   Number(10,2) num_1=1002;

3. **Boolean Literals :** The character Boolean data is stored.

   Boolean bln='True';

Q.4 Explain DDL commands. **(4 Marks)**

Ans. : DDL Commands

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

  Example : Table, View, Sequence etc.

❑❑❑

Chapter – 5 : Tables

Q.1 Explain CREATE command with example. **(2 Marks)**

Ans. : Creating Tables Using CREATE Command

- To create database object like table, database, view etc. we use Data Definition Language (DDL).
- DDL statements are used to build and modify the structure of your tables and other objects in the database.
- CREATE statement is used to create any database object.

Syntax :

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

Example :

```
CREATE TABLE Employee
      (  Eid varchar2(10),
      First_Name char (50),
      Last_Name char (50),
      Address char (50),
      City chars (50),
      Country char (25),
      Birth_Date date
      );
```

- To view the structure of table created in database;

```
DESCRIBE Employee;
```

Q.2 Explain ALTER command with example. **(2 Marks)**

Ans. : Modifying Table Using ALTER Command

- Once table is created in database, we may require to change structure of database object. This can be done with help of ALTER command.
- The alter table statement may be used as you have seen to specify primary and foreign key constraints, as well as to make other modifications to the table structure.
- Key constraints may also be specified in the CREATE TABLE statement.

Syntax :

```
ALTER TABLE <Table_Name>
ADD Column_1 datatype ;
OR
ALTER TABLE <Table_Name>
Modify Column_1 New_datatype;
OR
ALTER TABLE <Table_Name>
DROP Column_1;
```

Example :

```
ALTER TABLE Employee
ADD Address Varchar2 (100);
```

To view the changed structure of table

```
DESCRIBE TABLE Employee;
```

Q.3 Describe DROP TABLE command of SQL with both the options CASCADE and RESTRICT.

SPPU - Dec. 18, Oct.19, 2 Marks

Ans. : Removing Table using DROP Command

- This command can be used to remove database objects from our DBMS.
- DROP command removes data from your database.

Syntax :

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

Example :

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

```
DROP TABLE Employee;
```

Similarly, the command below would be used to remove the entire employees database.

```
DROP DATABASE employees ;
```

Q.4 Explain the purpose of foreign key. **(4 Marks)**

Ans. : Foreign Key

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

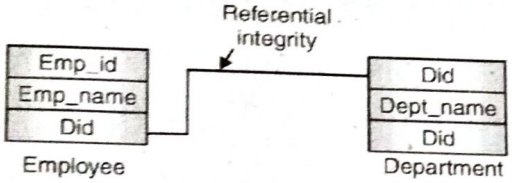


Fig. 5.1 : Referential integrity

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

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

- In the above example 'Emp' table has 'Did' as foreign key reference this is called as referential integrity.
- Here we are forcing database to check the 'Did' value key from the 'department' table while inserting any value of 'Emp' table in Did column if there is no value existing in department table of that 'Did' then we cannot insert that value in 'Emp' table. This helps to maintain data consistency.

Example :

For above shown relation we can write table as follows,

```
Create table Emp ( Emp_id integer,
    Emp_name varchar (100) not null,
    Did as integer references department(did))
Create table Department ( Did integer,
    Dept_name varchar (100) not null,
    Primary key (did));
```

Q.5 Write a note on DML. **(4 Marks)**

Ans. : DML

- DML is set of commands used to,
  - Insert data into table
  - Delete data from table
  - Update data of table

(1) INSERT Statement

Insert statement used to add records to the existing table.

Syntax :

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

Example :

```
/* To add data in employee table*/

INSERT INTO Employee
VALUES (1001, 'Mahesh');

INSERT INTO Employee
VALUES (NULL, 'Jayendra');

INSERTINTO Employee (Name,Eid)
VALUES ('Sachin' , 1002);

INSERTINTO Employee (Name,Eid)
VALUES ('Suhas' , NULL);
```

To check inserted rows, write following query.

```
/*To Print all data in employee table*/
SELECT  *
FROM Employee;
```

Output :

Eid	Name
1001	Mahesh
1002	Sachin
NULL	Jayendra
NULL	Suhas

(2) DELETE Statement

Delete statement is used to delete some or all records from the existing table.

Syntax :

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

Example :

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

- If we omits the WHERE condition then all rows will get deleted.
- To check inserted rows write following query.

```
SELECT  *
FROM    Employee;
```

Output :

Eid	Name
1001	Mahesh
1002	Sachin

(3) UPDATE Statement

Insert statement used to modify the records present in existing table.

Syntax :

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

Example :

```
UPDATE  Employee
SET     Eid=1003
WHERE   name = 'suhas';
```

- If we commit the WHERE condition then all rows will get updated with given value.
- To check inserted rows write following query.

```
SELECT  *
FROM    Employee;
```

Output :

Eid	Name
1001	Mahesh
1002	Sachin
1003	Suhas
NULL	Jayendra

Chapter – 6 : Views

Q.1 Explain the concept of view along with its operations. **SPPU - Dec. 19, 4 Marks**

Ans. :

1. Definition

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

2. Base table

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

3. View – as a window of entire table

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

Example :

Consider a student table contains following columns,

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

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

Stud_Sports_View (Stud_Id, Stud_Name, Sports)

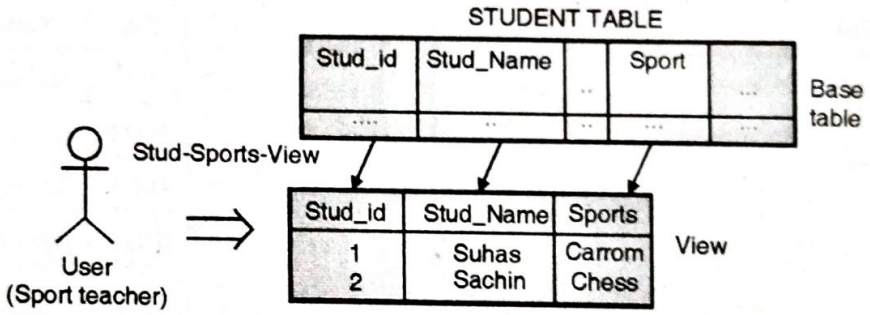


Fig. 6.1 : Overview of view

4. Types of views

(a) Simple view (b) Complex View

(a) Simple view

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

(b) Complex view

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

5. Working of views

- When we call view in SQL query it refers to database and finds definition of views which is already stored in database.

- Then the DBMS convert this call of view into equal request on the base tables of the view and carries out the operations written in view definition and returns result set to query from which view is called.

Q.2 What are advantages of views ? (4 Marks)

Ans. : Advantages of Views

(a) Security

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

(b) Hides complexity

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

(c) Dynamic nature

- View definition remains unaffected although there is any change in structure of a table.
- This dynamic nature does not hold true in case if base table is dropped or the column selected by view is altered.
- **Example :** If view is made on two tables, selecting two columns from first table and two columns from second table if we add one more column to first table does not cause any change on view.

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

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

(e) Data integrity

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

Q. 3 Write short note on SQL Indexes ? (6 Marks)

Ans. : SQL Indexes

1. Introduction

- An index can be created in a table to access data in database more quickly and efficiently.
- MySQL uses indexes to quickly search rows with specific column values as specified in query. Without an index, MySQL engine need to scan the entire table to locate the relevant rows. As table size increases the search speed will be reduced further.
- A database index is a data structure which will improves the speed of operations in a table. Indexes can be created using one or more columns

2. Need of Indexing

- Sometime while fetching data one or more columns in table are more repeating frequently in a WHERE clause of query then indexes on such columns can improve performance and speed of data retrieval.
- The index is generally required on column contains a wide range of values or a large number of null values.

3. Working of Index

- An index will make use of some data structure like B-Tree which will improves the speed of data retrieval on a table, it may include some additional write operations and storage for maintaining it.
- When we create any column of a table with a primary key or unique key, MySQL will automatically create an index named as PRIMARY.
- This type of index is also called the clustered index
- The index creation must consider all columns used in SQL queries and create one or more indexes on such columns.
- Practically, indexes are type of table, which keep key field and a pointer to each record into the table.
- The INSERT and UPDATE statements will take more time on tables due to updates required in index information which creates extra burden on system, whereas the SELECT statements will become fast on such tables.

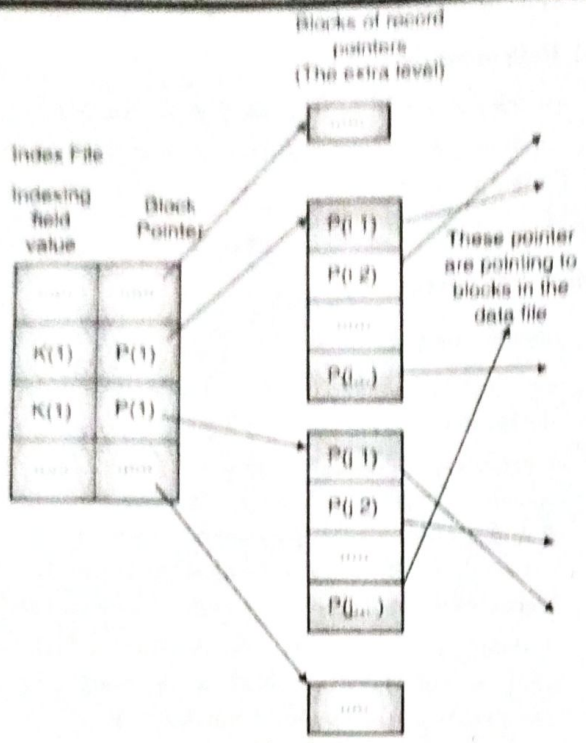


Fig. 6.2

- **The PRIMARY index is stored together with the data in the same table. The clustered index will maintain the order of rows in the table.**
- **The indexes other than the PRIMARY index are called secondary indexes or also known as non-clustered indexes.**
- **The indexing concept may not be useful if table is very small or there is no condition is applied on column in query also is table is updated very frequently.**

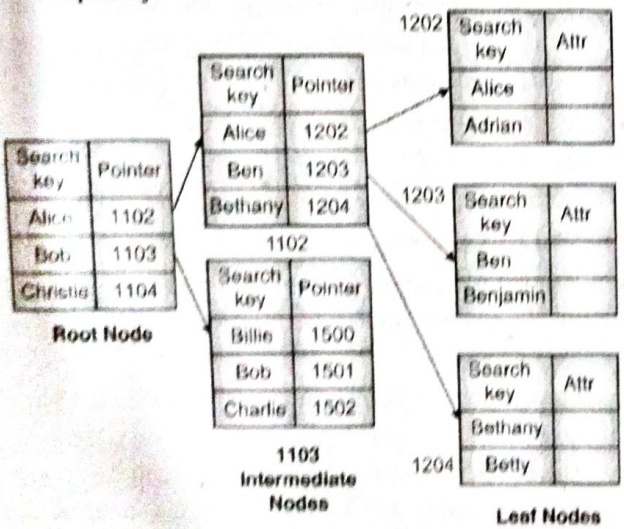


Fig.6.3

4. Performance enhancement by indexing

- The query optimizer may use indexes to locate data at faster speed and without having to scan every row in a table for a given query.
- The index may not be useful if table is very small or there is no condition is applied on column in query also is table is updated frequently.

5. Types of Simple Indexes

(i) Automatic

A unique index is created automatically when you define a PRIMARY KEY or UNIQUE constraint in a CREATE TABLE statement.

(ii) Manual

Users can create non unique indexes on columns to speed up access to the rows.

Syntax (simple Index)

```
CREATE INDEX index
ON table (column[, column]...);
```

Example of Simple Index

- Create index on LAST_NAME column in the EMPLOYEES table.

```
CREATE INDEX emp_last_name_idx
ON employees(last_name);
Index created.
CREATE TABLE Employee
(   Name INT PRIMARY KEY,
    ID INT NOT NULL,
    DEPT INT NOT NULL,
    GROUP VARCHAR(10),
    INDEX (DEPT, GROUP);
);
Table created.
```

- **Create Function based index on LAST_NAME column in the EMPLOYEES table.**

```
CREATE INDEX emp_last_name_idx
ON employees (UPPER(last_name));
Index created.
```

6. Removing Index

- Drop above created index.

```
DROP INDEX upper_last_name_idx;
Index dropped.
```

Chapter - 7 : SQL Queries

Q.1 Explain 'ORDER BY' clause. **(4 Marks)**

Ans. : Order BY Clause :

(1) Introduction

- The ORDER BY keyword is used to sort the result-set by a specified column.
- The ORDER BY keyword sorts the records in ascending order by default.
- If you want to sort the records in a descending order, you can use the DESC keyword.

(2) Syntax

```
SELECT    column_name
FROM      table_name
ORDER BY  column_name ASC|DESC;
```

Example :

- The "Faculty" table

Table 7.1 : Faculty

Faculty_code	Faculty_Name	DOB	Subject	Hours
100	Yogesh	17/07/64	DSA	16
101	Amit	24/12/72	MIS	16
102	Omprakash	03/02/80	PWRC	8
103	Nitin	28/11/66	DT	10
104	Mahesh	01/01/86	DT	10

- Now we want to select all the Faculties from the table above, however, we want to sort the Faculty by their Faculty_Name.
- We use the following SELECT statement :

```
SELECT    *
FROM      Faculty
ORDER BY  Faculty_Name;
```

- In second query ORDER BY 2 indicates order by second column of table.
- The result-set will look like this. (Table 7.2)

Table 7.2

Faculty_code	Faculty_Name	DOB	Subject	Hours
101	Amit	24/12/72	MIS	16
104	Mahesh	01/01/86	DT	10
103	Nitin	28/11/66	DT	10
102	Omprakash	03/02/80	PWRC	8
100	Yogesh	17/07/64	DSA	16

Example :

- Now we want to select all the Faculties from the table above, however, we want to sort the Faculties descending by their Faculty_Name.
- We use the following SELECT statement:

```
SELECT    *
FROM      Faculty
ORDER BY  Faculty_Name DESC;
```

- The result-set will look like (Table 7.2.3)

Table 7.3

Faculty_code	Faculty_Name	DOB	Subject	Hours
100	Yogesh	17/07/64	DSA	16
102	Omprakash	03/02/80	PWRC	8
103	Nitin	28/11/66	DT	10
104	Mahesh	01/01/86	DT	10
101	Amit	24/12/72	MIS	16

Q.2 Describe SET Operations. **(4 Marks)**

Ans. : SET Operations

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

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

- In order to calculate the union, intersection, or difference of two queries, the two queries must be "union compatible", which means that they both

return the same number of columns, and that the corresponding columns have compatible data types.

1. UNION

- Union effectively appends the result of Query_2 to the result of Query_1.
- Furthermore, it eliminates all duplicate rows, in the sense of **DISTINCT**, unless ALL is specified.

DISTINCT : Duplicate row shown only once.

ALL : Duplicate row shown only once.

Example :

```
SELECT    *
FROM      stud ;
```

Id	Name	Semester	Diploma	Gender
1	Fred	2	Bio	M
3	Tom	1	Bio	M
4	John	3	Phy	M
2	Lisa	2	Bio	F

(4 Rows Selected)

```
SELECT    *
FROM      stud_extern ;
```

Id	Name	Semester	Diploma	Gender
3	Tom	1	Bio	M
4	John	3	Phy	M
5	Simon	4	Inf	F

(3 Rows Selected)

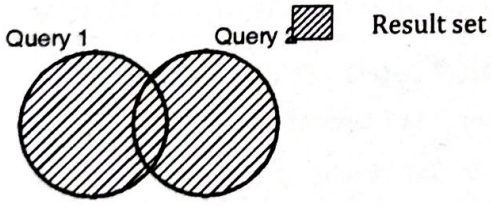


Fig. 7.1 : Union operation

```
SELECT    *
FROM      stud
UNION
SELECT    *
FROM      stud_extern ;
```

Id	Name	Semester	Diploma	Gender
1	Fred	2	Bio	M
2	Lisa	2	bio	F
3	Tom	1	Bio	M
4	John	3	Phy	M
5	Simon	4	Inf	F

(5 Rows Selected)

2. INTERSECT

- Returns all rows that are both in the result of Query_1 and in the result of Query_2. Duplicate rows are eliminated unless **ALL** is specified.

Example :

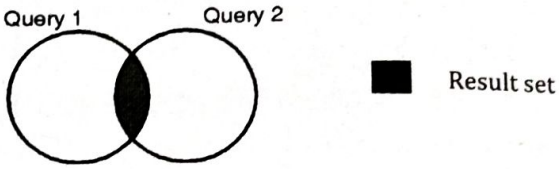


Fig. 7.2 : Intersect operation

```
SELECT    *
FROM      stud
INTERSECT
SELECT    *
FROM      stud_extern;
```

Id	Name	Semester	Diploma	Gender
3	Tom	1	Bio	M
4	John	3	Phy	M

(2 Rows Selected)

3 EXCEPT

- Returns all rows that are in the result of query1 but not in the result of query2.
- Again, duplicates are eliminated unless **ALL** is specified.

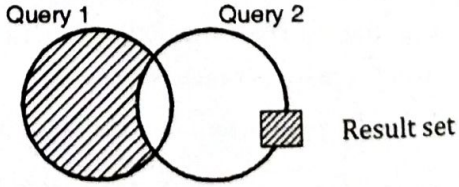


Fig. 7.3 : Except Operation

```
SELECT    *
FROM      stud
EXCEPT
SELECT    *
FROM      stud_extern ;
```

Id	Name	Semester	Diploma	Gender
1	Fred	2	Bio	M
2	Lisa	2	bio	F

(2 Rows Selected)

4 Nesting SET Operations

Set operations can be nested and chained like below Example

```
Query1 UNION query2 UNION query3
```

This really says

```
(Query1 UNION query2) UNION query3
```

Q.3 Write a note on various aggregate function. (4 Marks)

Ans. : Aggregate Functions

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

Types of Aggregate Functions

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

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

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

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

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

Example :

Table 7.4 : Exam_Marks

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

(A) COUNT ()

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

Example :

Find total number of students in above Table 7.4.

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

COUNT
6

Let us consider above table modified as below, (Some duplicate rows are present in Table 7.5).

Table 7.5

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

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

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

COUNT
8

So we can modify query as given below.

Example :

Find total number of diffrent students in above Table 7.5.

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

COUNT
6

(B) SUM ()

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

Example :

Find total of marks scored by all students.

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

SUM
446

(C) AVG ()

- This function is used to calculate Average of column values in a table selected by query.
- This function first calculates sum of column and then divide by total number of rows.
- AVG returns the average of all the values in the specified column.
- Column in the query must be numeric.

Example :

Find avarage marks of students.

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

AVG
89.33

(D) MIN()

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

Example :

Find minimum marks scored by students.

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

MIN
80

(E) MAX()

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

Example :

Find maximum marks scored by students.

