CST204 Database Management System

Module 3

Introduction

Syllabus – Overview

- Module 1 : Introduction and Entity Relationship (ER) model
- Module 2 : Relational Model
- Module 3 : SQL DML and Physical Data Organization
- Module 4 : Normalization
- Module 5: Transaction, concurrency and recovery, recent topics

Introduction

Syllabus – Module 3

- SQL DML (Data Manipulation Language) –
  - SQL queries on single and multiple tables
  - Nested queries (correlated and non-correlated)
  - Aggregation and grouping
  - Views, assertions, Triggers, SQL data types
- Physical Data Organization –
  - Review of terms: physical and logical records
  - blocking factor, pinned and unpinned organization
  - Heap files, Indexing, Singe level indices, numerical examples
  - Multi-level-indices, numerical examples
  - B-Trees & B+-Trees (structure only, algorithms not required)
  - Extendible Hashing, Indexing on multiple keys – grid files

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

- SQL supports few set operations to be performed on table data
- Set operations are used to join the results of two or more SELECT statements
- The set operators available in SQL are UNION, UNION ALL, INTERSECT, EXCEPT (MINUS)

UNION

- UNION is used to combine the results of two or more SELECT statements
- However it will eliminate duplicate rows from its result set
- In case of union, number of columns and datatype must be same in both the tables

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

- Consider the table first

ID	Name
1	abhi
2	adam

Click to add text

- Consider the table second

ID	Name
2	adam
3	Chester

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

- The query *SELECT * FROM FIRST UNION SELECT * FROM SECOND* will result in the below result:

ID	NAME
1	abhi
2	adam
3	Chester

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

UNION ALL

- The UNION ALL is similar to UNION operation except the fact that it will list all table items including the duplicate items
- Consider table First as

ID	Name
1	abhi
2	adam

- Consider table Second as

ID	Name
2	adam
3	Chester

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

- The query *SELECT *FROM FIRST UNION ALL SELECT * FROM SECOND* will produce the below result:

ID	NAME
1	abhi
2	adam
2	adam
3	Chester

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

INTERSECT

- It is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement
- This means INTERSECT returns only common rows returned by the two SELECT statements
- Consider table FIRST as:

ID	NAME
1	abhi
2	adam

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

INTERSECT

- Consider table SECOND as:

ID	NAME
2	adam
3	Chester

- The query *SELECT * FROM FIRST INTERSECT SELECT * FROM SECOND* will produce the below result:

ID	NAME
2	adam

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

EXCEPT (MINUS)

- Minus operation combines result of two SELECT statements and return only those results which belongs to first set of results
- Consider table FIRST as:

ID	NAME
1	abhi
2	adam

- Consider table SECOND as:

ID	NAME
2	adam
3	Chester

SQL Data Manipulation Language (DML)

SQL Queries on Single and Multiple Tables

EXCEPT (MINUS)

- The query *SELECT * FROM FIRST EXCEPT SELECT * FROM SECOND* will produce the below result:

ID	NAME
1	Abhi

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

- An aggregate function allows us to perform a calculation on a set of values to return a single scalar value
- We often use aggregate function with GROUP BY and HAVING clauses of the SELECT statements
- Used to summarize the information from multiple tuples into single tuples
- The following are the most commonly used SQL aggregate functions:

  COUNT() – Counts rows in a specified table or view

  SUM() – Calculates the sum of values

  AVG() – Calculates the average of a set of values

  MIN() – Gets the minimum value in a set of values

  MAX() – Gets the maximum value in a set of values

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

COUNT()

- The COUNT() function returns the number of rows that matches a specified criteria
- Syntax:

  SELECT COUNT(column_name) FROM table_name WHERE condition;
- Example:

  Consider table named PRODUCT as:

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

COUNT()

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Cpm1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50
Item10	Com3	4	30	120

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

COUNT()

- SELECT COUNT(*) FROM PRODUCT;
- Output: 10
- SELECT COUNT(*) FROM PRODUCT WHERE RATE>=20;
- Output: 7
- SELECT COUNT(DISTINCT COMPANY) FROM PRODUCT;
- Output: 3
- SELECT COMPANY, COUNT(*) FROM PRODUCT GROUP BY COMPANY;
- Output: Com1 5

  Com2 3

  Com3 2

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

COUNT()

- SELECT COMPANY, COUNT(*) FROM PRODUCT GROUP BY COMPANY HAVING COUNT(*)>2;
- Output: Com1 5

  Com2 3

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

SUM()

- The SUM() function returns the total sum of a numeric column
- Syntax:
- SELECT SUM(column_name) FROM table_name WHERE condition;
- Example:
- SELECT SUM(COST) FROM PRODUCT;
- Output: 670
- SELECT SUM(COST) FROM PRODUCT WHERE QTY>3;
- Output: 320

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

SUM()

- SELECT SUM(COST) FROM PRODUCT WHERE QTY>3 GROUP BY COMPANY;
- Output: Com1 150

  Com2 170
- SELECT COMPANY, SUM(COST) FROM PRODUCT GROUP BY COMPANY HAVING SUM(COST)>=170;
- Output: Com1 335

  Com3 170

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

AVG()

- The AVG() function returns the average value of a numeric column
- Syntax:
- SELECT AVG(column_name) FROM table_name WHERE condition;
- Example:
- SELECT AVG(COST) FROM PRODUCT;
- Output: 67.00

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

MAX()

- The MAX() function returns the largest value of the selected column
- Syntax:
- SELECT MAX(column_name) FROM table_name WHERE condition;
- Example:
- SELECT MAX(RATE) FROM PRODUCT;
- Output: 30

SQL Data Manipulation Language (DML)

Aggregate Functions in SQL

MIN()

- The MIN() function returns the smallest value of the selected column
- Syntax:
- SELECT MIN(column_name) FROM table_name WHERE condition;
- Example:
- SELECT MIN(RATE) FROM PRODUCT;
- Output: 10

SQL Data Manipulation Language (DML)

Nested Queries in SQL

- A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause
- A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved
- Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, >=, <=, IN, BETWEEN etc
- Syntax:

SELECT * FROM table_name WHERE column_name operator (Nested Query);

- Example:

SELECT * FROM CUSTOMERS WHERE ID = (Nested Query);

SQL Data Manipulation Language (DML)

Nested Queries in SQL

- Sub queries must be enclosed within parantheses
- A sub query can have only one column in the SELECT clause, unless multiple columns are in the main query for the sub query to compare its selected columns
- An ORDER BY clause cannot be used in a subquery, although the main query can use an ORDER BY
- The GROUP BY command can be used to perform the same function as the ORDER BY in a sub query
- Sub queries that return more than one row can only be used with multiple value operators such as the IN operator
- A subquery cannot be immediately enclosed in a set function

SQL Data Manipulation Language (DML)

Nested Queries in SQL

Sub queries with SELECT statement

- Subqueries are most frequently used with the SELECT statement
- The basic syntax is as follows –
- SELECT column_name FROM table_name1 WHERE column_name OPERATOR (SELECT column_name FROM table_name1 WHERE condition)
- For example, Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

SQL Data Manipulation Language (DML)

Nested Queries in SQL

- SELECT * FROM CUSTOMERS WHERE ID IN (SELECT ID FROM CUSTOMERS WHERE SALARY > 4500) ;
- This will produce the following result as:

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

SQL Data Manipulation Language (DML)

Nested Queries in SQL

Subqueries with the INSERT Statement

- The INSERT statement uses the data returned from the subquery to insert into another table
- The selected data in the subquery can be modified with any of the character, date or number functions
- The basic syntax is as follows:
- INSERT INTO table_name (column1 , column2) SELECT * FROM table1 WHERE VALUE OPERATOR

SQL Data Manipulation Language (DML)

Nested Queries in SQL

- Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table
- Now to copy the complete CUSTOMERS table into the CUSTOMERS_BKP table, you can use the following syntax
- SQL> INSERT INTO CUSTOMERS_BKP SELECT * FROM CUSTOMERS WHERE ID IN (SELECT ID FROM CUSTOMERS) ;

SQL Data Manipulation Language (DML)

Nested Queries in SQL

Subqueries with the UPDATE Statement

- The subquery can be used in conjunction with the UPDATE statement
- Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement
- The basic syntax is as follows:
- UPDATE table SET column_name = new_value WHERE OPERATOR [VALUE] (SELECT COLUMN_NAME FROM TABLE_NAME)[WHERE)]
- For example, assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table

SQL Data Manipulation Language (DML)

Nested Queries in SQL

Subqueries with the UPDATE Statement

- The following example updates SALARY by 0.25 times in the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27
- SQL> UPDATE CUSTOMERS SET SALARY = SALARY * 0.25 WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP WHERE AGE >= 27);
- This would impact two rows and finally CUSTOMERS table would have the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	125.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	2125.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

SQL Data Manipulation Language (DML)

Nested Queries in SQL

Subqueries with the DELETE Statement

- The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above
- The basic syntax is as follows:
- DELETE FROM TABLE_NAME [WHERE OPERATOR [VALUE] (SELECT COLUMN_NAME FROM TABLE_NAME) [WHERE)]
- Assuming, we have a CUSTOMERS_BKP table available which is a backup of the CUSTOMERS table
- The following example deletes the records from the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27

SQL Data Manipulation Language (DML)

Nested Queries in SQL

Subqueries with the DELETE Statement

- SQL> DELETE FROM CUSTOMERS WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP WHERE AGE >= 27);
- This would impact two rows and finally the CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

SQL Data Manipulation Language (DML)

Views in SQL

- A view is a single table that is derived from other tables
- These other tables can be base tables or previously defined views
- A view does not necessarily exist in physical forms
- It is considered to be a virtual table
- A view is nothing more than a SQL statement that is stored in the database with an associated name
- A view is actually a composition of a table in the form of a predefined SQL query
- A view can be created from one or many tables which depends on the written SQL query to create a view

SQL Data Manipulation Language (DML)

Views in SQL

- Views, which are a type of virtual tables allow users to do the following –
- Structure data in a way that users or classes of users find natural or intuitive
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more
- Summarize data from various tables which can be used to generate reports

Creating Views

- Database views are created using the CREATE VIEW statement
- Views can be created from a single table, multiple tables or another view
- To create a view, a user must have the appropriate system privilege according to the specific implementation

SQL Data Manipulation Language (DML)

Views in SQL

Creating Views

- The basic CREATE VIEW syntax is as follows –
- CREATE VIEW view_name AS SELECT column1, column2..... FROM table_name WHERE [condition];
- For example consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

SQL Data Manipulation Language (DML)

Views in SQL

Creating Views

- Following is an example to create a view from the CUSTOMERS table.
- This view would be used to have customer name and age from the CUSTOMERS table
- SQL > CREATE VIEW CUSTOMERS_VIEW AS SELECT name, age FROM CUSTOMERS;
- Now, we can query CUSTOMERS_VIEW in a similar way as we query an actual table
- Following is an example for the same:
- SQL > SELECT * FROM CUSTOMERS_VIEW;

SQL Data Manipulation Language (DML)

Views in SQL

Creating Views

- This would produce the following result as:

```
+----------+-----+
| name     | age |
+----------+-----+
| Ramesh   |  32 |
| Khilan   |  25 |
| kaushik  |  23 |
| Chaitali |  25 |
| Hardik   |  27 |
| Komal    |  22 |
| Muffy    |  24 |
+----------+-----+
```

SQL Data Manipulation Language (DML)

Views in SQL

Updating Views

- Database views can be updated using UPDATE statement
- The basic syntax for update a view is as follows:
- UPDATE view_name SET columnName = value WHERE condition;
- For example, the following query has an example to update the age of Ramesh.
- SQL > UPDATE CUSTOMERS_VIEW SET AGE = 35 WHERE name = 'Ramesh';
- This would ultimately update the base table CUSTOMERS and the same would reflect in the view itself
- Now, try to query the base table and the SELECT statement would produce the following result:

SQL Data Manipulation Language (DML)

Views in SQL

Updating Views

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

SQL Data Manipulation Language (DML)

Views in SQL

Inserting rows into Views

- Rows of data can be inserted into a view
- The same rules that apply to the UPDATE command also apply to the INSERT command
- Here, we cannot insert rows in the CUSTOMERS_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in a similar way as you insert them in a table

SQL Data Manipulation Language (DML)

Views in SQL

Deleting rows in Views

- Rows of data can be deleted from a view
- The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command
- Following is an example to delete a record having AGE = 22
- SQL > DELETE FROM CUSTOMERS_VIEW WHERE age = 22;
- This would ultimately delete a row from the base table CUSTOMERS and the same would reflect in the view itself

SQL Data Manipulation Language (DML)

Views in SQL

Deleting rows in Views

- Now, try to query the base table and the SELECT statement would produce the following result

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

SQL Data Manipulation Language (DML)

Views in SQL

Dropping Views

- Obviously, where you have a view, you need a way to drop the view if it is no longer needed
- The syntax is very simple and is given below −

```
DROP VIEW view_name;
```

- Following is an example to drop the CUSTOMERS_VIEW from the CUSTOMERS table

```
DROP VIEW CUSTOMERS_VIEW;
```

SQL Data Manipulation Language (DML)

Views in SQL

Advantages of Views

- Restrict data access and/or simplify data access
- Simplify data manipulation
- Import and export data
- Merge data

SQL Data Manipulation Language (DML)

Assertions in SQL

- Assertions are used to specify general restrictions on data stored in tables
- These restrictions cannot be expressed using integrity constraints
- Syntax:

  CREATE ASSERTION assertion_name CHECK (condition);

- Suppose we want to make sure that the salary of an employee does not exceed that of his/her manager, then the following query can be used

CREATE ASSERTION SALARY_CONSTRIANTS CHECK (NOT EXISTS (SELECT * FROM EMPLOYEE E, EMPLOYEE M, DEPARTMENT D WHERE E. SALARY > M.SALARY AND E.DNO = D.NUMBER AND D.MGRSSN = M.SSN));

- The assertion part comes within the CHECK clause
- For every update on salary of the employee, the database checks the condition given by the assertion and alarms if it fails