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

MAX
80

Q.4 Explain various types of outer join operations with example. **SPPU - Dec. 17, 6 Marks**

Ans. : Outer Join

- In an inner join or in case of a simple join, the resultant table contains only the combinations of rows that satisfy the join conditions. Rows that do not satisfy the join conditions are discarded. Outer join, joins two table although there is no match between two joining tables.
- An outer join makes one of the tables dominant. Such table is called the outer table and other table is called as subordinate table.
- In an outer join, the resultant table contains the combinations of rows from dominant table that satisfy the join conditions and also rows that do not having matching rows in the subordinate table.
- The rows from the dominant table that do not have matching rows in the subordinate table contain NULL values in the columns selected from the subordinate table.
- The syntax for performing an outer join in SQL is DBMS dependent.

Definition

The join that can be used to see rows of table that do not meet the join condition along with rows that satisfies join condition is called as **outer join.**

Following two types of syntax are used generally.

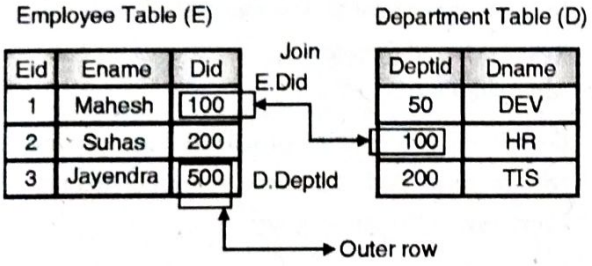
SQL 3 syntax

```
SELECT          Column_List
FROM            Table1
[LEFT/RIGHT/FULL] OUTER JOIN Table2
                ON (JOIN_Condition);
```

(A) Left Outer Join

- In the syntax of a left outer join, the dominant table of the outer join appears to the left of the keyword 'outer join'.
- A left outer join returns all of the rows for which the join condition is true and, in addition, returns all other rows from the dominant table and displays the corresponding values from the subordinate table as NULL.

Example :



Dominant Table (E) Subordinate Table

```
SELECT  E.Eid [Eid],
        E.Did [Did],
        D.Dname [Dname]
FROM    Employee E
        LEFT OUTER JOIN
        Department D
ON      E.Did = D.DeptId;
```

Eid	Did	Dname
1	100	HR
2	200	TIS
3	500	NULL

- The above table returns rows in employees table those having or not having any department. The query below will returns only rows in which employees do not have any department. In this query, the database server applies the filter in the WHERE clause after it performs the outer join on Did column of the Employee and Department tables.

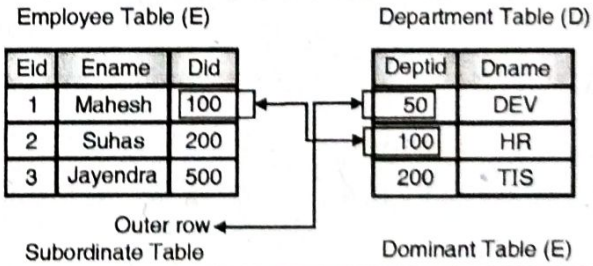
```
SELECT  E.Eid [Eid],
        E.Did [Did],
        D.Dname [DName]
FROM    Employee E
        LEFT OUTER JOIN Department D
ON      E.Did = D. DeptId
WHERE   D.DeptId IS NULL;
```

Eid	Did	DName
3	500	NULL

(B) Right Outer Join

- In the syntax of a right outer join, the dominant table of the outer join appears to the right of the keyword 'outer join'.
- A right outer join returns all of the rows for which the join condition is true and also includes all other rows from the dominant table and displays the corresponding values from the subordinate table as NULL.

Example :



Subordinate Table Dominant Table (E)

```
SELECT  E.Eid [Eid],
        E.Did [Did],
        D.Dname [Dname]
FROM    Employee E
        RIGHT OUTER JOIN Department D
ON      E.Did = D. DeptId;
```

Eid	Did	Dname
1	100	HR
2	200	TIS
NULL	50	DEV

- Above table returns all rows from the dominant table Department and, as necessary, displays the corresponding values from the subordinate table Employee as NULL.

(C) Full Outer Join

- In full outer join both the table acts as dominant tables alternatively.
- A full outer join returns all of the rows for which the join condition is true and also includes,
  (a) All other rows from the right table and displays the corresponding values from the left table as NULL.
  (b) All other rows from the left table and displays the corresponding values from the right table as NULL.

Example :

Employee Table (E)

Eid	Ename	Did
1	Mahesh	100
2	Suhas	200
3	Jayendra	500

Department Table (D)

Deptid	Dname
50	DEV
100	HR
200	TIS

Outer row ← Outer row →

Dominant Table Dominant Table

```
SELECT  E.Eid [Eid],
        E.Did [Did],
        D.Did [Did],
        D.Dname [Dname]
FROM    Employee E
        FULL OUTER JOIN
        Department D
ON      E.Did = D. DeptId;
```

Eid	Did	Dname
1	100	HR
2	200	TIS
3	500	NULL
NULL	50	DEV

Q.5 What do you mean by correlated subquery? SPPU - May 18, 4 Marks

Ans. :

Correlated Sub Queries

- In case of simple subquery, first subquery will be solved then using result of subquery, outer query will be solved
- Simple subquery is evaluated once for each query.
- If the previous request is changed slightly, the subquery in the **HAVING** clause becomes a correlated subquery.
- Correlated subquery is evaluated once for each resultant row in query.
- In case of correlated subquery, first row of main query is found then for this result of main query we solve subquery.
- Correlated sub queries processes data in row-by-row pattern. Each subquery is executed once for every row of the outer query.
- In case of nested queries, we can refer to the table in the FROM clause of the outer query in the inner query using correlated sub-queries.
- We can use a correlated subquery in the HAVING clause.

Syntax :

```
SELECT  column1, column2, ...
FROM    table1 [ref]
WHERE   column1 operator ( SELECT colum1,
column2
                    FROM   table2
                   WHERE  expr1 = ref.expr2);
```

The subquery references a column from a table in the main query.

Example :

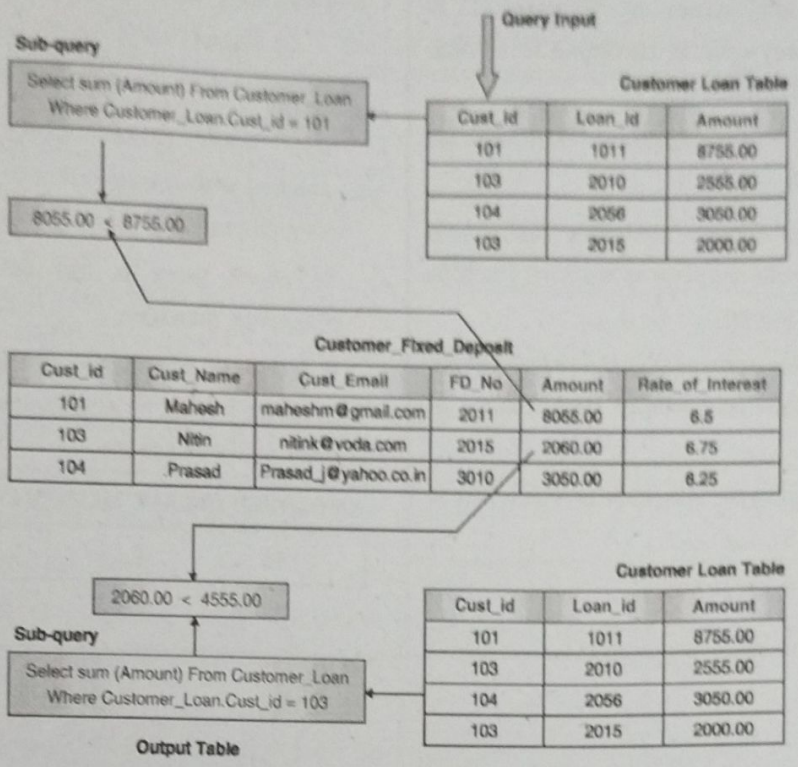


Fig. 7.4

The inner query is executed separately for each row of the outer query. i.e. in correlated sub-queries, SQL performs a sub-query over and over again – once for each row of the main query.

Example :

1. Find customers with fixed deposit less than the sum of all their loans.

```
SELECT    Cust_id, Cust_Name
FROM      Customer_Fixed_Deposit  [CFD]
WHERE     Amount < ( SELECT  sum(Amount)
          FROM    Customer_Loan [CL]
          WHERE  [CL].Cust_id = [CFD].Cust_id );
```

Find all employees who earn less than the average salary in own department.

```
SELECT   Faculty_Id, department
FROM  employees outer
WHERE   salary < ( SELECT  AVG(salary)
          FROM    employees
          WHERE  department = outer.department);
```

2. Find out the employees who have at least one person reporting to them.

```
SELECT     employee_id, last_name
FROM     employees e
WHERE    EXISTS (SELECT   'X'
                 FROM     employee
                 WHERE    manager_id = e.employee_id);
```

TRUE - Print row
FALSE - Do not print that row

- Inner query checks values of 'manager_id' is matched with the any value in 'employee_id' column of employee table (of outer query).
- To check there is any employee working for respective manager.
- If result set of inner query is nonempty depicts that manager having some employees working for him. So, print result on screen.
- The EXISTS operator ensures that the search should not continue further to optimize performance of query.
- To find out the employees who have no one which is reporting to them.

```
SELECT employee_id, last_name
FROM   employees e
WHERE  NOT EXISTS (SELECT 'X'
                   FROM employees
                   WHERE manager_id = e.employee_id);
```

- Inner query checks values of 'manager_id' is matched with the any value in 'employee_id' column of employee table (of outer query).
- To check there is any employee working for respective manager.
- If result set of inner query is empty depicts that manager having no employees working for him. So, print result on screen.

EMPLOYEE_ID	LAST_NAME	MANAGER_ID	SALARY
141	Rajs	142	3500

❑❑❑

Q. 1 What are triggers? Explain trigger with example. SPPU – Dec. 19, 8 Marks

Ans. : Trigger

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

1. Components of Trigger (E-C-A model) – How to Apply Trigger ?

- Trigger is used as ECA Model : Event-Control-Action Model.
- ECA will explain how to use trigger in applicable scenario.
- **Event (E)** - SQL statement that causes the trigger to fire (or activate). This event may be insert, update or delete operation database table.
- **Condition (C)** - A condition that must be satisfied for execution of trigger.
- **Action (A)** - This is code or statement that execute when triggering condition is satisfied and trigger is activated on database table.

2. Trigger Syntax

```
CREATE [OR REPLACE] TRIGGER <Trigger_Name>
[<ENABLE | DISABLE>]
<BEFORE | AFTER>
<INSERT | UPDATE | DELETE>
ON <Table_Name>
   [FOR EACH ROW]
DECLARE
   <Variable_Definitions>;
BEGIN
   <Trigger_Code>;
END;
```

Table 8.1 : Trigger parameter

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

3 Trigger Types

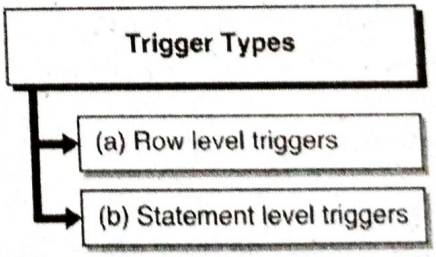


Fig. 8.1 : Trigger types

(a) Row level triggers

- A row level trigger is fired each time the table is affected by the triggering statement.

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

(b) Statement level triggers

- A **statement level trigger** is fired only once on behalf of the triggering statement, irrespective of the number of rows in the table that are affected by the triggering statement
- This trigger executes once even if no rows are affected.
- **Example :** if a DELETE statement deletes several rows from a table, a statement-level DELETE trigger is fired only one time.
- This is Default type, when FOR EACH ROW clause is not written in trigger that means trigger is statement level trigger

4 Trigger Example

- Creating a trigger on employee table whenever new employee added a comment is written to EmpLog Table.
- Let us write a trigger and study its effect.

Example :

```
SQL> CREATE OR REPLACE TRIGGER AutoRecruit
 2   AFTER INSERT ON EMP
 3   FOR EACH ROW
 4   BEGIN
 5       Insert into EmpLog values ('Employee
Inserted');
 6   END;
 7   /
Trigger created.
SQL> INSERT INTO EMP
2 VALUES
3 (1,'Mahesh','Manager','1-JAN-1986',3000,null,10);
1 row created.
SQL> SELECT * FROM EmpLog;
STATUS
----------
Employee Inserted;
```

Consider another example, whenever there comes a new student add him to CS (Computer Science).

Example :

```
SQL>  CREATE TRIGGER CSAutoRecruit
   AFTER INSERT ON Student
   FOR EACH ROW
   BEGIN
   INSERT INTO Take VALUES (111, 'CS');
   END;
```

5 Trigger Operations

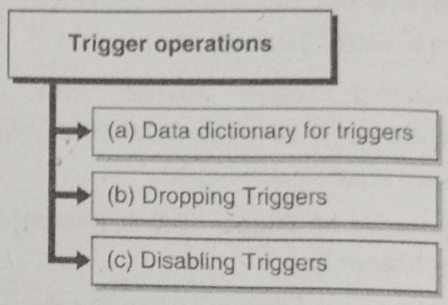


Fig. 8.2 : Triggers operations

(a) Data dictionary for triggers

Once triggers are created their definitions can be viewed by selecting it from system tables as shown below :

Syntax :

```
MySQL>   Select   *
   From   User_Triggers
   Where  Trigger_Name = '<Trigger_Name>';
```

This statement will give you all properties of trigger including trigger code as well.

(b) Dropping Triggers

To remove trigger from database we use command DROP

Syntax :

```
MySQL>Drop trigger <Trigger_Name>;
```

(c) Disabling Triggers

To deactivate trigger temporarily this can be activated again by enabling it.

Syntax :

```
MySQL> Alter trigger <Trigger_Name> {disable | enable};
```

6 Trigger Advantages \ Uses of Trigger

1. Triggers are useful for enforcing referential integrity, which preserves the defined relationships between tables when you add, update, or delete the rows in those tables. Make sure that a column is filled with default information.
2. After finding that the new information is inconsistent with the database, raise an error that will cause the

7 Trigger Disadvantages

A trigger **hampers the performance of system** as database operations will go on slower due to triggering action.

1. Restrictions on triggers

- You cannot modify the same table on which triggering event is written.
- You cannot modify a table which is connected to the triggering table by primary key, foreign key relation.

2. Mutating table errors

- This happens when the trigger is querying or modifying table whose modification activates the trigger, or a table that might need to be updated because of a foreign key constraint with a CASCADE policy.
- This problem is called as **mutating table problem.**
- Error : ORA-04091: table name is mutati trigger/function may not see it.

Chapter - 9 : Programmatic SQL

Q.1 Explain embedded SQL.

SPPU - Dec. 17, Dec.19, 4 Marks

Ans. : Embedded (Static) SQL

1. Introduction

- Embedded SQL is a programming technique that enables you to build SQL statements dynamically at runtime.
- With help of Embedded SQL we can put a SQL statement inside a variable and execute that statement.
- Embedded SQL is a code that is generated programmatically (in part or fully) by your program before it is executed.
- Embedded SQL is a very flexible and powerful tool.

2. Working with various languages

- Embedded SQL is generally build on the client and executed on server machine.
- Embedded SQL in C programming is given As Below,
- If SQL statements are to be embedded within the C program, then there is a need for a mechanism to pass values between the C program environment and the SQL statements that communicate with the Oracle database server.
- Special type of variables like host variables are defined in the embedded program for this purpose.
- We can use the SELECT Statement in embedded SQL of C language as,

```
EXEC SQL begin declare section;
    int sno;
    varchar sname[50];
    varchar city[50];
    char mobile[10];
EXEC SQL end declare section;
```

- We can use select into statement which can be used if it is guaranteed that the query returns exactly one or zero rows.

```
scanf("%d",&cno);
EXEC SQL  SELECT cnameinto :cname
          FROM customers
          WHERE cno= :cno;
printf("%d",cname);
...
```

3. Advantages

- These results in SQL code created are based on the user input. We can generate the query from this input to specify what we want to as output and in what format.
- Query can be changed just by changing input parameters.
- Facilitates the automatic generation and execution of program statements.
- Embedded SQL statements can change from one execution to the next.
- Embedded SQL statements can be written by people with comparatively less programming experience, because the program does most of the actual generation of the code.
- Embedded SQL statements can be built interactively with input from users having little or no knowledge of SQL.

4. Disadvantages

- Embedded SQL statements can be entered by the application programmer or it can be generated by the program itself, they are not embedded in the source program.
- As it works in changing data requirements so it hampers performance of a system.

Q.2 Explain dynamic SQL

SPPU - Dec. 17, Dec.19, 4 Marks

Ans. : Interactive (Dynamic) SQL

Introduction

- The Dynamic SQL statements are used in situations where the SQL statements to be executed in an application program are not known at compile time. These statements are built spontaneously or dynamically as and when program is executing.
- Such, SQL mechanism is called as Dynamic SQL.
- We can use dynamic SQL to accomplish tasks such as adding where clauses to a search based on fields filled on a front end form or to create tables with varying names.

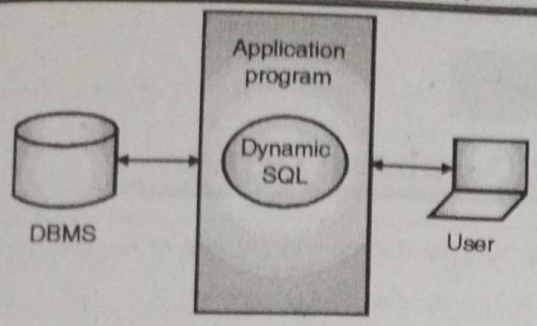


Fig. 9.1 : Working of dynamic SQL

Working with Various Languages

- Embedded SQL is generally build on the client and executed on server machine.
- **Execute Immediate Statement**
- EXEC SQL execute immediate (host-var);
- **Prepare Statement**
- EXEC SQL prepare s from :sql_stmt;
- **Execute Statement**

  EXEC SQL execute s using :var1, ..., :varn;
- **Embedded SQL in ASP programming,**

```
Dim sql
sql = "Select * FROM Employee WHERE EmpID = "
sql = sql&request.querystring("EmpID")
<!-  - Comment : Accepting Employee ID from user --
>
set results = objConn.execute(sql)
```

- The above example if you try to pass query table name as a parameter it will not be allowed in SQL server so we can use some other logic for passing table name in query, as follows

```
Create Procedure TableSelect
@TableNameVarchar(100)
AS
Declare @SQL VarChar(1000)
SELECT @SQL = 'SELECT * FROM '
SELECT @SQL = @SQL + @TableName
- - Comment: Build a SQL query as a string then
execute it
Exec ( @SQL)
- - Comment: Execute constructed SQL query as a
string
```

- Generally stored procedure can not cache the execution plan for this dynamic query. So, for complex queries performance may be hampered.
- The advantage is, of course, that you are able to achieve flexibility in your code that you can not get with standard SQL.

Advantages

- These results in SQL code created are based on the user input. We can generate the query from this input to specify what we want to as output and in what format.
- Query can be changed just by changing input parameters.
- Facilitates the automatic generation and execution of program statements.
- Embedded SQL statements can change from one execution to the next.
- Embedded SQL statements can be written by people with comparatively less programming experience, because the program does most of the actual generation of the code.
- Embedded SQL statements can be built interactively with input from users having little or no knowledge of SQL.

Disadvantages

- Not pre-parsed
- Static SQL is parsed at compile time of SQL statement and then any future executions can use the parsed version which is much more efficient. This is not possible with Dynamic SQL.
- Cannot be checked at compile time
- Dynamic SQL is valid or not we cannot determine it till you execute it. This may be results in a lot of complicated de-bugging issues.
- No dependency check
- Suppose your complex SQL query refers to a column or table that no longer exists. If the query is written as static SQL, then as soon as the column/table was removed the package will also be removed. But if it's dynamic SQL, then you will not know there's a problem until the query is actually executed.

Applications

Sometimes budget doesn't provide sufficient funding to pay for the database work or the staff isn't trained enough in writing Stored Procedures. The project is put into such development.

❑❑❑

Unit IV

Chapter - 10 : Relational Database Design

Q.1 State and explain Armstrong's axioms and its property. **SPPU - May 18, 4 Marks**

Ans. :

FD Properties (Armstrong's Axioms/ Closures of FD)

FD Properties (Armstrong's Axioms/ Closures of FD)

- Given that Relation R(X, Y, Z, W); represents a table R with set of indivisible attributes X, Y, Z and W.
- It is possible to derive many properties of functional dependencies.
- Axioms are nothing but rules of inference which provides a simple technique for reasoning about functional dependencies.

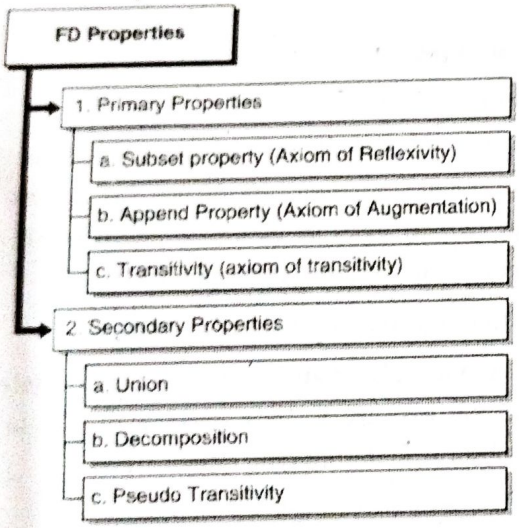
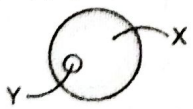


Fig. 10.1 : FD Properties

(1) Primary Properties

a. Subset property (Axiom of Reflexivity)



For given relation R(X, Y, Z, W),

If Y is a subset of X as shown in diagram,

Then **X → Y**

(Which can be referred as X is functionally dependent on Y)

b. Append Property (Axiom of Augmentation)

For given relation R(X, Y, Z, W),

If X → Y

Then XZ → YZ

It is possible to append attribute Z to both sides of FD provided that it is part of same table.

c. Transitivity (axiom of transitivity)

For given relation R(X, Y, Z, W),

If X → Y and Y → Z

Then X → Z

It is possible to use transitivity if attribute X, Y and Z are part of the same table.

(2) Secondary Properties

a. Union

For given relation R(X, Y, Z, W),

If X→ Y and X → Z

Then X → YZ

b. Decomposition

For given relation R(X, Y, Z, W),

If X → YZ

Then X → Y and X → Z

c. Pseudo Transitivity

For given relation R(X, Y, Z, W),

If X → Y and YZ → W

Q.2 What is lossless decomposition? **SPPU - Dec. 17, 4 Marks**

Ans. : Lossless-join Decomposition

- It is clear that decomposition must be lossless so that we do not lose any information from the relation that is decomposed.
- Lossless join decomposition ensures that we can never get the situation where spurious tuple are generated in relation, for every value on the join attributes there will be a unique tuple in one of the relations. For above join to become lossless we need to go for following steps.

Steps :

(i) Let R_1 and R_2 form decomposition of relation R as R_1 and R_2 are both sets of attributes from R.

(ii) Decompose the relation schema Department-Student into

Department-schema = (Dept_Id, Dname)

Student-schema = (Stud_id, Sname, Location)

(iii) The attributes in common must be a key for one of the relation for decomposition to be lossless. $R_1 \cap R_2 \neq \Phi$ There must not be null.

Note : You are joining a primary key and a foreign key of table.

Example :

Above relation can be lossless decomposed as follows,

Schema 1 : Department schema contains (Dept_Id, Dname)

$\therefore R_1 \leftarrow \Pi_{Dept\_Id,Dname}$ (Department_student)

Dept_Id	Dname
10	Development
20	Teaching
30	HR

Schema 2 : Student schema contains (Stud_Id, Dept_Id, Sname, Location)

$\therefore R_2 \leftarrow \Pi_{Stud\_Id, Dept\_Id, Location,Sname}$ (Department_student)

Stud_Id	Dept_Id	Location	Sname
1	10	Mahim	Sushant
2	20	Vashi	Snehal
3	30	Worli	Pratiksha
4	20	Dadar	Supraja

This is lossless-join decomposition as $R_1 \cap R_2 \neq \Phi$ common column is Dept_Id

Q.3 What is Normalization ? SPPU - May 19, 2 Marks

Ans. : Normalization

Introduction

- Normalization is a step by step decomposition of complex records into simple records.
- Normalization is a process of organizing data in database in more efficient form. It results in tables that satisfy some constraints and are represented in a simple manner.
- This process is also called as canonical synthesis.
- Normalization is a step by step decomposition and database designers may not normalize relation to the highest possible normal form.
- The relations may be left in a lower normal form like 2NF, which may cause some penalties like data anomalies.

Definition

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

Q.4 State and explain 2NF. SPPU - May 19, 6 Marks

Ans. : Second Normal Form (2NF)

1. Introduction

- This normal form makes use of full functional dependency and tries to remove problem of redundant data that was introduced by 1NF decomposition.
- Therefore before applying 2NF to a relation, it needs to satisfy 1NF condition.

2. Definition

- A relation is in 2NF, if it is in 1NF and all non-key attributes in relation are fully functionally dependent on the primary key of the relation.

OR

- A relation is in 2NF, if it is in 1NF and every non-key attribute is fully functionally dependent on the complete primary key of relation (and not depends on part of (partial) primary key).
  - In short 2NF means,
  - It should be in 1NF.
- There should not be any partial dependency on primary key attributes.
- 2NF prohibits partial dependencies

3. **Steps**

(a) Find and remove attributes that are related to only a part of the key or not related to key.

(b) Group the removed attributes in another table.

(c) Assign the new table a key that consists of that part of the old composite key.

(d) If a relation is not in 2NF, it can be further normalized into a number of 2NF relations.

4. **Example:**

- Consider an employee table with columns as shown in diagram,
- The relational schema **not in 2 NF** is represented as,

Consider an Employee table with following FDs,

Employee_Id → Ename, Salary

Employee_Id, Project_Id → Hours, Allowance

As

{Employee_Id, Project_Id} → Ename, Salary, Hours, Allowance

Therefore,

Candidate key {Employee_Id, Project_Id} is selected as primary key.

- As attributes Hours, Allowance of employee table are full functionally dependent on primary key whereas attributes Ename and Salary are partially depends on primary key. (As Ename, Salary are depends **on part of primary key**)
- The state of Employee relational schema is,

Table 10.2 : Non-2NF Employee Table

Eid	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000
15	Ganesh	26000	E001	24	20000
18	Mahesh	50000	B056	11	10000

- To normalize above schema to 2NF we can decompose tables as,
- Employee (Employee_Id, Ename, Salary)
- Employee_Id → Ename, Salary

Table 10.3 : 2NF Employee Table

Employee_Id	Ename	Salary
10	Mahesh	50000
12	Suresh	25000
15	Ganesh	26000
18	Mahesh	50000

- Project (Employee_Id, Project_Id, Hours, Allowance)
- Employee_Id, Project_Id → Hours, Allowance

Table 10.4 : 2NF Project Table

Employee_Id	Project_Id	Hours	Allowance
10	E001	44	40000
12	B056	31	30000
15	C671	23	20000
18	E002	12	15000
15	E001	24	20000
18	B056	11	10000

- Consider, Relation R(A, B, C, D, E, F) and the FDs as below,

A → BC,B→ DC,D → EF

(i) The candidate Key is {AD} → {A, D, B, C, E, F} selected as primary key.

All attributes are partially dependent on primary key.

Hence, Relation R is **not in 2NF**.

(ii) The 2NF Relation Schema is,

R1 (A, B, C, D) with FDs A → BC, B → DC

R2 (D, E, F) with FDs D → EF

5. **Minimizing Tuple Redundancy**

- The second normal form will avoid same tuples to be repeated in a table as it forces all non-key attributes must be full functionally depends on primary key of a relation.
- 2NF will create a new table for each partial key with all its dependent attributes.

Example:

- Let us consider the table we obtained after first normalization.

Sr. No.	Faculty code	Faculty name	Date of birth	Subject	Hours
1	100	Yogesh	17/07/64	DSA	16
2	100	Yogesh	17/07/64	SS	8
3	100	Yogesh	17/07/64	IS	12
4	101	Amit	24/12/72	MIS	16
5	101	Amit	24/12/72	PM	8
6	101	Amit	24/12/72	IS	12
7	102	Omprakash	03/02/80	PWRC	8
8	102	Omprakash	03/02/80	PCOM	8
9	102	Omprakash	03/02/80	IP	16
10	103	Nitin	28/11/66	DT	10
11	103	Nitin	28/11/66	PCOM	8
12	103	Nitin	28/11/66	SS	8
13	104	Mahesh	01/01/86	DT	10
14	104	Mahesh	01/01/86	ADBMS	8
15	104	Mahesh	01/01/86	PWRC	8

- While eliminating the repeating groups, we have introduced redundancy into table. Faculty code, Name and Date of birth are repeated since the same faculty is multi skilled.
- To eliminate this, let us split the table into 2 parts; one with the non-repeating groups and the other for repeating groups.

Faculty

Faculty code	Faculty Name	Date of birth
100	Yogesh	17/07/64
101	Amit	24/12/72
102	Omprakash	03/02/80
103	Nitin	28/11/66
104	Mahesh	01/01/86

Faculty_code → Faculty_name, Date_of_Birth

The other table is those with repeating groups.

Subject

Table 10.5

Sr. No	Faculty code	Subject	Hours
1	100	DSA	16
2	100	SS	8
3	100	IS	12
4	101	MIS	16
5	101	PM	8
6	101	IS	12
7	102	PWRC	8
8	102	PCOM	8
9	102	IP	16
10	103	DT	10
11	103	PCOM	8
12	103	SS	8
13	104	DT	10
14	104	ADBMS	8
15	104	PWRC	8

- Faculty code is the only key to identify the faculty name and the date of birth. Hence, Faculty code is the primary key in the first table and foreign key in the second table.
- Faculty code is repeated in the Subject table. Hence, we have to take into account the 'SNO' to form a composite key in Subject table. Now, SNO + Faculty code can unique identity each row in this Table 10.4.

Hence, the relation is now in Second Normal form.

6. Anomalies (Problems)

(a) Insertion

- New record needed to insert in both tables.
- Inserting the records of various Faculties teaching same subject would result the redundancy of hours information.

(b) Updation

- Updating some record in faculty table may exist in subject table.

- For a subject, the number of hours allotted to a subject is repeated several times. Hence, if the number of hours has to be changed, this change will have to be recorded in every instance of that subject. Any omissions will lead to inconsistencies.

(c) Deletion

- If a faculty leaves the organization, information regarding hours allotted to the subject is also needed to be deleted from subject table.
- Hence, This Subject table should therefore be further decomposed without any loss of information in 3rd normal form.

Q.5 State and explain 3 NF. **SPPU - May 19**

Ans. : Third Normal Form (3NF)

1. Introduction

- This normal form used to minimize the transitive redundancy.
- In order to remove the anomalies that arose in Second Normal Form and to remove transitive dependencies, if any, we have to perform third normalization.

2. Definition

- A relation is in 3NF, if it is in 2NF and no non-key attribute of the relation is transitively dependent on the primary key.
- 3NF prohibits transitive dependencies.
- In short 2NF means,
  1. It should be in 2 NF.
  2. There should not be any transitive partial dependency.

3. Example:

Now let us see how to normalize the second table obtained after 2NF.

Subject

Table 10.5

Sr. No	Faculty code	Subject	Hours
1	100	DSA	16
2	100	SS	8
3	100	IS	12
4	101	MIS	16
5	101	PM	8
6	101	IS	12
7	102	PWRC	8
8	102	PCOM	8
9	102	IP	16
10	103	DT	10
11	103	PCOM	8
12	103	SS	8
13	104	DT	10
14	104	ADBMS	8
15	104	PWRC	8

- In this Table 10.5, hours depend on the subject and subject depends on the Faculty code and Sr. No.
- But, hours is neither dependent on the faculty code nor the SNO. Hence, there exists a transitive dependency between Sr. No., Subject and Hours.
- If a faculty code is deleted, due to transitive dependency, information regarding the subject and hours allotted to it will be lost.
- For a table to be in 3rd Normal form, transitive dependencies must be eliminated.
- So, we need to decompose the table further to normalize it.

Fac_Sub

Sr. No	Faculty code	Subject
1	100	DSA
2	100	SS
3	100	IS
4	101	MIS
5	101	PM
6	101	IS
7	102	PWRC
8	102	PCOM
9	102	IP
10	103	DT
11	103	PCOM
12	103	SS
13	104	DT
14	104	ADBMS
15	104	PWRC

Sub_Hrs

Subject	Hours
DSA	16
SS	8
IS	12
MIS	16
PM	8
PWRC	8
PCOM	8
IP	16
DT	10
ADBMS	8

- After decomposing the 'Subject' table we now have 'Fac_Sub' and 'Sub_Hrs' table respectively.

4. Advantages

(i) Insertion

No redundancy of data for subject and hours while inserting the records.

(ii) Updation

- Subject and hours are stored in the separate table.
- So updation becomes much easier as there is no repetitiveness of data.

(iii) Deletion

Even if the faculty leaves the organization, the hours allotted to a particular subject can be still retrieved from the Sub_Hrs table.

Q.6 Specify codd's Norms to be satisfied by RDBMS?

SPPU - May 19, 4 Marks

Ans. : Boyce Codd Normal Form (BCNF)

1. Introduction

- This normal form is governed by Raymond F. Boyce and E.F. Codd (1974)
- BCNF is more rigorous form of 3NF.
- The intention of Boyce-Codd Normal Form (BCNF) is that 3NF does not satisfactorily handle the case of a relation processing two or more composite or overlapping candidate keys.
- **Candidate key** is a column in a table which has the ability to become a primary key.
- **A determinant** is any attribute (simple or composite) on which some other attribute is fully functionally dependent.

  $a \rightarrow b$

  Then, attribute 'a' is determinant.

2. Definition

A relation R is said to be in BCNF, if and only if every determinant is a candidate key.

3. Sample relations

ID	Ename	Qualification	Grade	Did	DName
1	Satish	B.E.	C	20	EX
2	Savita	M.E.	B	30	CE
3	Mahesh	Ph.D.	A	10	IT

- Suppose Grade of faculty depends on his qualification. This relation has 3 determinants as ID, Qualification and Did. But if only (ID, Did) is candidates key then relation may not be in BCNF. For relation to be in BCNF every determinant must be candidates key.
- For table,

ID, Qualification $\rightarrow$ Grade

ID $\rightarrow$ Enami, Did

Did $\rightarrow$ DName

ID	Ename	Did
1	Satish	20
2	Savita	30
3	Mahesh	10

- This relation has only one determinant as ID and it is also a candidate's key then relation is in BCNF.

ID	Qualification	Grade
1	B.E.	C
2	M.E.	B
3	Ph.D.	A

- This relation has determinants as ID and Qualification they are in candidates key then relation is in BCNF.

Example :

- Soldiers are part of one or many units, and each unit is under the control of an officer.

SOLDIERID	OFFICERID	UNITID
1	A	1
2	A	1
3	B	2

- Firstly, we'll identify the dependencies.
- There is a dependency between (SOLDERID + OFFICERID) and UNITID, a soldier and an officer implies their respective unit, but there is also a dependency between UNITID and OFFICERID.

  SOLDERID → UNITID

  UNIT ID → OFFICEID

  SOLDIERID, OFFICEID → UNITID
- This last dependency however is not partial (dependence on part of a prime attribute), nor transitive (dependence of a nonprime attribute on another nonprime attribute, and OFFICERID is a prime attribute).
- What we have is a table where a determinate in the table is not a candidate key (UNITID).
- Candidate key are SOLDERID and OFFICEID.
- We can convert the above to BCNF by realizing that a better composite key is one of SOLDERIERID and UNITID, which creates a dependency between UNITID and OFFICERID, which is a partial dependency.
- This is then resolved by dividing the table, the solution being as follow :

  Candidate key (SOLDERID, SOLDERID) → UNITID

SOLDIERID	UNITID
1	1
2	1
3	2

  Candidate key (UNIT ID) AND UNIT ID → OFFICEID

UNITID	OFFICERID
1	A
1	A
2	B

  The above table is now in BCNF.

Fourth Normal Form (4NF)

1. Introduction

This normal form is given by Ronald Fagin (1977).

- Fourth normal form tries to remove multi valued dependency among attributes.

2. Definition

A relation is said to be in **fourth normal form** if each table contains no more than one multivalued dependency per key attribute.

3. Example :

Seminar	Faculty	Topic
DBP-1	Brown	Database Principles
DAT-2	Brown	Database Advanced Techniques
DBP-1	Brown	Data Modeling Techniques
DBP-1	Robert	Database Principles
DBP-1	Robert	Data Modeling Techniques
DAT-2	Maria	Database Advanced Techniques

- In the above example, same topic is being taught in a seminar by more than 1 faculty and each Faculty takes up different topics in the same seminar.
- Hence, Topic names are being repeated several times.
- This is an example of multivalued dependency. (No multivalued dependency)

Seminar	Topic
DBP-1	Database Principles
DAT-2	Database Advanced Techniques
DBP-1	Data Modeling Techniques

- To eliminate multivalued dependency, split the table such that there is no multivalued dependency. (One multivalued dependency).

Seminar	Faculty
DBP-1	Brown
DAT-2	Brown
DBP-1	Robert
DAT-2	Maria

Chapter - 11 : Query Processing

Q.1 Define query processing. **SPPU - May 18, 4 Marks**

Ans. : Introduction to Query Processing

(1) Relational query processing refers to the range of activities which includes in extracting data from a database using a database query.