TUTORIAL NO 2

- **Consider the following relations:**
- **FACULTY(FNO, NAME, GENDER, AGE, SALARY, DNUM)**
- **DEPARTMENT(DNO, DNAME, DPHONE)**
- **COURSE(CNO, CNAME, CREDITS, ODNO)**
- **TEACHING(FNO, CNO, SEMESTER)**
- **DNUM is a foreign key that identifies the department to which a faculty belongs**
- **ODNO is a foreign key identifying the department that offers a course**

- **Write SQL expressions for the following queries:**
- Names and department number of faculty members
- Names and Phone number of department No.2
- Names and credits of Course no. 5
- Faculty number and corresponding course no of 4th semester

TUTORIAL NO 2

- Names and department number of faculty members

  SELECT NAME, DNUM FROM FACULTY;

- Names and Phone number of department No.2

  SELECT DNAME, DPHONE FROM DEPARTMENT WHERE DNO = 2;

- Names and credits of Course no. 5

  SELECT CNAME, CREDIT FROM COURSE WHERE CNO = 5;

- Faculty number and corresponding course no of 4th semester

  SELECT FNO, CNO FROM TEACHING WHERE SEMESTER = 4;

TUTORIAL NO 2

- **Consider the following relations:**
- **FACULTY(FNO, NAME, GENDER, AGE, SALARY, DNUM)**
- **DEPARTMENT(DNO, DNAME, DPHONE)**
- **COURSE(CNO, CNAME, CREDITS, ODNO)**
- **TEACHING(FNO, CNO, SEMESTER)**
- **DNUM is a foreign key that identifies the department to which a faculty belongs**
- **ODNO is a foreign key identifying the department that offers a course**

- **Write SQL expressions for the following queries:**
- Names and department name of faculty members
- Names of faculty members not offering any course
- Names of departments offering more than three courses in alphabetic order

TUTORIAL NO 2

- Names and department names of faculty members

SELECT NAME, DNAME FROM FACULTY, DEPARTMENT WHERE DNUM = DNO;

- Names of faculty members not offering any course

SELECT NAME FROM FACULTY WHERE FNO NOT IN (SELECT FNO FROM TEACHING);

- Names of departments offering more than three courses in alphabetic order

SELECT DNAME FROM DEPARTMENT WHERE DNO IN (SELECT ODNO FROM COURSE GROUP BY ODNO HAVING COUNT(CNO) > 3);

TUTORIAL NO 2

- **Consider the following relations:**
- **FACULTY(FNO, NAME, GENDER, AGE, SALARY, DNUM)**
- **DEPARTMENT(DNO, DNAME, DPHONE)**
- **COURSE(CNO, CNAME, CREDITS, ODNO)**
- **TEACHING(FNO, CNO, SEMESTER)**
- **DNUM is a foreign key that identifies the department to which a faculty belongs**
- **ODNO is a foreign key identifying the department that offers a course**

- **Write SQL expressions for the following queries:**
- Course numbers and names of 3-credit courses offered by CSE department
- Names of faculty members teaching maximum 3 courses
- Names of departments along with number of courses offered by each of them, in the increasing order of number of courses

TUTORIAL NO 2

- Course numbers and names of 3-credit courses offered by CSE department

SELECT CNO, CNAME FROM COURSE WHERE CREDITS = 3 AND ODNO = (SELECT DNO FROM DEPARTMENT WHERE DNAME = “CSE”);

- Names of faculty members teaching maximum 3 courses

SELECT NAME FROM FACULTY, TEACHING, COURSE WHERE FACULTY.FNO = TEACHING.FNO AND TEACHING.CNO = COURSE.CNO HAVING COUNT(CNO) <= 3;

- Names of departments along with number of courses offered by each of them, in the increasing order of number of courses

SELECT DNAME, COUNT(CNO) FROM DEPARTMENT, COURSE WHERE DNO = ODNO GROUP BY DNAME ORDERBY COUNT(CNO);

SQL Data Manipulation Language (DML)

Triggers in SQL

- Triggers are stored program, which automatically executed or fired when some events occur
- Triggers are in fact written to be executed in response to any of the following events:
  - A DML statement (INSERT, UPDATE or DELETE)
  - A DDL statement (CREATE, ALTER or DROP)
  - A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN)
- Triggers can be defined on the table, view, schema, or database with which the event is associated

SQL Data Manipulation Language (DML)

Triggers in SQL

- Syntax:
- CREATE TRIGGER [trigger_name] [BEFORE | AFTER]

INSERT | UPDATE | DELETE} ON [table_name] [FOR EACH ROW]

[trigger_body]

- BEFORE triggers run the trigger action before the triggering statement is run
- AFTER triggers run the trigger action after the triggering statement is run
- For Example:
- Given Student Report Database, in which student marks assessment is recorded
- In such schema, create a trigger so that the total and average of specified marks is automatically inserted whenever a record is insert

SQL Data Manipulation Language (DML)

Triggers in SQL

- Here, as trigger will invoke before record is inserted so, BEFORE Tag can be used

```
mysql> desc Student;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| tid   | int(4)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30) | YES  |     | NULL    |                |
| subj1 | int(2)      | YES  |     | NULL    |                |
| subj2 | int(2)      | YES  |     | NULL    |                |
| subj3 | int(2)      | YES  |     | NULL    |                |
| total | int(3)      | YES  |     | NULL    |                |
| per   | int(3)      | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
```

SQL Data Manipulation Language (DML)

Triggers in SQL

create trigger stud_marks

before INSERT

on

Student

for each row

set Student.total = Student.subj1 + Student.subj2 + Student.subj3, Student.per = Student.total * 60 / 100;

SQL Data Manipulation Language (DML)

Triggers in SQL

- Above SQL statement will create a trigger in the student database in which whenever subjects marks are entered, before inserting this data into the database, trigger will compute those two values and insert with the entered values. i.e.,
- mysql> insert into Student values(0, "ABCDE", 20, 20, 20, 0, 0);

```
mysql> select * from Student;
+-----+-------+-------+-------+-------+-------+------+
| tid | name  | subj1 | subj2 | subj3 | total | per  |
+-----+-------+-------+-------+-------+-------+------+
| 100 | ABCDE |    20 |    20 |    20 |    60 |   36 |
+-----+-------+-------+-------+-------+-------+------+
```

SQL Data Manipulation Language (DML)

Data Types in SQL

- There are three main data types in SQL
  - String
  - Numeric
  - Date and Time
- String data types include:
  - CHAR(size)
  - VARCHAR(size)
  - BINARY(size)
  - TEXT(size)
  - LONGTEXT etc

SQL Data Manipulation Language (DML)

Data Types in SQL

- Numeric data types include:
  - BIT(size)
  - INT(size)
  - FLOAT(size, d)
  - DOUBLE(size, d)
  - DECIMAL(size, d) etc
- Date and Time data types include:
  - DATE
  - TIMESTAMP()
  - TIME()
  - YEAR etc

SQL Data Manipulation Language (DML)

Physical Data Organization

- The complete DBMS creation is based on the following step:
- Software Requirement Specification (SRS)

  ↓
- Relational Model

  ↓
- Normalization

  ↓
- File Structure (Indexing & Physical Structure)
- Indexing and Physical structure is how data is stored in physical form
- Indexing is like index of a book to access data quickly
- Physical structure is the actual structure of the data which is stored in some data structures like B tree or B+ tress

SQL Data Manipulation Language (DML)