(2) A query processing involves below steps,

1. Scan 2. Parse 3. Validate

1. **Scan :** The scanner reads the language tokens such as SQL keywords, relation names in the text of the query.
2. **Parse :** The Parser check the query syntax to verify it is as per the syntax rules of the query language.
3. **Validate :** The query must also be validated by checking that all attributes and relation names are valid in the schema of the particular database being queried by query.

(3) Query Execution Plan

- Query execution plan will give idea about how query will be executing in stepwise manner.
- An internal representation of the query can be created as a tree data structure which is called a **query tree.**
- The DBMS must find all alternative execution strategies for retrieving the result of the query from the database.

Example : Department information can be accessed in following ways,

1. Department table
2. Employee_Department_Join table

- A query can have many possible execution strategies.
- Process of selecting a suitable strategy for processing a query is known as query optimization.
- Query optimization is achievable for simple queries but it becomes very complex for difficult queries.

(4) Each DBMS has a number of general database access algorithms that such as SELECTION, PROJECTION or JOIN or combination of these operations.

Q.2 What are the measures of Query cost ? **(SPPU - Dec. 18, 4 Marks)**

Ans. : Measures of a Query Cost

(1) Query Cost

- Query cost do mean by the predicted execution time required for query execution.
- The cost is the time spent on query execution, plus the CPU time required, plus all other types of time required for query execution.

(2) Measures Used for Query Cost

The query cost can be measured with help of following factors :

a. Access cost to secondary storage

This is the cost for operations searching, reading, and writingdata blocks that reside in secondary storage or disk. Factorssuch as whether the contiguous allocation used to store block on the same disk or it is scattered on the disk which affect the access cost.

b. Storage cost

This is the cost for storing intermediate files which is generated byan execution strategy for the query.

c. Computation cost

This is the cost of memory operations during query execution or it is CPU time to execute query.Such operation includes searching and sortingrecords, merging records for a join or performing computations on various field values.

d. Memory usage cost

This is the cost which shows the number of memory buffersrequired during query execution.

e. Communication cost

This is the cost of sending the query results from the one database site to the other site. (In case of a distributed or parallel database).

f. **Number of different resources** used like printer, disk accesses etc.

g. **Data transfer rate**

h. **Cost of scanning disk segment** containing tuples.

i. **Cost models for different index** access methods (tree structures-hashing).

(3) Other measures used for Query Cost

- The response time for a query evaluation plan can be act like a good measure of the cost of the query evaluation plan.
- In large DBMS number of blocks transferred from disk in unit time is usually the most important cost.
- Block transfer from **disk** is slow as compared to in **memoryoperations**. Hence, disk access cost is important measure of the cost of a query-evaluation plan.
- Estimating the CPU time is difficult as compared to estimating the disk access cost.
- We need to consider differences between following factors also;
  - Variance between rotational latency and seek time
  - Rotational latency is waiting time for the desired data to roll under the read write head and seek time is time that it takes to move the head on to the desired data.
  - Distinguish between Sequential I/O and random I/O.
  - **Sequential I/O** is when the blocks that we want to read are contiguous on disk memory while **random I/O** is when the blocks are noncontiguous or random.
  - Distinguish between Read and write operations
  - Time required to write a block of data to disk is more than to time required to read a block of data from disk.
- So more accurate measure to estimate will be,
  1. The number of seek operations
  2. The number of blocks Read
  3. The number of blocks written
- Above cost estimates generally ignores the cost of writing the final result of an operation back on to the disk.
- The costs of all the algorithms used depend on the size of the buffer present in main memory.

  Best case
  - Entire data can be read into the buffers and once it is loaded no need to access disk again.
  - Worst case
  - Buffer can hold only one or few blocks per relation.
- Generally, for estimating cost we assume the worst case.

(4) Example :

Measuring Query cost in SQL Server 2000

```
SELECT  A.AU_LNAME [AUTHOR],T.TITLE [TITLE]
FROM    AUTHORS A,TITLEAUTHOR L,  TITLES T
WHERE   L.AU_ID    = A.AU_ID
AND     T.TITLE_ID = L.TITLE_ID;
```

The above query will we join three different tables and extract data from those tables. So, with help of SQL server we can look at various cost of each table access operation (Read Operation) individually. (Shown in below diagram)

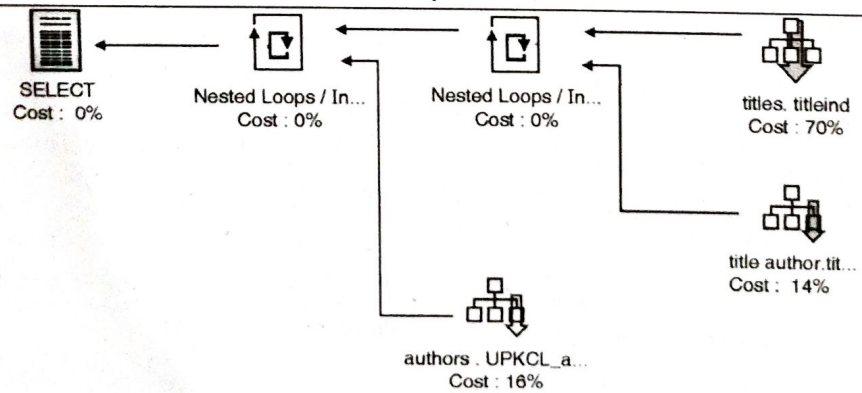


Fig. 11.1 : Sample Query Cost in SQL Server 2000

Unit V

Chapter - 12 : Transaction Management

Q.1 What is transaction. **SPPU- Oct.19, 6 Marks**

Ans. : Transaction

1. Introduction

- Single SQL command is sent to database server as a query and server will reply with answer.
- Multiple SQL commands (DML,DRL etc.) are sent to database server which executed one after other (as shown in Fig. 12.1)

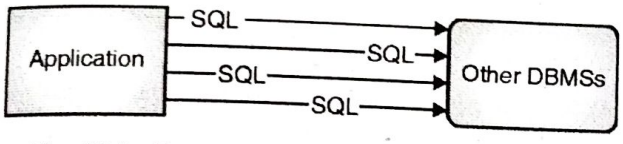


Fig. 12.1 : Executing Single Operation in DBMS

- In place of sending one by one SQL command to server we can combine multiple operations that are logically similar and send to server as a single logical unit called **transaction**.

Example :

Transferring Rs.100 from one account to other

1. Withdraw Rs.100 from account_1
2. Deposit Rs.100 to account_2

- Simple query fired on DBMS is called **SQL operation**.
- Collection of multiple operations that forms a single logical unit is called as **transaction**.
- A transaction is a sequence of one or more SQL statements that combined together to form a single logical unit of work.

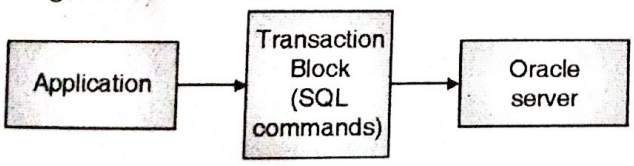


Fig. 12.2 : Executing Transaction in DBMS

- Types of operations that can be done inside transaction

(a) Read operation :

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

Example :

Data Selection / Retrieval Language

```
SELECT  *
FROM    Students;
```

(b) Write operation :

- Write operation transfer data from local memory to database.
- Write operation transfers one data item from the local buffer of the transaction that executed the write back to the database.

Example :

Data manipulation language (DML)

```
UPDATE Student
SET    Name = 'Bhavna'
Where  Sid  = 1186;
```

- Information processing in DBMS divides operation individual, indivisible operational logical units, called transactions.
- A transaction is a sequence of small database operations.
- Transactions will execute to complete all set of operations successfully.

Example :

During the transfer of money between two bank accounts it is problematic if one of transaction fails.

```
BEGIN TRANSACTION transfer
    UPDATE accounts
    SET balance=balance - 100
    WHERE account=A

    UPDATE accounts
    SET balance=balance + 100
    WHERE account=B
if no errors then
        Commit Transaction
else
        Rollback Transaction
End If
END TRANSACTION transfer
```

Fig. 12.3 : Sample Implementation of Transaction in SQL

2. Transaction structure

- The transaction consists of all SQL operations executed between the begin transaction and end transaction.
- A transaction starts with a BEGIN transaction command.
- BEGIN command instructs transaction monitor to starts monitoring the transaction status.
- All operations done after a BEGIN command is treated as a single large operation.

3. Transaction boundaries

- Transaction must ends either by executing a COMMIT or ROLLBACK command.
- Until a transaction commits or rolls back the data in database remains unchanged.

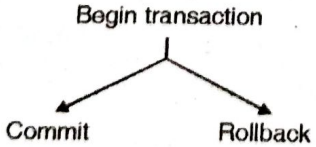


Fig. 12.4

(a) Commit transaction

All successful transactions are required to be committed by issuing commit command. To make all changes permanent and available to other users of the database.

(b) Rollback transaction

- If transaction is unsuccessful due to some error then it must be rolled back.
- Roll back command will remove all changes to the database which are undone and the database remains unchanged by effect of transaction.
- The DBMS should take care of transaction it should be either complete or fail.
- When a transaction fails all its operations and the database is returned to the original state it was in before the transaction started.

Q.2 What are the possible causes of transaction failure ?

SPPU - May 19, 4 Marks

Ans. : Transaction Failure :

- If transaction complete successfully it may be saved on to database server called as **committed** transaction.
- A committed transaction transfers database from one consistent state to other consistent state, which must persist even if there is a system failure.
- Transaction may not always complete its execution successfully. Such a transaction must be **aborted**.
- An aborted transaction must not have any change that the aborted transaction made to the database. Such changes must be undone or **rolled back**.
- Once a transaction is committed, we cannot undo its effects by aborting it.

(A) Transaction States

A transaction must be in one of the following states :

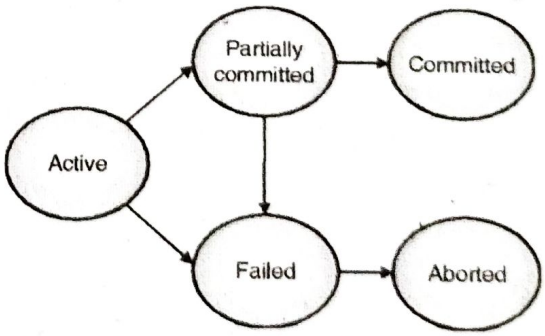


Fig. 12.5 : State diagram of a transaction

(a) Active

- This is initial state of transaction execution.
- As soon as transaction execution starts it is in active state.
- Transaction remains in this state till transaction finishes.

```
BEGIN TRANSACTION transfer
    UPDATE accounts
    SET balance=balance - 100
    WHERE account=A

    UPDATE accounts
    SET balance=balance + 100
    WHERE account=B
if no errors then
    Commit Transaction
else
    Rollback Transaction
End If
END TRANSACTION transfer
```

Fig. 12.3 : Sample Implementation of Transaction in SQL

2. Transaction structure

- The transaction consists of all SQL operations executed between the begin transaction and end transaction.
- A transaction starts with a BEGIN transaction command.
- BEGIN command instructs transaction monitor to starts monitoring the transaction status.
- All operations done after a BEGIN command is treated as a single large operation.

3. Transaction boundaries

- Transaction must ends either by executing a COMMIT or ROLLBACK command.
- Until a transaction commits or rolls back the data in database remains unchanged.

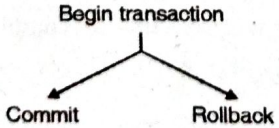


Fig. 12.4

(b) Rollback transaction

- If transaction is unsuccessful due to some error then it must be rolled back.
- Roll back command will remove all changes to the database which are undone and the database remains unchanged by effect of transaction.
- The DBMS should take care of transaction it should be either complete or fail.
- When a transaction fails all its operations and the database is returned to the original state it was in before the transaction started.

Q.2 What are the possible causes of transaction failure ?

SPPU - May 19, 4 Marks

Ans. : Transaction Failure :

- If transaction complete successfully it may be saved on to database server called as **committed** transaction.
- A committed transaction transfers database from one consistent state to other consistent state, which must persist even if there is a system failure.
- Transaction may not always complete its execution successfully. Such a transaction must be **aborted**.
- An aborted transaction must not have any change that the aborted transaction made to the database. Such changes must be undone or **rolled back**.
- Once a transaction is committed, we cannot undo its effects by aborting it.

(A) Transaction States

A transaction must be in one of the following states :

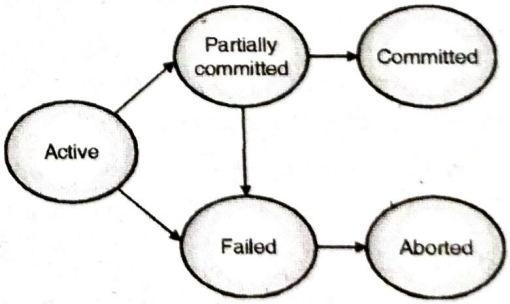


Fig. 12.5 : State diagram of a transaction

(a) Active

(b) Partially committed

- As soon as last operation in transaction is executed transaction goes to partially committed state.
- At this condition the transaction has completed its execution and ready to commit on database server. But, it is still possible that it may be aborted, as the actual output may be there in main memory, and thus a hardware failure may prohibit its successful completion.

(c) Failed

A transaction enters the failed state after the system determines that the transaction can no longer proceed with its normal execution.

Example : In case of Hardware or logical errors occurs while execution.

(d) Aborted

- Failed transaction must be rolled back. Then it enters the aborted state.
- Transaction has been rolled back restoring into prior state.
- In this stage system have two options :

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

(2) Kill the transaction : Because the bad input or because the desired data were not present in the database an error can occurs. In this case we can kill transaction to recover from failure.

(e) Committed

- When last data item is written out, the transaction enters into the committed state.
- This state occurs after successful completion of transaction. A transaction is said to have terminated if has either committed or aborted.

Q.3 Define conflict serializable schedule with example. **SPPU - Oct.19**

Ans. : Conflict Serializability

1. Introduction

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

2. Conflict

- A pair of consecutive database actions (reads, writes) is in conflict if changing their order would change the result of at least one of the transactions.

		Transaction T_j	
		Read(D)	**Write(D)**
Transaction T_i	**Read(D)**	No Conflict	Conflict
	Write(D)	Conflict	Conflict

- Consider schedule S has two consecutive instructions I_i and I_j from transactions T_i and T_j respectively.
- If I_i and I_j access to different data items then they will not conflict and can be swapped, without any problem.
- If I_i and I_j access to same data item D then consider following consequences:
  - I_i = READ (D), I_j = READ (D) then **no conflict** as they only read value.
  - This operation is called as non conflicting swap.
  - I_i = READ (D), I_j = WRITE (D) then they **conflict** and can not be swapped.
  - I_i = WRITE (D), I_j = READ (D) then they **conflict** and can not be swapped.
  - I_i = WRITE (D), I_j = WRITE (D) then they **conflict** and cannot be swapped.
- So we can say that instructions conflict if both consecutive instructions operate on same data item and from different transactions and one of them is WRITE operation.
- If I_i and I_j **access to different data item D** then consider following all consequences **no conflict** as they only read or writing different values.
- I_i = READ (D)/WRITE (D), I_j = READ (P) /WRITE (P) then **no conflict** as they only reading or writing different data.
- The following set of actions is conflicting :

  T_1:R(X), T_2:W(X), T_3:W(X)
- While the following sets of actions are not :

  T_1:R(X), T_2:R(X), T_3:R(X)

  T_1:R(X), T_2:W(Y), T_3:R(X)

3. **Conflict equivalence**

Two schedules are conflict equivalent if they can be turned into one another by a sequence of non conflicting swaps of adjacent actions.

4. **Example :**

- A schedule is conflict serializable if it is conflict equivalent to a serial schedule.

T_1	T_2
Read(P)	
Write(P)	
	Read(P)
Read(Q)	
	Write(P)
Write(Q)	
	Read(Q)
	Write(Q)

Fig. 12.6 : Schedule S

- Now the schedule S after performing swapping can transformed into schedule R as shown above which also results in same values of P and Q.
- The above schedule is same as serial schedule $<T_1, T_2>$.
- If a schedule S can be transformed into a schedule R by a series of swap operations on non conflicting instructions, then we can say **schedule S and R are conflict equivalent**.
- If a concurrent schedule is conflict equivalent to a serial schedule of same transactions then it is conflict Serializable. So schedule **S is conflict serializable** to serial schedule $<T_1, T_2>$.

Q.4 When are two schedules said to be view equivalent?

SPPU - May 18, Dec. 18, 4 Marks

Ans. : View Serializability

1. Introduction

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

2. Conditions for view equivalence

Let, D = Data item

S_1, S_2 = Transaction schedules

T_i, T_j = Database transaction

- Schedules S_1 and S_2 are view equivalent if they satisfy following conditions for each data item D,
  (a) S_1 and S_2must have same transactions included and also they are performing same operations on same data. If T_i reads initial value of D in S_1, then T_i also reads initial value of D in S_2.
  (b) If T_i reads value of D written by T_j in S_1, then T_i also reads value of D written by T_j in S_2.
  (c) If T_i writes final value of D in S_1, then T_i also writes final value of D in S_2.
- First 2 conditions ensure that transaction reads same value in both schedules.
- Condition 3 ensures that final consistent state.
- If a concurrent schedule is view equivalent to a serial schedule of same transactions then it is **view serializable.**
- Consider following schedule S_1 with concurrent transactions $<T_1, T_2, T_3>$.
- In both the schedules S_1 and a serial schedule $S_2<T_1, T_2, T_3>$ T_1 reads initial value of D. Transaction T_3 writes final value of D. So schedule S_1 satisfies all three conditions and is view serializable to $<T_1, T_2, T_3>$.

3. Example :

Below two schedules S and R are view equivalent,

Schedule S		
T_1	T_2	T_3
Read(P)		
	Write (P)	
Write (P)		
		Write (P)
Schedule R		
T_1	T_2	T_3
Read(P)		
Write (P)		
	Write (P)	
		Write (P)

4. Note

- Every conflict serializable schedule is view serializable but not vice versa.
- In above example T_2 and T_3Writes data without reading value of data item by themselves so they are called as "Blind Writes".

2. **Example**

Consider two transactions given below :

T1 : This transaction first read and then write data on data item X, then after that performs read and write on data item Y.

T1
Read (X)
Write (X)
Read (Y)
Write (Y)

T2 : This transaction first read and then write data on data item Y, then after that performs read and write on data item X.