Physical Data Organization

- Databases are stored physically on storage devices and organised as files and records
- The overall performance of a database system is determined by the physical database organization
- A file is a sequence of records
- A file with a given record structure consisting of several fields or attributes
- In many cases, all records in a file are of the same record type
- If every record in the file has exactly the same size (in bytes), the file is said to be made up of fixed length records
- If different records in the file have different sizes, the file is said to be made up of variable-length records

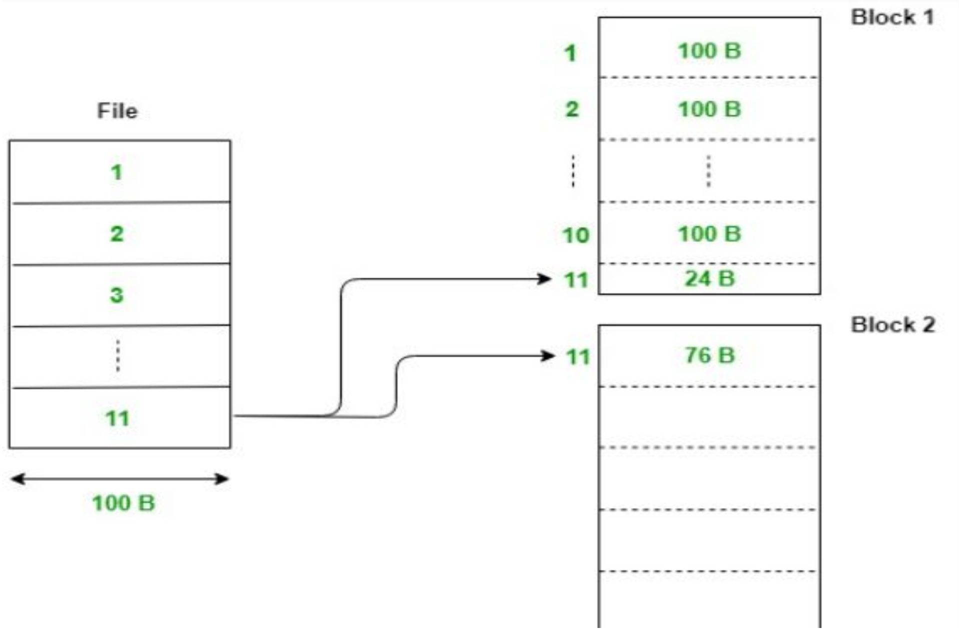
SQL Data Manipulation Language (DML)

Physical Data Organization – Spanned and Un-spanned organization

- There are various strategies ways to organize file records into blocks of disk:

Spanned Organization

- The record of a file is stored inside the block even if it can only be stored partially and hence, the record is spanned over two blocks giving it the name Spanned Organization



SQL Data Manipulation Language (DML)

Physical Data Organization – Spanned and Un-spanned organization

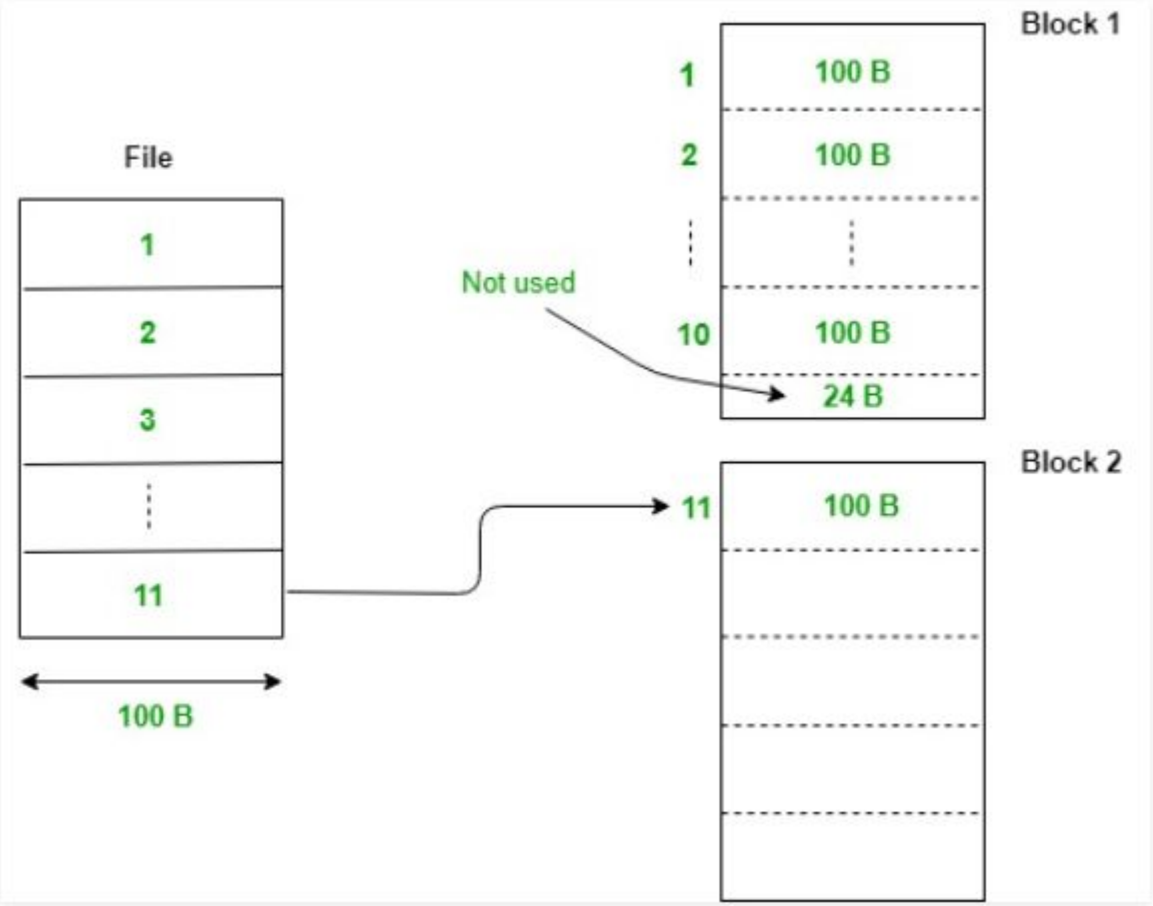
- Advantages: No wastage of memory (no internal fragmentation)
- Disadvantages: The record which has been spanned, while accessing it we would be required to access two blocks and searching time of a block is greater than the searching time of a record inside a block as the number of blocks on the disk are too large

Un-spanned Organization

- In un-spanned organization, unlike spanned strategy, the record of a file is stored inside the block only if it can be stored completely inside it
- Advantages: Access time of a record is less
- Disadvantages: Wastage of memory is more (internal fragmentation)

SQL Data Manipulation Language (DML)

Physical Data Organization – Spanned and Un-spanned organization



SQL Data Manipulation Language (DML)

Physical Data Organization – Spanned and Un-spanned organization

- The records of a file must be allocated to disk blocks because a block is the unit of data transfer between disk and memory
- When the block size is larger than the record size, each block will contain numerous records, although some files may have unusually karge records that cannot fit in one block
- Suppose that the block size is B bytes
- For a file of fixed-length records of size R bytes, with B>=R, we can fit **bfr = B/R** records per block
- The **bfr** is called the **blocking factor** for the file