T2
Read (Y)
Write (Y)
Read (X)
Write (X)

Consider above two transactions are executing using locking protocol as below :

T1	T2	
Lock-X(X)		
Read (X)		
Write (X)		
	Lock-X(Y)	
	Read (Y)	
	Write (Y)	
Lock-X(Y)		Wait for transaction T2 to Unlock Y
Read (Y)		
Write (Y)		
	Lock-X(X)	Wait for transaction T1 to Unlock X
	Read (X)	
	Write (X)	

So in above schedule consider two transactions given below transaction T1 is waiting for transaction T2 to Unlock data item Y and transaction T2 is waiting for transaction T1 to Unlock data item X.

So system is in a deadlock state as there are set of transactions T1and T2 such that, both are waiting for each other transaction to complete. This state is called as deadlock state.

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

(a) Deadlock prevention

We can use deadlock prevention techniques to ensure that the system will never enter a deadlock state.

(b) Deadlock detection and recovery

We can allow the system to enter a deadlock state and then we can detect such state by deadlock detection techniques and try to recover from deadlock state by using and deadlock recovery scheme.

4. Both of above methods may result in transaction rollback.
5. Deadlock Prevention is commonly used if the probability that the deadlock occurs in system is high else detection and recovery are more efficient if probability of deadlock occurrence is less.

Approaches for Deadlock Prevention

1. **Approach 1**

A simplest form, in which a transaction acquires lock on all data items which will be required by transaction at the start of execution. It is effective as other transactions can't hold lock on those data items till first unlocks data item.

Disadvantages

(i) It is difficult to know in advance which data items need to be locked.

(ii) Data utilization is very low as may be unused data items locked by transaction.

2. **Approach 2**

This approach uses timestamp ordering of data items. It is something like tree protocol and every transaction have to access data item in given sequence only. The variation of this approach with

two phase protocol assures deadlock prevention. Order of data items must be known to every transaction.

(i) To have concurrency control, two phase locking protocol is used which will ensure locks are requested in right order.

(ii) Timestamp ordering is determined by validation (Ti) i.e. TS (Ti) = validation (Ti) to achieve serializability.

(iii) The validation test for transaction Tj requires that for all transaction Ti with TS (Ti) < TS (Tj) then one of the following conditions must be hold.

(a) Finish (Ti) <start (Tj) as Ti finishes before Tj starts the serializability should be maintained.

(b) Ti completes its write phase before Tj starts its validation phase (Start (Tj) < finish (Ti) <Validation (Tj)).

(iv) This ensures writes of both transactions do not take place at same time.

(v) Read of Tj not affected by writes of Tj and Tj can't affect read of Ti so serializability is maintained.

3. Approach 3 : Prevention and transaction rollbacks

(i) Pre-emptive Technique

(ii) Pre-emption means, if transaction Ti wants to hold lock on data item hold by Tj. then system may preempt (UNLOCK all previous locks) Ti by rolling it back and granting lock to Tj on that data item.

(iii) To control Pre-Emption, Transaction can be assigned a unique timestamp.

(iv) System will use this timestamp to decide whether to wait or rollback transaction.

(v) The transaction keeps its old time stamp if it is rollback and restarted.

(vi) Various deadlock prevention techniques using timestamps,

1. Wait-Die

- This is Non pre-emptive technique of deadlock prevention.
- When transaction Ti wants to hold data item, currently hold by Tj, then Ti is allowed to wait if and only if it is older than Tj otherwise Ti is rolled back (die).
- If T1 requests for data item hold by T2 then as TS (T1) < TS (T2) so T1 should wait.

2. Wound-Wait

- This is Pre-emptive Technique of deadlock prevention.
- When Ti wants to hold data item, currently hold by Tj, then Ti is allowed to wait if and only if Ti is younger to Tj otherwise Tj is rollback (wounded).
- Consider T1, T2, T3 transactions.
- If T1 requests for data item hold by T2 then data item is pre-empted from T2 and given to T1 and T2 is rollback.
- If T3 requests for data item hold by T2 then T3 need to roll back.

(vii) Prevention may lead to starvation of transaction, if they are rollback again and again. But both schemes wait-die and wound-wait avoid starvation by making use of timestamps.

Deadlock Detection and Recovery

1. Introduction

(i) When system is having deadlock detection and recovery techniques, the detection is done periodically to check whether the system is in deadlock, if yes then the recovery techniques are used to resolve this deadlock.

(ii) This can be done with help of some deadlock detection algorithms.

(iii) To achieve this, system must do the following :

(a) System should maintain information about current data items allocation to transactions and requests to be satisfied.

(b) Algorithm that uses this information to detect deadlock.

(c) Recovery techniques to be applied afte detection of deadlock.

2. Deadlock detection

(i) Deadlock can be detected using directed grap called as wait-for-graph.

(ii) The graph G = (V,E) can be seen as V is of vertices i.e. set of transaction in execution concurrently and E is set of edges.

(iii) Such that edge Ti→ Tj if Ti→ Tj is present in graph it shows that, transaction Tj is waiting for transaction Ti to release a data item it needs.

(iv) If cycle is present in wait-for-graph then deadlock is present and transactions in cycle are deadlock.

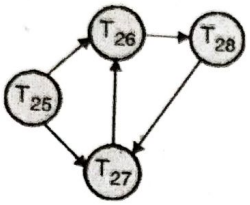


Fig. 13.1 : Wait-for graph with a cycle

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

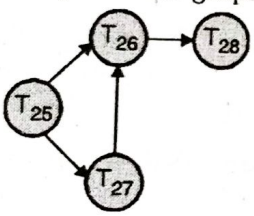


Fig. 13.2 : Wait-for graph with no cycle

(vi) If no cycle is present it mean no deadlock in system.

(vii) If deadlock occurs frequently, then detection algorithm should be invoked more frequently problem in this scenario is the data items locked by deadlock transaction will be unavailable until the deadlock is problem.

(viii) This may tend to more cycles in graph degrading capacity of granting lock requests.

3. Recovery from deadlock

- When deadlock is detected in the system, then system should be recovered from deadlock using recovery schemes.
- A most common solution is rollback one or more transaction to break the deadlock.
- Methods for recover from deadlock :

(a) Selection of victim

- If deadlock is detected then a transaction one or more transactions (victims) to be selected to break the deadlock.
- Transactions with minimum cost should be selected for rollbacks.
- Cost can be detected by following factors.
  - For how much time transaction has computed and how much time it needs to do.
  - How many data items it has locked ?
  - How many data items it may need ahead ?
  - Number of transaction to be rollback.

(b) Rollback

- Once victims are decided then there are two ways to do so.
- Total rollback : The transaction is aborted and then restart it.
- Partial rollback : Rollback the only transaction which is needed to break the deadlock.
- But this approach needs to record some more information like state of running transactions, locks on data item held by them, and deadlock detection algorithm specifies points up to which transaction to be rollback and recovery method has to rollback.
- After sometime transaction resume; partial transaction.

(c) Starvation

- It may happen every time same transaction is selected as victim and this may lead to starvation of that transaction (minimum cost transaction it selected every time).
- System should take care that every time same transaction should not be selected as victim. So it will not be starved.

Q.3 Explain time stamp based concurrency control.

SPPU - Dec. 17, Oct.19, 4 Marks

Ans. : Timestamp Based Protocols

(1) Introduction

- To achieve serializability order of transactions for execution can be decided in advance using its time at which transaction entered in system.

- A general method to achieve this is using time stamp ordering protocol.

(2) Concept of Timestamp

- A fixed timestamp is assigned at start of execution of the transaction. Every transaction Tj has been assigned a timestamp by database system denoted as TS (Tj).
- A transaction which has entered in system recently will have greater timestamp. If transaction Tj starts after Ti then,

  TS (Ti) < TS (Tj)
- System clock is used as timestamp i.e. system time when transaction Tienters system or a logical counter can be used as timestamp and incremented after every assignment.
- If TS(Ti) < TS(Tj), then system must ensure that serializable schedule is equivalent to a serial schedule as < Ti, Tj>
- Every data item X is with two timestamp values :

(a) W-timestamp (X)

- It denotes the largest timestamp of any transaction that executed WRITE (X) successfully on given data item X.
- That mean it is timestamp of recent WRITE (X) operation.
- On execution of next WRITE (X) operation (O1), W-timestamps will be updated to new timestamp of O1 i.e. TS (O1).

(b) R-timestamp (X)

- Denotes the largest timestamp of any transaction that executed READ (X) successfully on data item (X).
- That mean it is timestamp of recent READ (X) operation.
- On execution of next READ (X) operation (O1), R-timestamps will be updated to new timestamp of O1 i.e. TS (O1).

(3) Timestamp - ordering Protocol

- This protocol ensures any conflicting READ or WRITE is executed in order of timestamp.
- If by any reasons, if transaction is aborted then on restarting, new timestamp is assigned.

Working

(a) For transaction Ti to execute READ (X) Operation,

If TS (Ti) < W - timestamp (X)

Then Ti is trying to read value of X that is overwritten by other transaction.

W - timestamp (X) = 149 (Recent Write)			
TS	**Ti**	**Tx**	**Operations**
148	READ (X)	...	TS (Ti) < W - timestamp (X) i.e. 148 < 149 (W-Timestamp)
149	Unlock (X)	WRITE (X)	Record W - timestamp (X) = 149

So this READ is rejected (as Tx is already performing write operation on data so no other operation can be performed by any other transaction) and Ti is rolled back.

If TS (Ti) ≥ W - timestamp (X)

Then READ executed and set

R-timestamp(X) = max {TS (Ti), R-timestamp}			
TS	Ti	Tx	Operations
148	....	WRITE (X)	Record W - timestamp (X) = 148 (recent write)
149	READ (X)	....	TS (Ti) ≥ W - timestamp (X) i.e. 149 ≥ 148 (W-Timestamp) Record R-timestamp(X) = 149
150	....	....	....
151	READ (X)	....	TS (T1) ≥ W - timestamp (X) i.e. 151 ≥ 148 (W-Timestamp) Record R-timestamp(X) = max{149,151} = 151

(b) If transaction Tiexecute WRITE (X) Operation,

If TS (Ti) < R - timestamp (X)

Then Ti has produced value of X which is not needed now so rollback Ti.

TS	Ti	Tx	Operations
148	....	...	...
149	WRITE(X)	....	TS (T1) < R - timestamp (X) i.e. 149 < 151 (R-Timestamp) Reject WRITE roll back Ti
150	....	....	....
151		READ (X)	Record R-timestamp(X) = 151 (Recent READ Operation)

If TS (Ti) < W - timestamp (X)

- Then Ti is trying to write obsolete value of X, So rollback Ti

TS	Ti	Tx	Operations
148	....	...	...
149	WRITE (X)	....	TS (T1) < W - timestamp (X) i.e. 149 < 151 (W-Timestamp)
			Cannot write as other transaction using data X for writing.
150	....	....	....
151		WRITE (X)	Record W-timestamp(X) = 151

(c) Otherwise, system executes WRITE (X) and sets W-timestamp (X) = TS (Ti)

TS	Ti	Tx	Operations
150	....	....	....
151	WRITE (X)		Record W-timestamp(X) = 151

(4) Advantages

- This protocol ensures conflict serializability, as conflicting operations are processed in order of timestamp of operation.
- Ensures that it is free from deadlock.

(5) Disadvantages

- Starvation is possible for long transaction if short transaction conflicts with it causes restorative of long transaction again and again.
- It may not give recoverable schedules.

❑❑❑

Chapter - 14 : Recovery Methods

Q.1 Explain the concept of log and how it helps with database recovery with suitable diagram.

SPPU - May 19, Dec. 19, 4 Marks

Ans. : Log-Based Recovery

- There can be problem in accessing database due to any reason can causes a database system failure.
- The most widely used structure for recording database modifications is **transaction log (Log).**
- The log is a sequence of log records, recording all the update activities done on the database by all database users.

Types of Transaction Log

There are several types of log records.

(a) Update log record

- An update log record describes a single database write.
- It also includes the value of the bytes of page before and after the page change.

(b) Compensation log record

- Compensation log record the rollback of a particular change to the database.
- Each corresponds with exactly one other Update Log Record.

(c) Commit record

Records a decision to commit a transaction.

(d) Abort record

Records a decision to abort and hence rollback a transaction.

(e) Checkpoint record

- Records a point when checkpoint has been made.
- These are used to speed up recovery.
- It also record information that eliminates the need to read a log's past.
- The time of recording varies according to checkpoint algorithm.

(f) Completion record

Records all work has been done for this particular transaction.

Fields in transaction log

It has these fields :

(a) Transaction identifier : It is the unique identifier of the transaction that performed the write operation.

(b) Data-item identifier : It is the unique identifier of the data item written. Typically, it is the location on disk of the data item.

(c) Old value : It is the value of the data item prior to the write.

(d) New value : It is the value that the data item will have after the write.

Working of Log Based Protocol

- There are some special log records exist to record events during transaction processing, such as the start of a transaction and the commit or abort of a transaction.
- We denote the various types of log records as :

  $< T_n$ start$>$: Transaction T_n has started.

  $< T_n, X_j, V_1, V_2>$: Transaction T_n; has performed a write on data item X_j. X_j had value V_1 before the write, and will have value V_2 after the write.

  $< T_n$; commit$>$: Transaction T_n has committed.

  $< T_n$ abort$>$: Transaction T_n has aborted.
- Whenever a transaction performs a write operation, it is essential that the log record for that transaction should be created before the database is modified by it.
- Once a log record exists, we can write the modification to the database only if it is desirable.
- Undo operation can be done for modification that has already been done to the database. We undo it by using the old-value field in log records.
- For log records to be useful for recovery from system and disk failures, the log must reside in stable storage.

- We assume that every log record is written to the end of the operation on stable storage as soon as log is created.

Q.2 What is the basis of immediate updates recovery technique ? What does the deterred updates recovery technique involve? **SPPU - May 19, 6 Marks**

Ans. : Immediate-Modification Technique (UNDO Algorithm)

1. Introduction

- The **immediate-modification technique** allows database modifications to be written to the database when the transaction is still in the active state. (Running state)
- Data modifications written by active transactions are called **uncommitted modifications.**
- In the event of a crash or a transaction failure, the system must use the old-value field of the log records to restore the modified data items to the value had prior to the start of the transaction.

2. UNDO operation

- Before a transaction T_n starts its execution, the system writes the record <T_n start> to the log.
- During transaction execution, any WRITE**(X) operation by T_n** is preceded by the writing of the appropriate new update record to the log.
- When **T_n partially commits**, the system writes the record <T_n commit> to the log.
- Since the information in the log is used in restoring the original database state, we cannot allow the actual update to the database to take place before the corresponding log record is written out to stable storage.
- We therefore require that, before the execution of an output (B) operation its log records is written to stable storage.
- This procedure is defined as follows :

(1) Use two lists of transactions maintained by the system;

(a) Committed transactions since the last checkpoint

(b) Active transactions

(2) Undo all the WRITE operations of the active transaction from the log, using the UNDO procedure.

(3) Redo the WRITE operations of the committed transaction from the log in the order in which they were written in the log, using the REDO procedure.

UNDO operation

- Undoing a WRITE operation, consists of examination its log entry of write T reading Old value, new value and setting the value of item X in the database to old value which is the before image.
- Undoing a number of WRITE operations from one or more transactions from the log must proceed in the reverse order from the order in which the operations were written in the log.

Example :

- Consider a simple banking system, with transactions T_0 and T_1 executed one after the other in the order T_0 followed by T_1.
- Figure below shows possible order of execution in both the database and the log as a result of the execution of T_0 and T_1

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>
< T_0 commit>
<T_1 start>
<T_1, C, 100, 200>
<T_1 commit>

Log	Database
<T_0 start> <T_0, A, 1200, 950> <T_0, B, 2000,2050>	
	A = 950 B = 2050
<T_0 commit> <T_1 start> <T_1, C, 700, 800> < T_1 commit >	C = 800

- Using the log file, the system can handle any failure that does not result in the loss of information in non volatile storage.
- The recovery scheme uses two recovery procedures :

Method 1 :

(a) Undo(T_n) restores the value of all data items updated by transaction T_n to the old values and redo(T_n) sets the value of all data items updated by transaction T_n to the new values.

(b) The set of data items updated by T_n and their respective old and new values can be found in the log.

(c) The undo and redo operations must guarantee to correct behaviour, even if a failure occurs during the recovery process.

(d) After a failure has occurred, the recovery scheme consults the log to determine which transactions need to be redone, which need to be undone:

- Transaction T_n needs to be undone if the log contains the record < T_n start>, but does not contain the record <T_n commit>.
- Transaction T_n needs to be redone if the log contains both the record < T_n start> and the record <T_n commit>.
- In our banking example, with transaction T_0 and T_1 executed one after the other in the order T_0 followed by T_1.

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>

Fig. 14.1

Method 2 :

(a) If system crashes before the completion of the transactions

- First, let us assume that the crash occurs just after the log record for the step of transaction T_0 has been written to stable storage.
- When the system comes back up, it finds the record <T_0 start> in the log, but no corresponding <T_0 commit> record.
- Thus, transaction T_0 must be undone, so an undo (T_0) is performed.
- As a result, the values in accounts A and B (on the disk) are restored to Rs.100 and Rs.100, respectively.

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>
<T_0 commit>
<T_1 start>
<T_1, C, 300, 200>

Fig. 14.2

(b) If crash comes occurs after the log record for the step write (C) of transaction T_1 has been written to stable storage.

- When the system comes back up, two recovery actions need to be taken.
- The operation UNDO(T_1) must be performed, since the record <T_1 start> appears in the log, but there is no record <T_1 commit>.
- The operation REDO(T_0) must be performed, since the log contains both the record <T_0 start> and the record <T_0 commit>.
- At the end of the entire recovery procedure, the values of accounts A, B, and C are Rs.50, Rs.150, and Rs.300, respectively. Note that the UNDO(T_1) operation is performed before the redo(T_0).
- In this example, the same outcome would result if the order were reversed.
- However, the order of doing undo operations first, and then redo operations, is important for the recovery algorithm.

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>
<T_0 commit>
<T_1 start>
<T_1, C, 200, 300>
< T_1 commit>

Fig. 14.3

(c) If crash occurs just after the log record <T_1 commit> has been written to stable storage.

- When the system comes to backup, both T_0 and T_1 need to be redone, since the records <T_0 start> and <T_0 commit> appear in the log, as do the records <T_1 start> and <T_1 commit>.
- After the system performs the recovery procedures REDO(T_0) and REDO(T_1), the values in accounts A, B, and C, are Rs. 50, Rs. 150, and Rs. 300, respectively.

Q.3 What is a checkpoint? Explain the operations performed by a system during checkpoint. Explain the recovery mechanism during system crash.

SPPU - May 18, Dec.18

Ans. : Recovery Related Structures – Checkpoint

Introduction

- A database checkpoint is where all committed transactions are written to the redo/audit logs.
- The database administrator determines the frequency of the checkpoints based on volume of transactions.
- When a system failure occurs, we must consult the log to determine those transactions that need to be redone and those that need to be undone using above log files.
- Too frequent checkpoints can affect the performance.

Problems in This Approach

- The search process is time-consuming.
- Most of the transactions that, according to our algorithm, need to be redone as they have already written their updates into the database.