SQL Data Manipulation Language (DML)

Physical Data Organization - Physical Files and Logical Files

Physical Files

- Contains the actual data that is stored on the system
- Also contain a description of how data is to presented to or received from a program

Logical Files

- Do not contain data
- They contain a description of records found in one or more physical files

SQL Data Manipulation Language (DML)

Physical Data Organization - Pinned Records and Unpinned Records

Pinned Record

- A record is said to be pinned down or pinned record if there exists a pointer to it somewhere in the database
- For example, when a table look up approach is used to locate a record, the table contains a pointer to the record and the record becomes pinned down
- The pinned records cannot be moved without reason randomly because in that case the pointers pointing to these records will dangle
- Any movement of pinned records should be associated with appropriate modification of the pointers
- In fact, the file organization method which maintain pointers to pinned records, appropriate modify these pointer whenever the records are inserted or deleted

SQL Data Manipulation Language (DML)

Physical Data Organization - Pinned Records and Unpinned Records

Un-pinned Record

- A record is said to be unpinned record, if there does not exist any pointer pointing to it in the database
- It is the independent record

SQL Data Manipulation Language (DML)

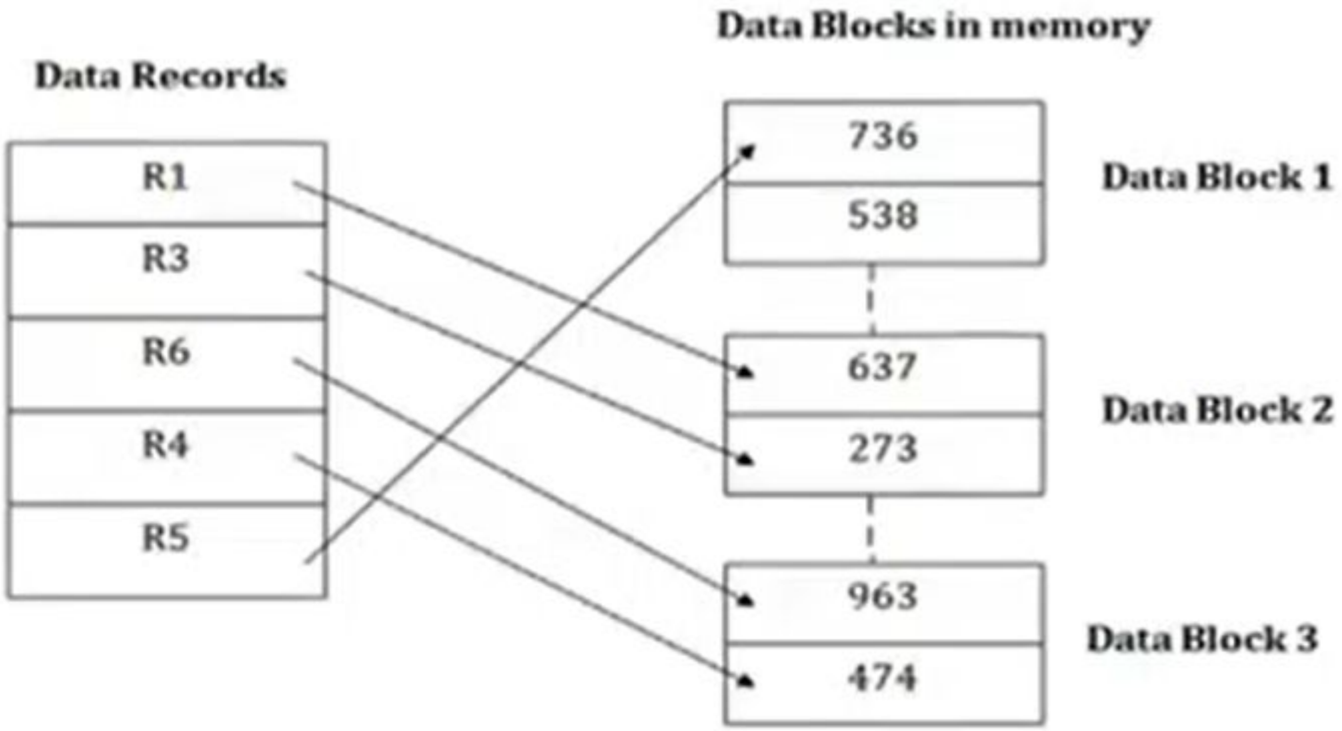
Physical Data Organization – Heap Files

- A heap file is an unordered set of records
- Heap file organization means that any record can be placed wherever there is space for that record, as there is no ordering of records
- Sequential file organization means that records are stored in a sequential order according to a search key, as the records are ordered
- The following operations are supported by Heap files:
  - Heap files can be created and destroyed
  - Existing heap files can be opened and closed
  - Records can be inserted and deleted
  - Records are uniquely identified by a record id (rid)
- A specific record can be retrieved by using the record id

SQL Data Manipulation Language (DML)

Physical Data Organization – Heap Files

- Sequential scans on heap file are also supported



SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

- Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done
- An index on a database table provides a convenient mechanism for locating a row (data record) without scanning the entire table and thus greatly reduces the time it takes to process a query
- The index is usually specified on one field of the file
- One form of an index is a file of entries **<field value, pointer to record>**, which is ordered by field value
- The index file usually occupies considerably less disk blocks than the data file because its entries are much smaller

SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

- Indexes can be characterized as dense and sparse
- A dense index has an index entry for every search key value (and hence every record) in that data file
- A sparse (or non-dense) index, on the other hand, has index entries for only some of the search values

Advantages

- Stores and organizes data into computer files
- Makes it easier to find and access data at any given time
- It is a data structure that is added to a file to provide faster access to the data

Disadvantages

- Index needs to be updated periodically for insertion or deletion of records in the main table

SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Structure of index

- An index is a small table having only two columns
- The first column contains a copy of the primary or candidate key of a table
- The second column contains a set of pointers holding the address of the disk block where that particular key value can be found
- If the indexes are sorted, then it is called as ordered indices
- Indexes are categorized into single level and multilevel indices
- Single Level is classified into follows:
  - Primary Index
  - Clustered Index
  - Secondary Index

SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Structure of index

Primary Index

- Primary Index is an ordered file which is fixed length size with two fields
- The first field is the same a primary key and second, filed is pointed to that specific data block
- In the primary Index, there is always one to one relationship between the entries in the index table
- Each block in the data file has one entry in the index file
- The two fields <K(i), P(i)>; P(i) is the pointer for the block in data file

SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Structure of index

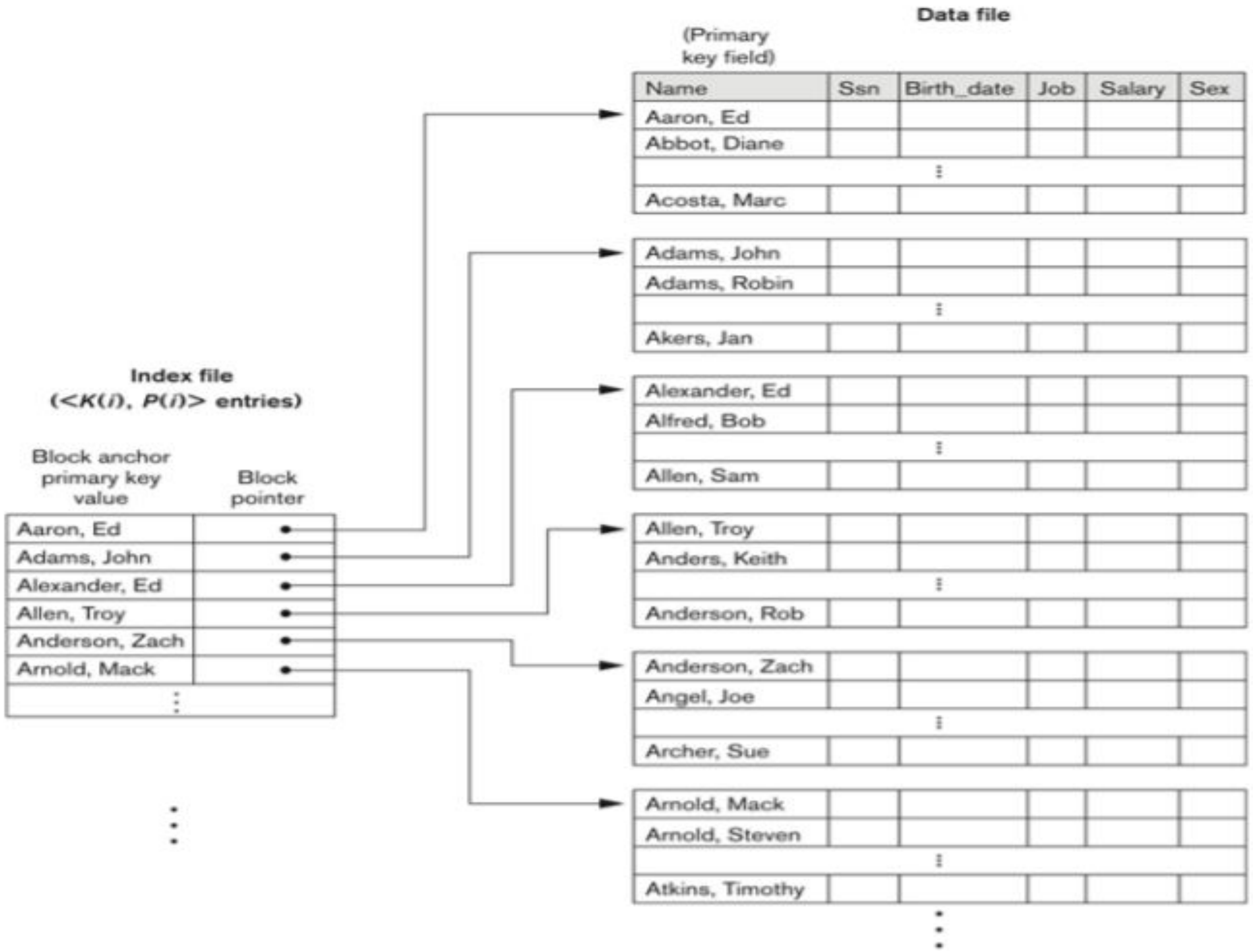
Primary Index

- First record in each block of the data file is called an block anchor or anchor record
- The number of index entries is equal to number of blocks
- Number of access required $= \log_2 n + 1$
- Indexes can be dense or parse
- A dense index has an entry for every search key value; a sparse index has index entries for only for some of the search values
- To retrieve a record given the value of its PK field, we do a binary search on the index file to find appropriate entry, and then retrieve the data field block whose address is P(i)

SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Structure of index



SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Structure of index

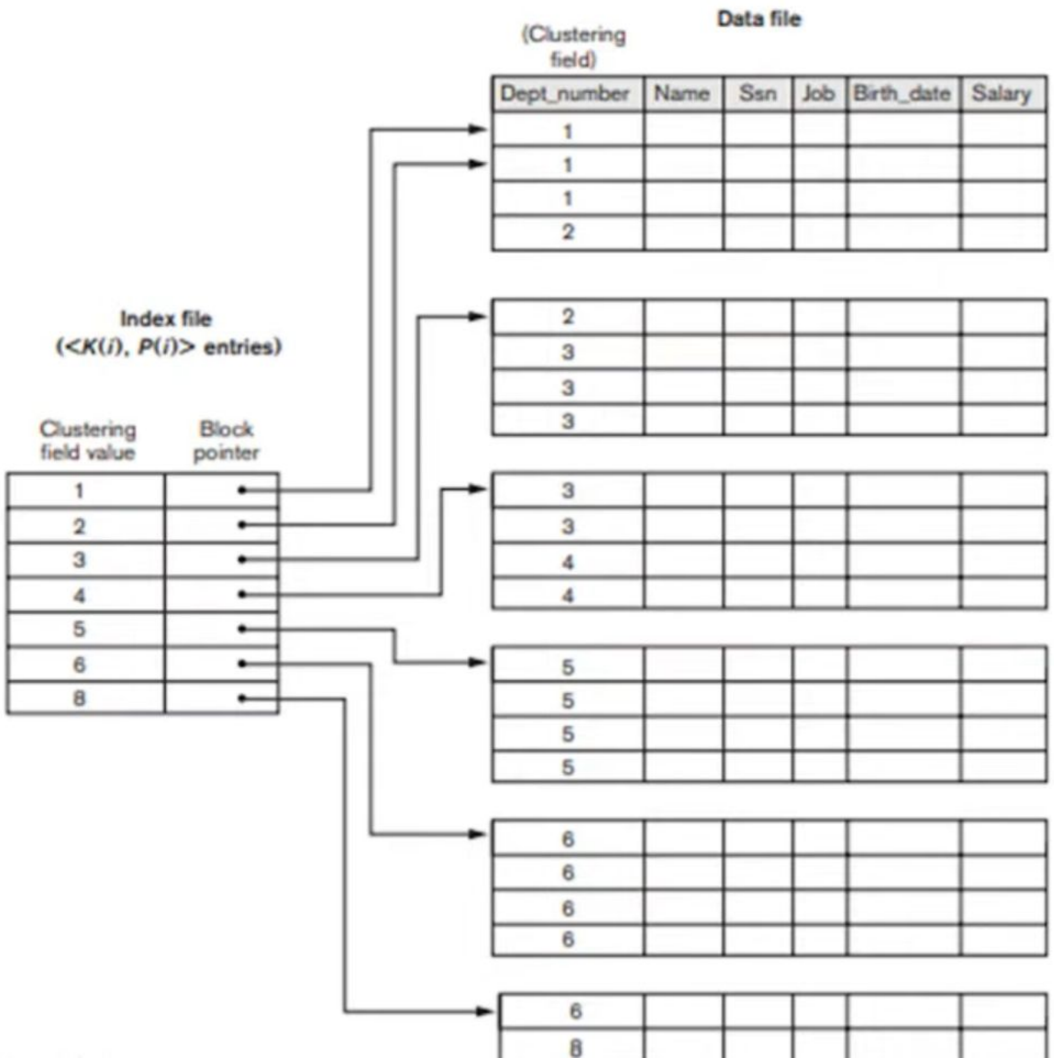
Clustered Index

- A clustering index is also an ordered file with two fields – clustering field and block pointer
- If file records are physically ordered on a non key field – which does not have a distinct value for each record that field is called the clustering field and the data file is called clustered file
- For example, students studying in each semester are grouped together. i.e. 1st Semester students, 2nd semester students, 3rd semester students etc are grouped

SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Structure of index



SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Structure of index

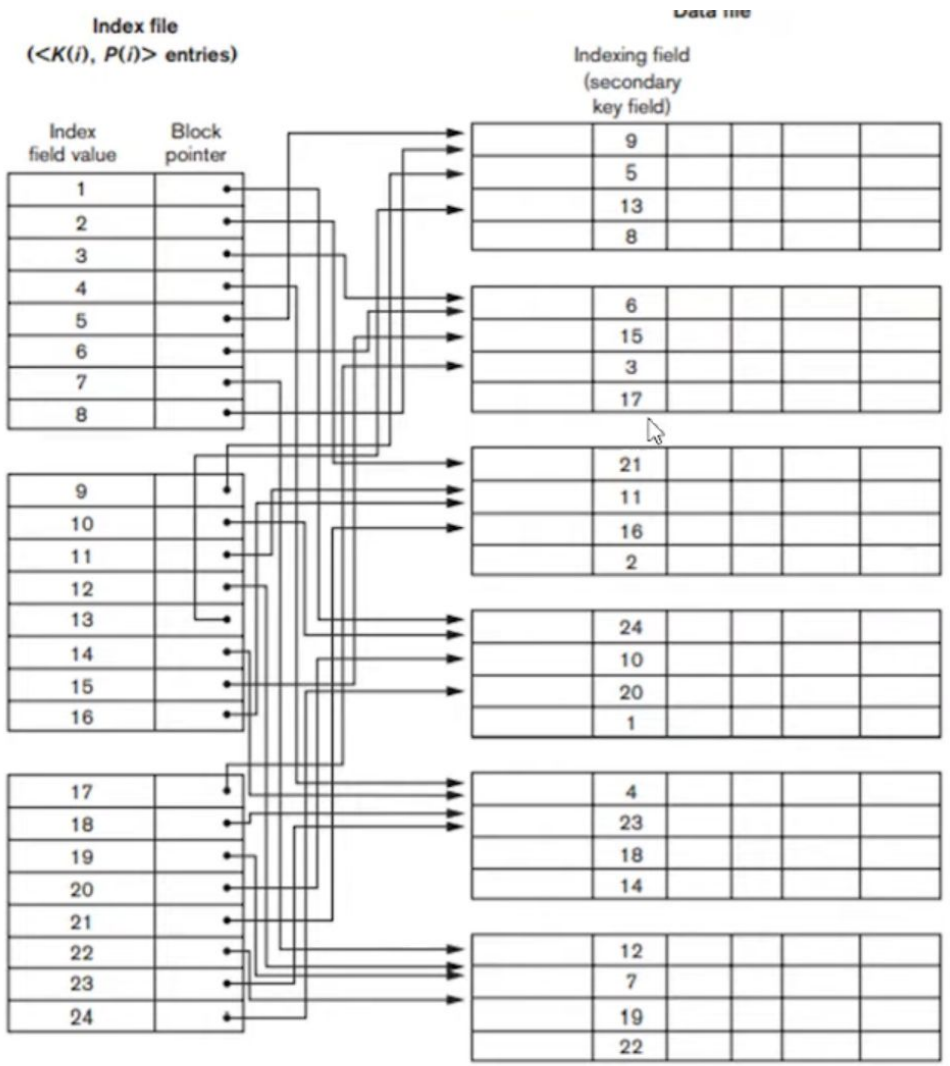
Secondary Index

- The secondary index may be created on a field that is a candidate key and has a unique value in every record, or on a non key field with duplicate values
- The number of entries in index file is equal to number of entries in main file

SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Structure of index



SQL Data Manipulation Language (DML)

Physical Data Organization – Index Structures

Example:

Suppose we have an ordered file with 30,000 records and stored on a disk of block size 1024 bytes and records are of fixed size, unspanned organisation. Record length = 100 bytes. How many block access needed to search a record.

No. of records,	r	$= 30{,}000$
Block size,	B	$= 1024$ bytes
Record size,	R	$= 100$ bytes
Blocking factor,	bfr	$= \lfloor B / R \rfloor = \lfloor 1024 / 100 \rfloor = 10$ records/block
No. blocks,	b	$= \lceil r / bfr \rceil = \lceil 30{,}000 / 10 \rceil = 3000$ blocks
If linear search		$= 3000 / 2 = 1500$ block access
If binary search		$= \log2(3000) = 12$ block access

SQL Data Manipulation Language (DML)

Physical Data Organization

Multilevel Indexing - Trees

- A tree is formed of nodes
- Each node in the tree, except for a special node called the root, has one parent node and zero or more child nodes
- The root node has no parent
- A node that does not have any child nodes is called a leaf node; a non-leaf node is called an internal node
- The level of a node is always one more than the level of its parent, with the level of the root node being zero

SQL Data Manipulation Language (DML)