Need of Checkpoints

- To reduce these types of overhead, we introduce checkpoints.

During execution, the system maintains the log, using one of the two techniques.

The system periodically performs checkpoints, with following sequence of actions :

1. Output all log records onto stable storage which are currently stored in main memory.
2. Output to the disk all modified buffer blocks.
3. Output onto stable storage a log record <checkpoint>.

- Transactions are not allowed to perform any update actions, such as writing to a buffer block or writing a log record, while a checkpoint is in working state.
- The presence of a <checkpoint> record in the log allows the system to restructure its recovery procedure.

Working

- Consider a transaction T_n that committed prior to the checkpoint.
- For such a transaction, the <T_n commit> record appears in the log before the <checkpoint> record.
- Any database modifications made by transaction T_n must have been written to the database either prior to the checkpoint or as part of the checkpoint itself. Thus, at recovery time, there is no need to perform a redo operation on T_n.
- After a database failure has occurred, the recovery scheme examines the log to determine the most recent transaction T_n that started executing before the most recent checkpoint took place.
- It can find such a transaction by searching the log backward, from the end of the log, until it finds the first <checkpoint> record (since we are searching backward, the record found is the final <checkpoint> record in the log); then it continues the search backward until it finds the next <T_n start> record. This record identifies a transaction T_n.
- Once the system has identified respective transaction T_n, the redo and undo operations need to be applied to only for transaction T_n and all transactions that started executing after transaction T_n.
- The exact recovery operations to be performed depend on the modification technique being used.
- For the immediate-modification technique, the recovery operations are :
  - For all transactions T_k in T that have no <Tk commit> record in the log, execute UNDO(Tk).

- For all transactions Tk in T such that the record <Tk commit> appears in the log, execute REDO(T_k).

- Obviously, the undo operation does not need to be applied when the deferred-modification technique is being employed.
- Consider the set of transactions {T_0, T_1 .... , T_{10}} executed in the order of the subscripts. Suppose that the recent checkpoint took place during the execution of transaction T_6. Thus, only transactions T_6, T_7, ... , T_{10} need to be considered during the recovery scheme. Each of them needs to be redone if it has committed; otherwise, it needs to be undone.

Advantages

(1) A Database storage checkpoint keeps track of block change information and thereby enables incremental database backup at the block level.

(2) A Database Storage Checkpoint helps recover data from incorrectly modified files.

(3) A Database Storage Checkpoint can be mounted, allowing regular file system operations to be performed. Mountable Database Storage Checkpoints can be used for a wide range of application solutions that include backup, investigations into data integrity, staging upgrades or database modifications, and data replication solutions.

Disadvantages

(1) Database Storage Checkpoints can only be used to restore from logical errors (E.g. a human error).

(2) Because all the data blocks are on the same physical device, Database Storage Checkpoints cannot be used to restore files due to a media failure.

Q.4 Explain shadow paging with proper example.

SPPU - Dec. 17, 4 Marks

Ans. : Shadow Paging

Introduction

- It is not always convenient to maintain logs of all transactions for the purposes of recovery. An alternative is to use a system of shadow paging.
- This is where the database is divided into pages that may be stored in any order on the disk.
- In order to identify the location of any given page, we use something called a **page table**.

Overview

- This method considers the database is made up of a number of fixed size pages.
- A directory with n entries is constructed which the entry points to the database page on disk.
- This directory is kept in the main memory.
- It is not too large and all references read and write to database pages on disk.

Method

(a) During the life of a transaction two page tables are maintained as below,

(i) Shadow page table

(ii) Current page table.

(b) When a transaction begins both of these page tables point to the same locations (are identical).

(c) During the lifetime of a transaction the shadow page table doesn't change at all.

(d) However during the lifetime of a transaction update values etc. may be changed.

(e) For pages updated by the transaction, two versions are kept. The old version is referenced by the shadow directory and the new version by the current directory.

(f) So whenever we update a page in the database we always write the updated page to a new location.

(g) This means that when we update our current page table it reflect the changes that have been made by that transaction.

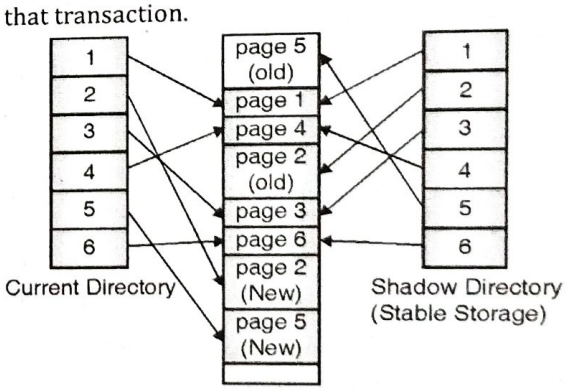


Fig. 14.4 : Page table -shadow paging

(h) Looking at below diagram we see how these tables appear during a transaction. As we can see the

shadow page table shows the state of the database just prior to a transaction, and the current page table shows the state of the database during or after a transaction has been completed.

Process of Recovery

- We now have a system whereby if we ever want to undo the actions of a transaction all we have to do is recover the shadow page table to be the current page table.
- As such this method makes the shadow page table particularly important, and so it must always be stored on stable storage.
- On disk we store a single pointer location that points to the address of the shadow page table.
- This means that to swap the shadow table for the current page table (commiting the data) we just need to update this single pointer (very unlikely to fail during this very short fast operation).
- In case of no failure means while committing transaction just discard the shadow directory.
- In case of multi-user environment with concurrent transaction, logs and checkpoints must be incorporated in shadow paging.

Advantages

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

Disadvantages

(1) Data fragmentation

The main disadvantage of this technique is the updated Data will suffer from fragmentation as the data is divided up into pages that may or not be in linear order for large sets of related hence, complex storage management strategies.

(2) Commit overhead

If the directory size is large, the overhead of writing shadow directories to disk as transaction commit is significant.

(3) Garbage collection

- Garbage will accumulate in the pages on the disk as data is updated and pages loose any references. For example if i have a page that contains a data item X that is replaced with a new value then a new page will be created. Once the shadow page table is updated nothing will reference the old value of X.
- The operation to migrate between current and shadow directories must be implemented as an atomic mode.

Q.5 Explain Query optimization with respect to SQL databases. **SPPU - Dec. 18, May 19, 6 Marks**

Ans. : Query Optimization in SQL

(1) **Query optimization** is an essential for many DBMS in which best query execution plan, out of multiple alternative query execution plans is to be identified for solving a query efficiently.

(2) Query optimization is nothing but selecting the efficient query execution plan among the available query plans for solving a given query.

(3) The query plan is constructed based on multiple cost factors.

(4) Query optimization tries to select query plan which is lower processing cost (but not always low cost plans are selected as other factors are also included in query evaluation).

(5) Query optimization come into picture when we want to develop a good system to minimize the cost of query evaluation.

(6) There is a trade off between the **amount of time spent to find out the best plan** and the **amount of time required for running the plan**. Different DBMS have different ways for balancing these two factors.

Goals of Query Optimization

(a) Eliminate all unwanted data

Query optimization tries to eliminate unwanted tuples, or rows from given relation.

Example :

```
SELECT  *
FROM    EMPLOYEES
```

If we require only employee Ids then select only employee_Id column from table. No need to select all the columns from table by specifying * in select clause.

So, Optimized query will be

```
SELECT  Emp_ID
FROM    EMPLOYEES
```

(b) Speed up queries

Query optimization try to find out query which gives result very fast.

(c) Increase query performance by performance tuning

- Break up the single complex SQL statement into several simple SQL statements which will increase performance of execution of query.
- In many cases, you can build an index on the temporary table (such tables used to store intermediate results of query) in order to speed up the query performance.

(d) Select best query plan out of alternative query plan

Consider following query to "Find all employees having age above 25 and working for department HR."

```
SELECT  e.Emp_Name
FROM    EMPLOYEES e
        JOIN
        DEPARTMENTS d
        ON      e.did=d.did
WHERE     e.age > 25
AND             d.dname= 'HR';
```

For above query we can write alternative relational queries like below,

Option 1

$$\pi_{e.Emp\_Name}(\sigma_{e.Age>25\ AND\ d.dname='HR'}(EMPLOYEES\ e \bowtie_{e.did=d.did} Departments\ d))$$

Option 2

$$\pi_{e.Emp\_Name}((\sigma_{e.Age>25}(EMPLOYEES\ e)) \bowtie_{e.did=d.did} (\sigma_{d.dname='HR'}(Departments\ d)))$$

Now, which of above query option is to be selected is identified by query optimizer.

Query Optimization Approaches / Performance Tuning

Relational approach

In RDBMS system attempts to find query expression which is equivalent to the given query expression, but execute in more efficient way.

Algorithm approach

Selecting query by choosing algorithm for operation or choosing specific indices to use and so on.

Other commonly used strategies for optimization

- **Use index**
  - Using an index is strategy that can used to speed up a query evaluation.
  - This strategy is very important for query optimization.
- **Aggregate table**
  - Tables which stores only higher level data so fewer amounts of data need to be parsed.
  - So, Query evaluation becomes faster.
- **Vertical partitioning**
  - Slicing the table vertically using columns.
  - Vertical partitioning will decrease the amount of data required for query processing.
- **Horizontal partitioning**
  - Partition the table by data value, or by row wise.
  - This method will decrease the amount of data query needs to process.
- **Denormalization**
  - The process of combining multiple tables into a single table is called as denormalization.
  - This speeds up query performance because less number of joins is required.
- **Server tuning**

  Each server has its own parameters, and often tuning server parameters so that it can fully take advantage of the hardware resources and significantly speed up the query performance.

Query Optimizers

- Query optimization is one of the most important tasks of a relational DBMS.
- The Query optimizer generates alternative plans and chooses the best plan with the least estimated cost.
- Query optimizer is responsible for identifying a best execution plan for evaluating the query.
- The optimizer generates multiple query plans and selects the plan with the least estimated cost.
- To estimate the cost of every plan, the optimizer uses system catalogs.
- The system catalogs contain the information needed by the optimizer to choose between alternate plans for a given query.

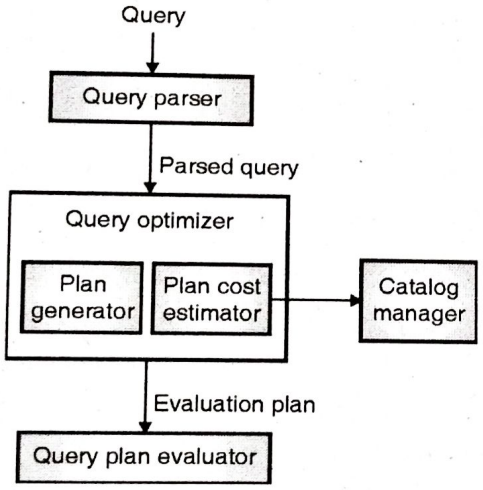


Fig. 14.5 : Query parsing, optimization and execution

- Query evaluation plan can have a remarkable impact on query execution time.

Query Optimization in NoSQL

(1) In case of NoSQL data, efficiency of read operations can be improved by reducing the amount of data that query operations need to process.

(2) If NoSQL queries a set of fields, then an index on such queried field can prevent the query from scanning entire data set.

(3) Governing methods to optimize query performance is done by Performance Tuning.

(4) Query selectivity can be used for performance tuning which refers to filtering out unrelated documents in a collection.

(5) Many of above SQL Query optimization approaches are applicable for NoSQL database also.

❑❑❑

Unit VI

Chapter - 15 : Database Architecture

Q.1 What are the key elements of Parallel DB processing? Explain.

SPPU - May 18, Dec. 19, 4 Marks

Ans. : Parallel Databases System

Introduction

- A parallel database improves data processing speed by using multiple resources (CPU and Disk) in parallel.
- Parallel operations are becoming increasingly common to improve speed of operation and therefore study of Parallel databases is becoming more important.
- In organizations huge amount of data is handled, such huge amount of data needs high data transfer rate.
- Centralized and client server system is not much powerful for managing such applications.
- Parallel databases try to improve performance of database through parallelization of operations such Data loading, Query evaluation etc.
- We can use thousands of small processors for making a parallel machine.

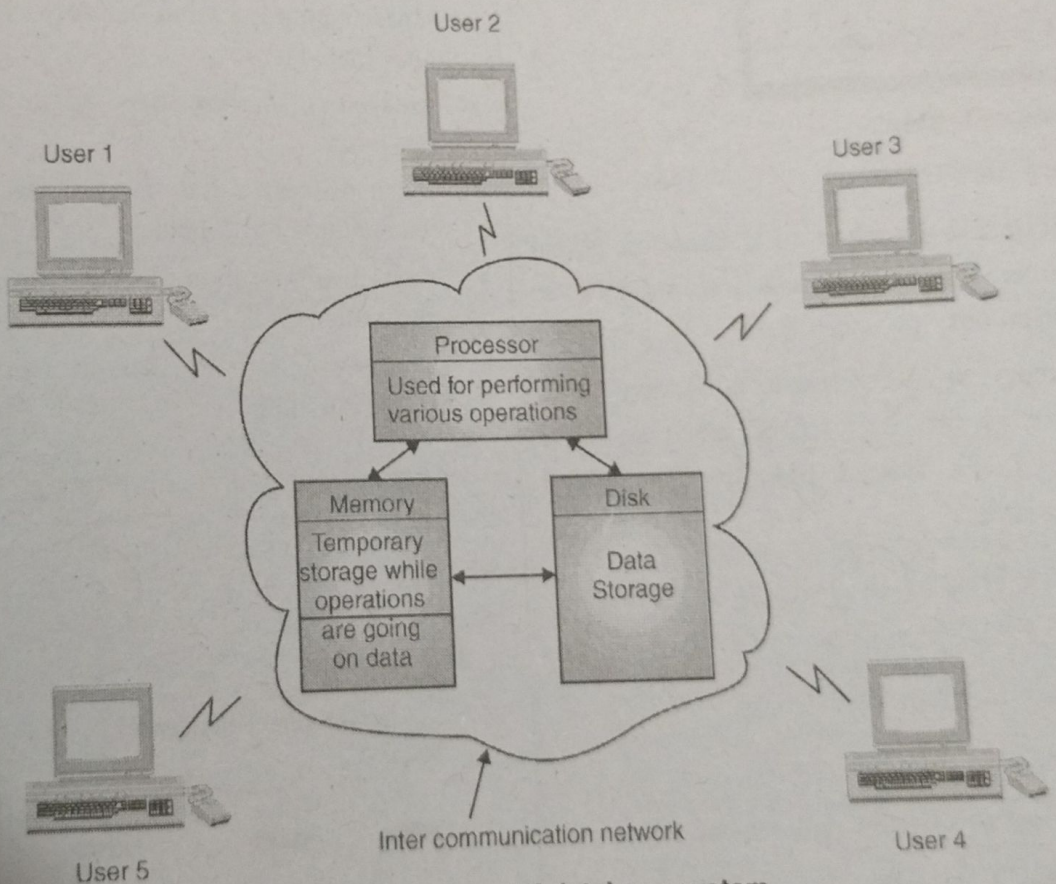


Fig. 15.1 : Parallel database system

Goals of Parallel Databases

(a) Improved performance

Using multiple resources (e.g., CPUs and disks) in parallel we can significantly improve performance of sy

(b) Increased availability

If a site containing a relation (table in database) is not available, then the relation continues to be availa another site which has a copy of that data.

(c) Increased reliability

If a site containing a relation (table in database) fails to work, the relation continues to be available from another site which has a copy of that data. Hence this leads to more reliable system.

(d) Distributed data access

An organization may need to access data which belongs to different sites, as it may be possible to have multiple branches of a company.

Q.2 Explain Architecture of Parallel Databases.

SPPU - Dec. 17, May 18, 6 Marks

Ans. : Architecture of Parallel Databases

Parallelism in databases represents one of the most successful instances of parallel computing system.

Types of Parallel Databases

1. Shared Memory System
2. Shared Disk System
3. Shared Nothing System

(A) Shared Memory System

(a) Architecture details

- Multiple CPUs are attached to a common global shared memory via interconnection network or communication bus.
- Shared memory architectures usually have large memory caches at each processor, so that referencing of the shared memory is avoided whenever possible.

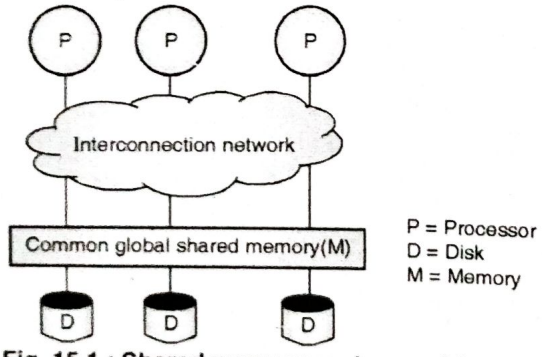


Fig. 15.1 : Shared memory system architecture

- Moreover, caches need to be coherent. That means if a processor performs a write to a memory location, the data in that memory location should be either updated at or removed cached data.

(b) Advantages

- Efficient communication between processors.
- Data can be accessed by any processor without being moved from one place to other.
- A processor can send messages to other processors much faster using memory writes.

(c) Disadvantages

- Bandwidth problem
- Not scalable beyond 32 or 64 processors, since the bus or interconnection network will get into a bottleneck.
- More number of processors can increase waiting time of processors.

(B) Shared Disk System

(a) Architecture details

- Multiple processors can access all disk directly via inter communication network. But, every processor has local memory.
- Shared disk has two advantages over shared memory.
- Each processor has its own memory; the memory bus is not a bottleneck.
- System offers a simple way to provide a degree of fault tolerance.
- The systems built around this architecture are called clusters.

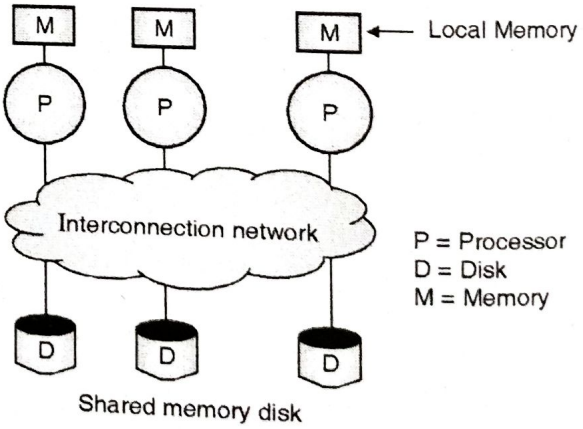


Fig. 15.2 : Shared memory disk architecture

(b) Advantages

- Each CPU or processor has its own local memory, so the memory bus will not face bottleneck.
- High degree of fault tolerance is achieved.

- Fault tolerance : If a processor (or its memory) fails, the other processor can take over its tasks, since the database is present on disks and are accessible to all processors.
- If one processor fails, other processors can take over its tasks, since database is on shared disk that can be accessible from all processors.

(c) Disadvantages

- Some memory load is added to each processor.
- **Limited scalability :** Not scalable beyond certain point. The shared-disk architecture faces this problem because large amounts of data are shipped through the interconnection network. So now the interconnection to the disk subsystem is a bottleneck.
- The basic problem with the shared-memory and shared-disk architecture is interference. As more CPUs are added, existing CPUs are slowed down because of the increased contention for memory accesses and network bandwidth.

(d) Applications

Digital equipment corporation : DEC cluster running Relational Databases were one of the early commercial user of shared disk database architecture. Now this is owned by Oracle.

(C) Shared Nothing Disk System

(a) Architecture details

- Each processor has its own local memory and local disk.
- A processor at one node may communicate with another processor using high speed communication network.

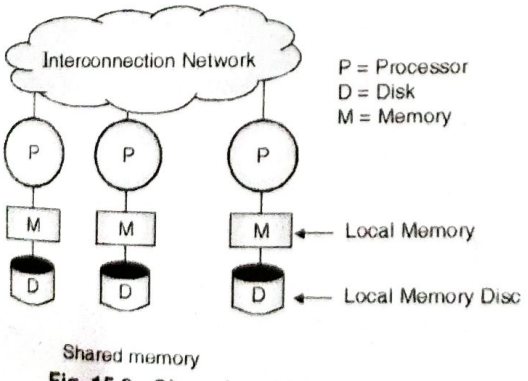


Fig. 15.3 : Shared nothing architecture

- Any terminal can act as a Node which functions as Server for data that is stored on local disk.
- Moreover, the interconnection networks for shared nothing systems are usually designed to be scalable, so that we can increase transmission capacity as more nodes are added to the network.

(b) Advantages

- In this type of architecture no need to go through all I/O. Only a single interconnection network queries which access non local disk can pass through the network.
- We can achieve High degree of parallelism. i.e. Number of CPU and disk can be connected as desired.
- Shared nothing architecture systems are more scalable and can easily support a large number of processors.

(c) Disadvantages

- Cost of communication and of non local disk access is higher than other two architectures since sending data involves software interaction at both ends.
- Requires rigid data partitioning.

(d) Applications :

- The teradata database machine uses shared nothing database architecture.
- Grace and the Gamma research prototypes.

(D) Hierarchical System

Architecture details

- The **hierarchical architecture** comes with combined characteristics of shared memory, shared disk and shred nothing architectures.
- At the top level, the system consists of nodes connected by an interconnection network and they do not share disks or memory with one another.
- This architecture is attempts to reduce the complexity of programming such systems yields to **distributed virtual-memory** architectures, where logically there is a single shared memory, the memory mapping hardware coupled with system software, allows each processor to view the disjoint memories as a single virtual memory.
- The **hierarchical architecture** is also referred to as **non uniform memory architecture**.

Q.3 Explain Data Replication and Data Fragmentation in Distributed Data Storage.

SPPU - May 18, Dec.18, May 19, Dec. 19, 6 Marks

Ans. : Data Fragmentation / Distributed Database Design

Introduction

- The process of decomposing the database into smaller multiple units called as **fragments.**
- These fragments may be stored at various sites, is called **data fragmentation.**

Completeness constraint :

The most important condition of data fragmentation process is that it must be complete i.e. once a database is fragmented, it must be always possible to reconstruct the original database from the fragments.

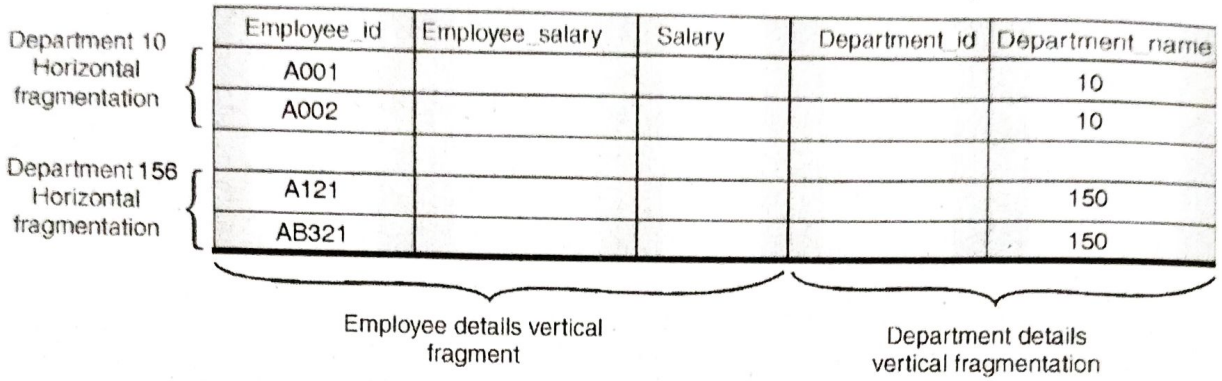


Fig. 15.4 : Horizontal and vertical fragmentation

Fragmentation Schema

It is a set of fragments that includes all attributes and tuples in the database and satisfies the condition that the whole database can be reconstructed from the fragments by applying some sequence of database operations.

Types of Data Fragmentation

(a) Horizontal data fragmentation
(b) Vertical data fragmentation
(c) Mixed data fragmentation

(A) Horizontal Fragmentation

(1) Introduction

- Horizontal fragmentation divides a relation horizontally into group of rows (tuple) to create subsets of tuples specified by a condition on one or more attributes of relation.
- The tuples that belong to the horizontal fragment is specified by some condition on one or more attributes of the relation.

(2) Overview

- Horizontal fragmentation is group of rows in relation.
- Horizontal fragments are specified by the 'SELECT' operation of the relational algebra on single or multiple attributes.

Example :

Select all students of computer branch.

$\sigma_{Branch = "COMP"}$ (students)

(3) Types

(a) Primary horizontal fragmentation
(b) Derived horizontal fragmentation
(c) Complete horizontal fragmentation

(a) Primary horizontal fragmentation

- Primary horizontal fragmentation is the fragmentation of primary relation.
- Relation on which other relations are dependent using foreign key is called as primary relation.

Example :

Partition 1 : All employees belong to department number 10.

$R_1 \leftarrow \sigma_{Dept = 10}$ (EMP)

Partition 2 : All employees belong to department number 20.

$R_2 \leftarrow \sigma_{Dept = 20}$ (EMP)

(b) Derived horizontal fragmentation

- Horizontal fragmentation of a primary relation introduces Derived horizontal fragmentation of other secondary relations that are dependent on primary relations.
- The fragmentation on some other fragmentation is called as Derived horizontal fragmentation.

Example :

Partition 1 : All employees belong to department number 10 and having age above 25.

$\sigma_{Age > 25}$ (R_1)

Partition 2 : All employees belong to department number 20 and having age above 35.

$\sigma_{(Age > 35)}$ (R_2)

(c) Complete horizontal fragmentation

- It generates a set of horizontal fragments that include each and every tuple of original relation.
- For reconstruction of relation completeness is required. As every tuple must belongs to at least one of the partition.
- Consider relation below as R now subdivided in P_1, P_2, P_3 and P_4 In case of complete horizontal fragment if relation R contains 50 tuples than total number o tuples in below 4 partitions should be 50 or mor than 50.

Example :

Partition 1 : All employees belong to departmer number less than 20.

$R_1 \leftarrow \sigma_{DeptNo <= 20}$ (EMP)

Partition 2 : All employees belong to departme number less than 30.

$R_2 \leftarrow \sigma_{DeptNo = 30}$(EMP)

Partition 3 : All employees belong to departm number less than 40.

$R_3 \leftarrow \sigma_{DeptNo = 40}$ (EMP)

Partition 4 : All employees belong to departm number above 40.

$R_4 \leftarrow \sigma_{Dept > 40}$(EMP)

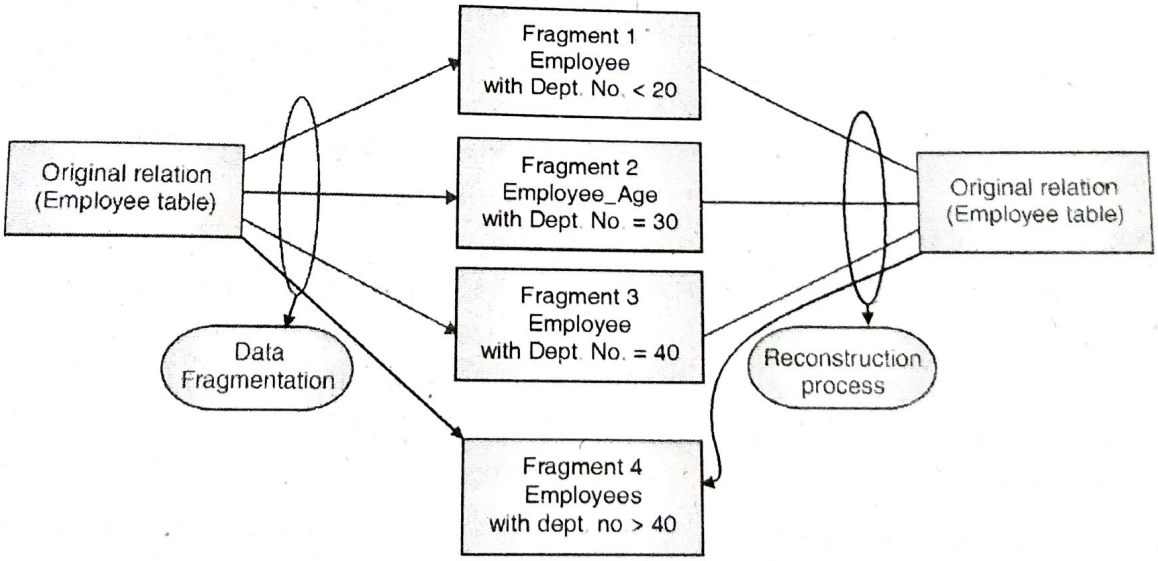


Fig. 15.5 : Complete horizontal fragmentation

(d) Disjoint horizontal fragmentation

- It generates a set of horizontal fragments where no two fragments have common tuples.
- That means every tuple of relation belongs to one and only one fragment.

Example :

Partition 1 : All employees having Age 18 or less.

$R_1 \leftarrow \sigma_{EmpAge <= 18}(EMP)$

Partition 2 : All employees having Age above 18 and below 65.

$R_2 \leftarrow \sigma_{EmpAge > 18 \text{ AND } EmpAge < 65}(EMP)$

Partition 3 : All employees having Age above 65.

$R_3 \leftarrow \sigma_{EmpAge >= 65}(EMP)$

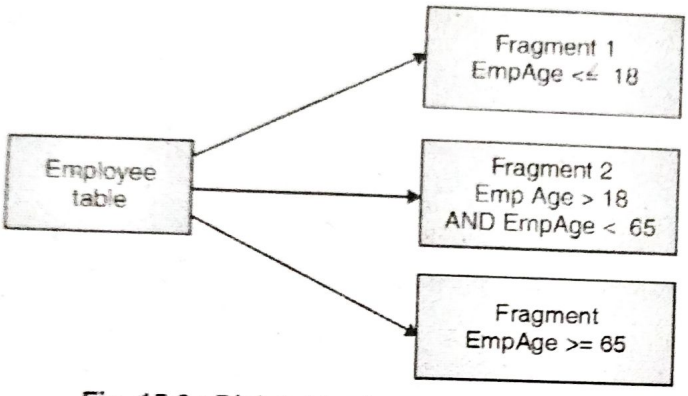


Fig. 15.6 : Disjoint horizontal partitioning

(4) Reconstruction process for horizontal fragments

- To reconstruct the original relation we need to perform set UNION (U) operation on fragments.
- The original relation can be reconstructed if and only if completeness constraint is satisfied.
- **Example :** Consider relation shown in complete horizontal partitioning we can reconstruct relation as it is satisfying completeness constraints

$R \leftarrow R_1 \cup R_2 \cup R_3 \cup R_4$

(B) Vertical Fragmentation

(1) Introduction

- Vertical fragmentation divides a relation vertically into group of columns.
- When each site does not need all the attributes of a relation, vertical fragmentation is used to fragment the relation vertically by columns.
- It is necessary to include primary key or some common candidate key in every vertical fragment to reconstruct the original relation from the fragments.

(2) Overview

- Vertical fragmentation is group of columns in relation.
- Vertical fragmentation can be specified by 'PROJECT' operation of the relational algebra.

Example :

Select Name and address of all students of computer branch.

$\pi_{Name, Address}(students)$

(3) Types of vertical fragmentation

- Complete vertical fragmentation
- It generates a set of vertical fragments that include all the attributes of original relation and share only primary key of original relation.
- Consider relation with following schema
- Student (sid, Name, Age, Address, Phone, class, fees)
  - $P_1 \rightarrow \pi_{Sid, Name, Address}(students)$
  - $P_2 \rightarrow \pi_{Sid, Age, Phone}(students)$
  - $P_3 \rightarrow \pi_{Sid, class, Fees}(students)$

(4) Reconstruction

- To reconstruct the original relation we need to perform FULL OUTER JOIN ($\bowtie$) operation on fragments.
- The original relation can be reconstructed if and only if completeness constraint is satisfied that means there should be either one column which is common between two partitions.

Example :

Consider above relation we can reconstruct relation as it is satisfying completeness constraints

$R \leftarrow P_1 \bowtie P_2 \bowtie P_3$

(C) Mixed (Hybrid) Fragmentation

(1) Introduction

- We can mix two types of fragmentation i.e. horizontal and vertical fragmentations yielding a mixed fragmentation.
- This fragmentation is generally used in many applications.

(2) Overview

- Mixed fragmentation is group of columns and rows in relation.
- Mixed fragmentation can be specified by 'PROJECT' and 'SELECT' operation of the relational algebra.

(3) Example

- Mixed fragmentation can be applied to student table for following student schema;

Student table

Sid	SName	age	Branch id	Bname

- Student table contains information about student and branches associated with particular student. Branches can be Computer Science (CS) or IT branch.

Table : Student database

	SName	Age	Sid	Branchid	Bname	
$F_1 \rightarrow$	Smriti	20	1	10	CS	$\leftarrow F_3$
	Jay	24	2	10	CS	
	Ashok	22	3	10	CS	
	Neha	21	4	20	IT	
	Raj	20	5	20	IT	$\leftarrow F_4$
$F_2 \rightarrow$	Harshad	23	6	20	IT	

(a) Fragment 1 : Student details of all students in 'CS' Branch.

$F_1 \rightarrow \pi_{Sid, SName, age}(\sigma_{Bname='CS'}(student))$

Sid	SName	Age
1	Smriti	20
2	Jay	24
3	Ashok	22

(b) Fragment 2 : Student details of all students in 'IT' Branch.

$F_2 \rightarrow \pi_{Sid, SName, age}(\sigma_{Bname='IT'}(student))$

Sid	SName	Age
4	Neha	21
5	Raj	20
6	Harshad	23

(c) Fragment 3 : Find all student's branch details of CS Branch

$F_3 \rightarrow \pi_{Sid, Branchid, BName}(\sigma_{Bname='CS'}(student))$

Sid	Branchid	Branch
1	10	CS
2	10	CS
3	10	CS

(d) Fragment 4 : Find all student's branch details of IT Branch.

$F_4 \rightarrow \pi_{Sid, Branchid, BName}(\sigma_{Bname='IT'}(student))$

Sid	Branchid	Branch
4	20	IT
5	20	IT
6	20	IT

(4) Reconstruction

- To reconstruct the original relation by performing Union and FULL OUTER JOIN ($\bowtie$) operation in appropriate order on fragments.
- The original relation can be reconstructed if and only if completeness constraint is satisfied that means there should be either one column which is common between two partitions.

Example :

To reconstruct above fragmentation we will first find union of $F_1$and F_2 which will give me details of all students.

$F_1 \cup F_2$

Sid	SName	Age
1	Smriti	20
2	Jay	24
3	Ashok	22
4	Neha	21
5	Raj	20
6	Harshad	23

Now, we will find details of or department in which student study by taking Union of F_3 and F_4.

$F_3 \cup F_4$

Sid	Branchid	Bname
1	10	CS
2	10	CS
3	10	CS
4	20	IT
5	20	IT
6	20	IT

Now to reconstruct main table we Join above two tables using Join ($\bowtie$) with help of 'Sid' as common column.

$$R \Rightarrow (F_1 \cup F_2) \bowtie (F_3 \cup F_4)$$

Above query will returns the original relation as in table: student database.

❑❑❑

Chapter - 16 : NoSQL Database

Q.1 Write a short note on Internet database.
SPPU - May 18, May 19, 4 Marks

Ans. : Internet Database

1. Introduction

- Large amount of data can be easily managed by database technology.
- The internet / web is a good way to present information to the user.
- In internet databases data management is separated from presentation.
- This improves the efficiency of the application. Updating data, finding information becomes very easy.

  Example : Employee database, airline / railway booking, e-commerce etc.
- Web-To-Database Middleware (ColdFusion)

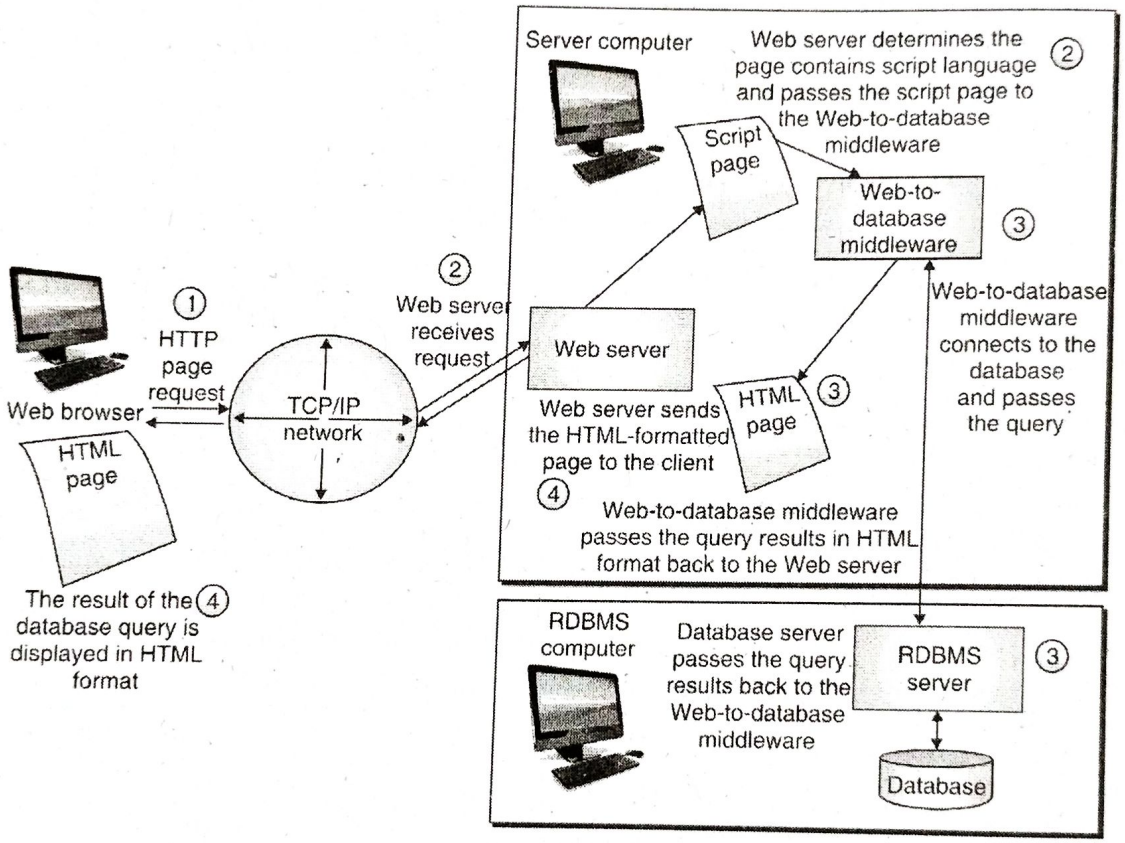


Fig. 16.1

- Cold fusion application middleware helps us in following ways :
  - Through web pages we can connect to a database easily.
  - Data can be represented on the web page in any format.
  - Dynamic web pages creation, updation becomes very easy.
  - Relationships, referential integrity etc. can be easily defined.
  - Database fields can be easily created or deleted.
  - Following Fig. 16.2 shows thathow cold fusion works in internet databases.

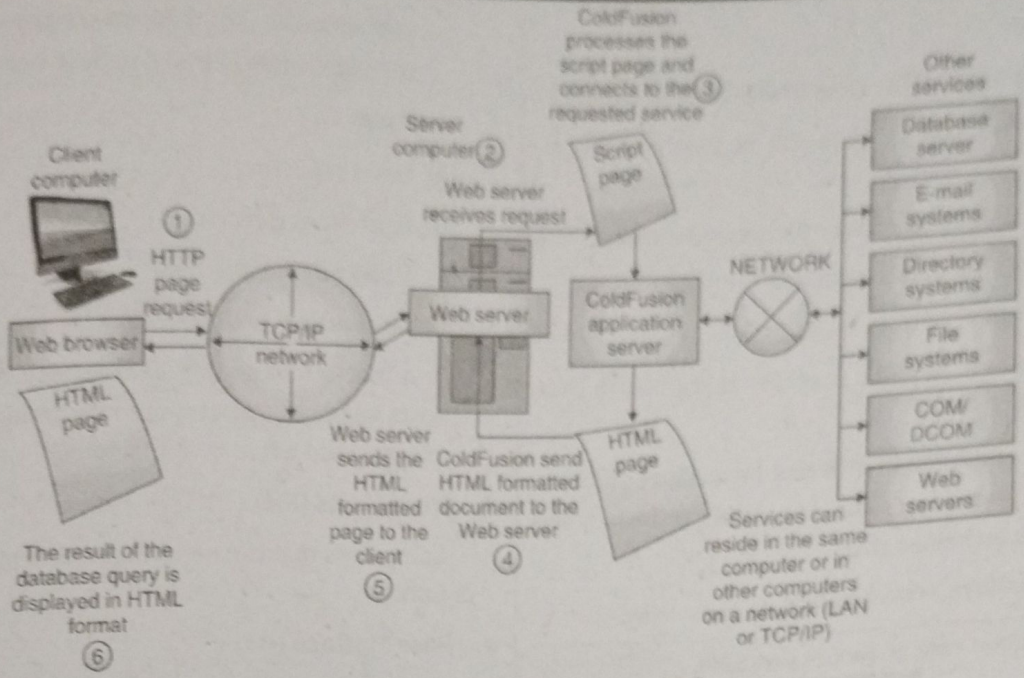


Fig. 16.2 : How cold fusion works ?

2. Web Server Interfaces

- **Common Gateway Interface (CGI)**

Scripts are nothing but small programs written in programming languages link C, VB, C++. Script files are used by CGI to perform a specific function. Client passes parameters to the webservers.

- **Application Programming Interfaces (APIs)**

APIs Web server interfaces are much more efficient and faster than CGI scripts. APIs are implemented as shared code or as dynamic-link libraries (DLLs). Thw following Fig. 16.2.3 shows API and CGI web Server Interfaces.

The API And CGI Web Server Interfaces.

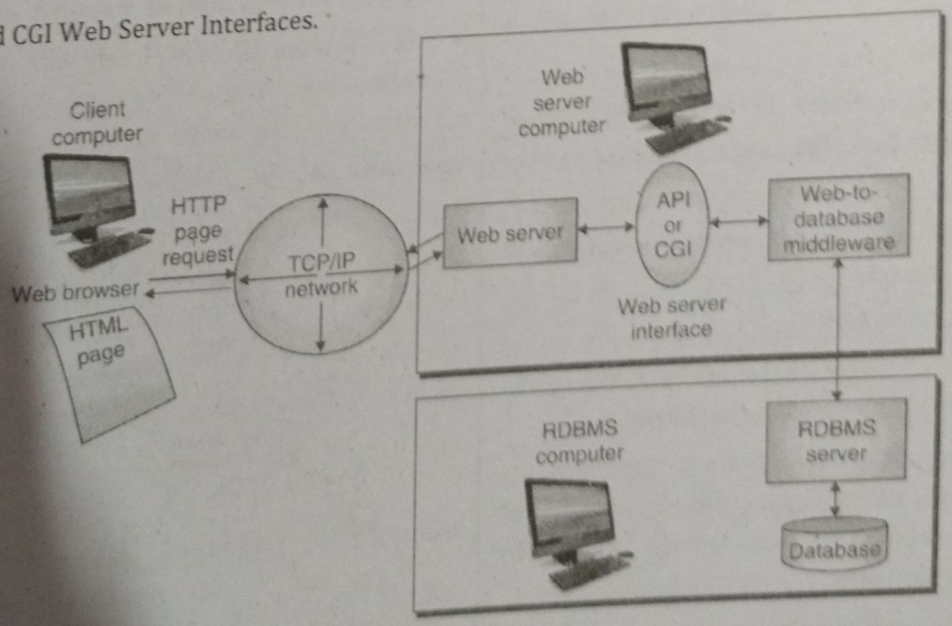


Fig. 16.3

3. Advantages

- Data is available all the time irrespective of location.
- Scalability is good i.e. data can be easily added or deleted.
- Saves hardware and software management cost.
- Huge amount of data is available for searching.

4. Disadvantages

- Secuiry issue : As data is online there is always a security threat.
- As data is available in different formats, at different locations, switching from one database technology to another increases cost.
- Network failure, server failure.

Q.2 Write a short note on Mobile database.

SPPU - May 18, Dec. 19, 6 Marks

Ans. : Mobile Databases

1. Introduction

Recent developments in wireless networking and introduction of portable devices have given rise to new era of mobile computing.

2. Goals

(a) Mobility

Mobile computing allows user or client to access required information from anywhere and at any point of time.

(b) Battery life

- Mobile devices works on battery which should have unlimited life (or unlimited power supply).
- Hence limited battery power should not delimit mobility of user.

(c) Changing networks

- Changing topology of the network should not cause any problems to mobile devices.
- In mobile environment, these problems are more difficult, mainly because of the limited and discontinuous connectivity offered by wireless networks.

(d) Some of the software issues which may involve data management problem, transaction management or database recovery are also present in distributed database systems

3. Examples :

(a) News reporting

(b) E- brokering services

(c) Mobile Billing services

Mobile Computing Architecture

1. In a mobile computing architecture made of fixed hosts and base stations, are interconnected through a high-speed wired network.

2. Fixed Hosts (FH)

Computers that can be configured to manage mobile units.

3. Base Stations (BS)

- Working as gateways for the mobile units.
- They are ready with wireless interfaces for mobile units or clients.

4. Cell (C)

- To manage the mobility of units, the entire coverage area is divided into one or smaller domains called as cell.
- Each cell has at least one base station.

5. Mobile Units (MU)

- Users or Clients for base station in mobile network
- Mobile units can move freely within cell or between cells.

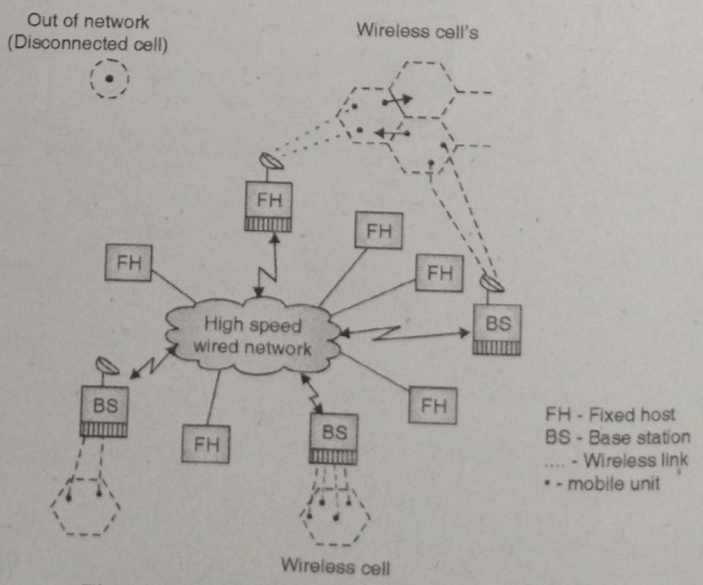


Fig. 16.1 : Mobile computing architecture

Characteristics of Mobile Environments

Mobile devices have some special characteristics that must be taken into account when designing applications for such environments.

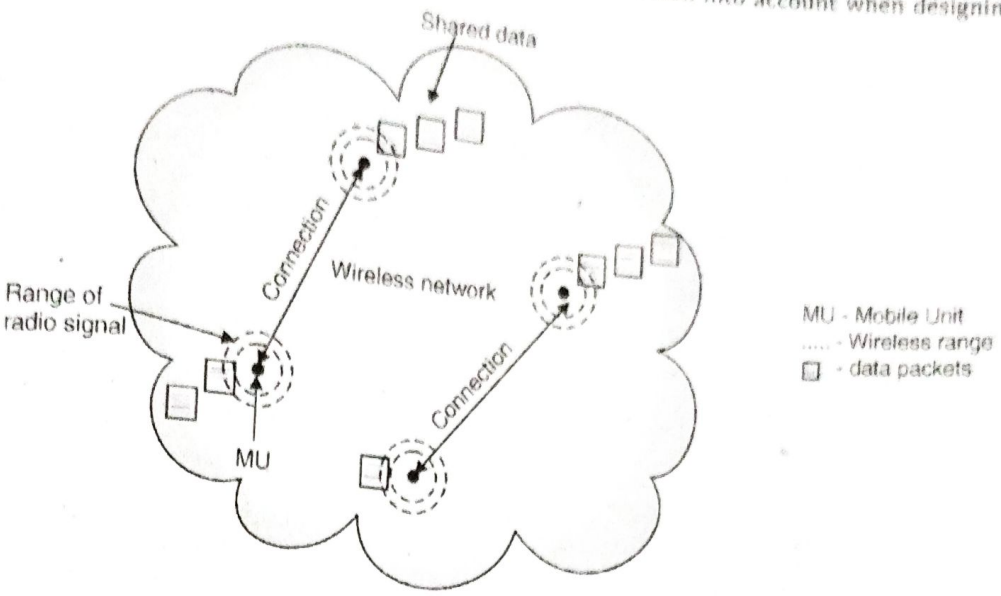


Fig. 16.2 : Mobile environment

1. Battery

- Applications must be designed to be as energy-efficient as possible due to the limited battery power of mobile devices.
- The mobile applications should therefore be designed to stop processes when idle and save CPU usage when possible.
- Battery life is directly related to battery size, and indirectly related to the mobile device's capabilities.
- Caching data can also reduce power consumption by eliminating the need to make energy consuming wireless data transmissions for each data access.

2. Data rate

- The wireless mediums on which mobile units and base stations communicate have bandwidths significantly lower than those of a wired network.
- Hence the data rate of mobile systems is generally low.

3. Client connectivity

- Applications must be designed to provide maximum mobility to mobile device or user.
- Applications are designed in wireless environment, have lower data rate and limited connectivity.
- Intermittent connectivity can be intentional or unintentional.
- Unintentional disconnections happen in areas where wireless signals cannot reach i.e. no coverage area.
- Intentional disconnections occur by user intent e.g., during an airplane takeoff, or when the mobile device is powered off.

4. Server connectivity

- Servers must keep track of client locations in order to efficiently send messages to them.
- Client data should be stored in the network location that minimizes the traffic.

5. Memory

- Applications must be designed for environments that have less storage space and dynamic memory than desktop environments; this can be done using memory-efficient data structures and reuse of code.
- Applications must also be robust, since users generally find applications that crash easily, very tedious to use, especially so because devices are usually always on.

6. **User interface**

- Applications must be designed for environments that have small screens and input methods that differ from those of the desktop environment, for example the 12-key numeric keypad, stylus, and soft keys
- In addition, there are different kinds of devices, which means that attention must be paid to the various display sizes, different keyboards, and the different look and feel of devices.

7. **Execution of applications**

- Applications can be interrupted by, for example, an SMS or a phone call.
- These events cause the application to be moved to the background. For situations such as these, care must be taken to ensure that the application can recover from the situation smoothly.

Q.3 Write a short note on Cloud database.

SPPU - May 18, May 19, Dec.19, 6 Marks

Ans. : Cloud Database

What is Cloud?

- Cloud refers to a network, internet or communication medium through which we can access computing resource located at remote location.
- Cloud can contain services like email, storage and web conferencing etc.

Cloud Features

1. **On demand services / self service**

- On-demand service refers to the service provided by cloud computing vendor enables user use cloud resources on demand whenever they actually need it.
- In on-demand self-service, the user makes use of online control panel to accesses cloud services.
- It is possible to upgrade or downgrade the cloud services as organisation needs it without investing lot of extra cost.

2. **Pay per use pricing model**

- Cloud Computing is affordable for start-up or small scale organisations to use infrastructure and services at lower cost without investing lot of upfront setup cost.
- User will pay only for computing resources which is actually in use. So, Organisation requires paying as they use computing resources.

3. **Quality of Services (QoS)**

- Cloud computing will provide best possible services to users. Cloud computing service guarantees 24x7 availability, provision of extra computing resources, Good performance.
- Customer can change cloud computing platform at any point of time based on the quality of services provided by vendor.

4. **Elasticity**

- Cloud computing offers elasticity in adopting changes in computing resources as workload changes by acquiring new resources and releasing extra resources in an autonomic manner.
- In general for cloud computing or for cloud application Cost, Quality, and Resources are basic parameters which require elasticity.

5. **Customization**

There are thousands of users are using cloud at Sam point of time. This environment requires lot of customization for every user for giving data security and simplicity for accessing data.

Q.4 Write a short note on SQLite database.

SPPU -Dec.17 , May 18, Dec. 19, 4 Marks

Ans. : SQLite Database

- SQLite is a free open souce RDBMS engine.
- SQLite is a self-contained, serverless, zero-configuration and transactional SQL embedded RDBMS engine.
- SQLite implements most of the SQL-92 standard for SQL.
- The SQLite library is small. It takes less than 500 KB space. Moreover SQLite database is a single file which we can easily store anywhere in the system.
- SQLite file is platform independent and can be easily portable. It supports many operating systems irrespective of their architectures i.e. 32 bit or 64 bit.

- C programming language is used to develop SQLite. It has support for many different and popular languages like Java, C++, C#, Python, VB, TCl etc.
- As it is an open source technology the source code of SQLite is easily available for public.
- It is very useful in memory constrained gadgets such as cell phones, PDAs, and MP3 players.
- Transactions are ACID even if interrupted by system crashes or power failures.

Features of SQLite

- Ensures ACID properties even after system crashes and power failures.
- No setup or administration needed.
- Small size.
- Portable and platform independent.
- Open source.
- Full SQL implementation with advanced features.
- Support for terabyte-sized databases and gigabyte-sized data.
- Easy and simple to use API.
- No external dependencies hence self contained.
- Readable code and available in public domain.

Applications of SQLite

Embedded devices and the internet of things

With no administration, SQLite can be easily used with cell phones, game consoles, televisions, set top boxes, watches, remote sensors, robots, automobiles, home appliancesetc.

Application file format

It is a cross platform file. It can be used on various operating systems, both 32 and 64 bit architectures.

Websites

SQLite works great with websites having low to medium traffic.

Data analysis

With full SQL implementation it can be used to analyze large datasets. Tcl or python can be used for more complex analysis.

Education and Training

Simple and easy to install and use. Open source, Very small in size. Hence very useful for education and training.

Prototyping and extensions

As it is lightweight, platform independent it is good for experimenting.

Good file archives

SQL lite can be used as a substitute for ZIP archives or Tarballs.

Q.5 Explain in brief the advantage of Mongo DB over RDBMS. **SPPU –Dec. 17, May 18, Dec. 18, 6 Marks**

Ans. : MongoDB

- MongoDB is an open-source document type database.
- MongoDB is implemented in C++.
- High performance, high availability and automatic scaling are the important features. MongoDB has its own ad-hoc query language with rich features set.
- Large numbers of use cases can be easily handled.
- Mobile applications, CMS (Content Management System), E-commerce, Gaming applications, Analytics, Archiving and Logging are the applications areas of MongoDB.
- Multi-document transactions are not possible in MongoDB.
- MongoDB does provide atomic operations on a single document.
- The biggest advantage of MongoDB is that, as a cache memory it automatically uses all free memory available on the machine.

Collection and Document

- Collection is equivalent to a table in RDBMS. A collection is a group of documents which exists within a single database. Collection is schemaless.
- Different fields can be created with different documents in a collection. But typically, all the documents in a single collection are of similar or related purpose.

- A document is a set of key-value pairs. Documents have dynamic schema.

RDBMS terminology with MongoDB teminology

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	Mongod
mysql/sqlplus	Mongo

Document Database

In MongoDB document is used to store data. Each document consists of fields and values. MongoDB documents and JSON objects are similar to each other. Other documents, arrays etc. can be included in the field values.

Documents have following advantages :

- Due to embedded documents and arrays, joins are avoided. Hence less expensive.
- Document uses dynamic schema.
- In popular programming languages documents are native data types.
- A MongoDB document is as follows,

```
{  movie : "Hanuman",
   Ticket : 100,
   Screen : 02,
}
```

MongoDB Schema

MongoDB uses dynamic schemas. Documents in a collection may have different set of fields. Due to this polymorphism can be easily used. Without defining the structure, i.e. the fields or the data types of the documents we can create a structure. Adding new field and deleting existing fileds can be easily done.

❑❑❑