Physical Data Organization

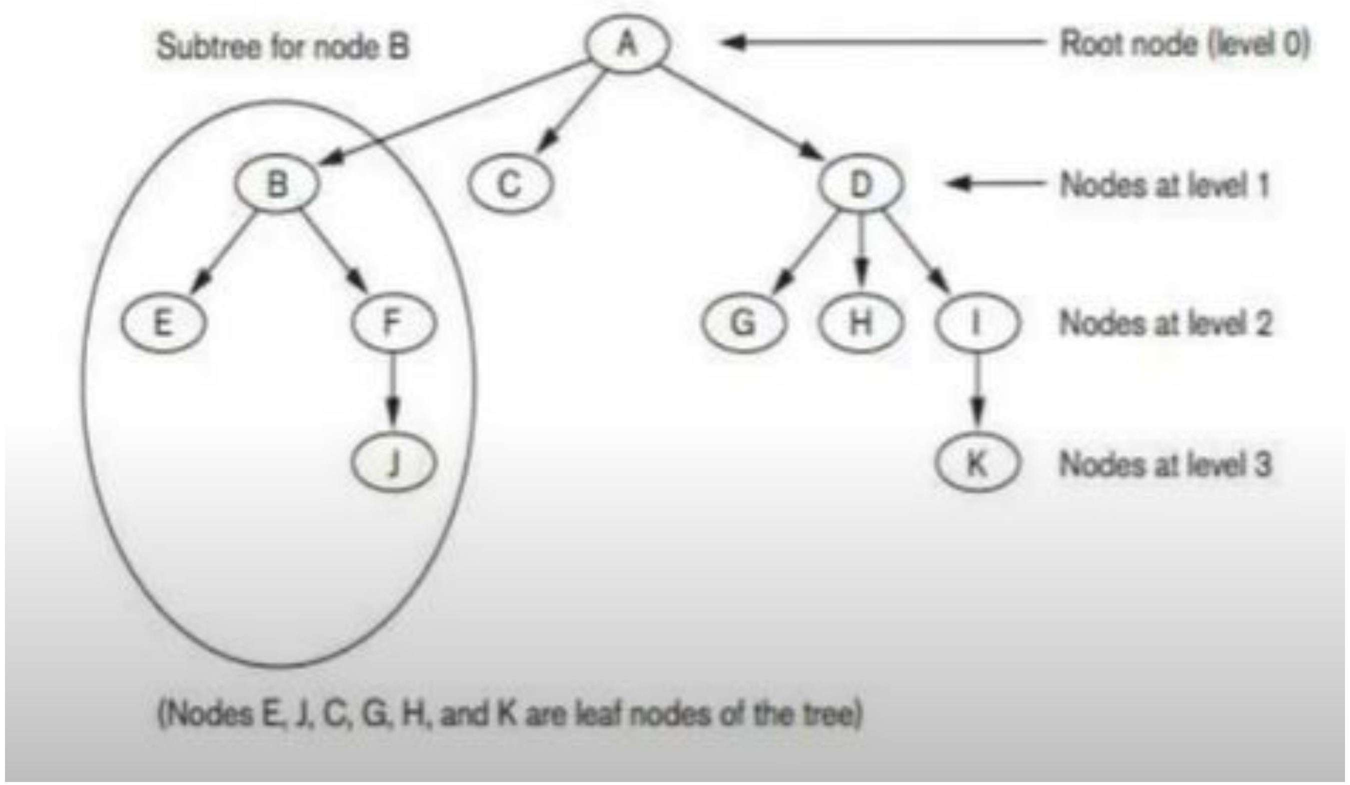
Trees

- A subtree of a node consists of that node and all its descendant nodes – its child nodes, the child nodes of its child nodes, and so on
- A precise recursive definition of a subtree is that it consists of a node n and the subtrees of all the child nodes of n
- The leaf nodes are at different levels of the tree, this tree is called unbalanced

SQL Data Manipulation Language (DML)

Physical Data Organization

Trees



SQL Data Manipulation Language (DML)

Physical Data Organization

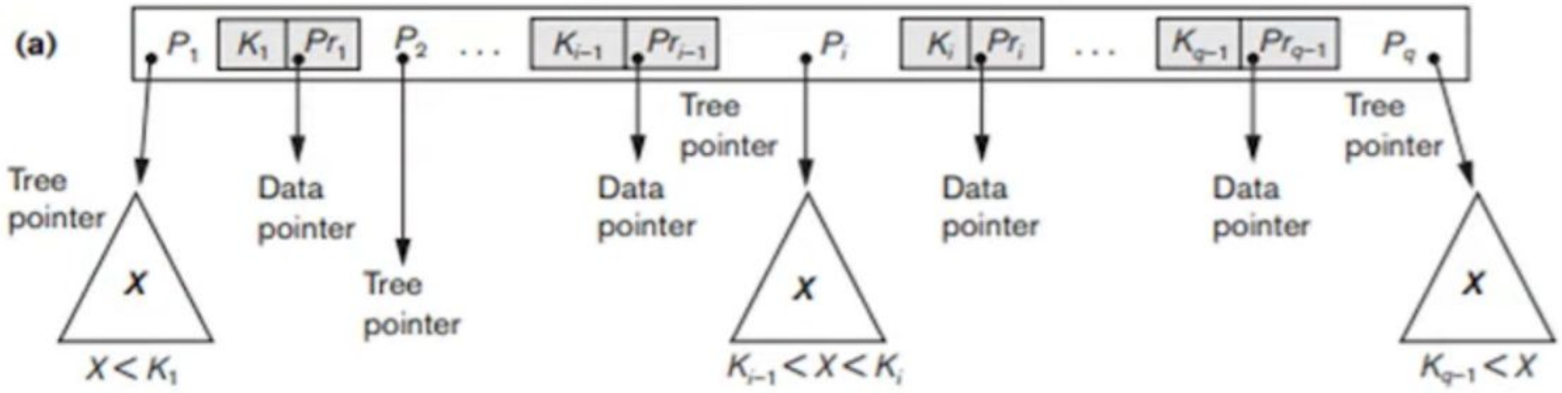
B-Trees

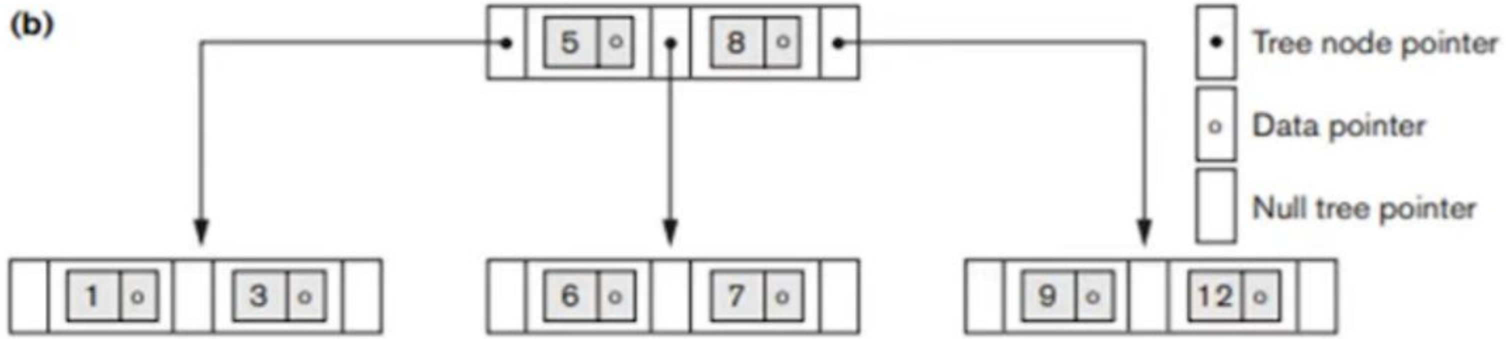
- A B Tree is known as a self balancing tree
- In B tree, a node can have more than two children
- In the B tree data is sorted in a specific order, with the lowest value on the left and the highest value on the right
- To insert the data or key in B tree is more complicated than a binary tree
- All the leaf nodes of the B tree must be at the same level

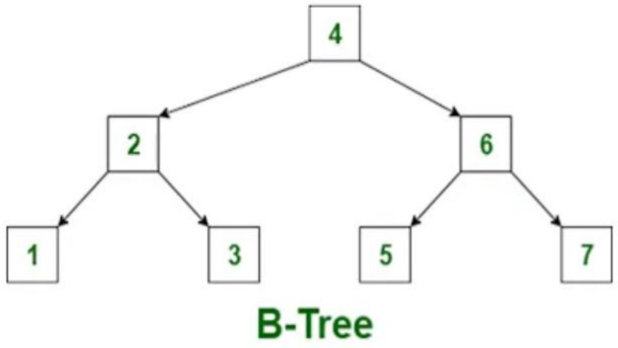
SQL Data Manipulation Language (DML)

Physical Data Organization

B-Trees







SQL Data Manipulation Language (DML)

Physical Data Organization

B+ Trees

- In order, to implement dynamic multilevel indexing, B-tree and B+ tree are generally employed
- The drawback of B-tree used for indexing, however is that it stores the data pointer (a pointer to the disk file block containing the key value), corresponding to a particular key value, along with that key value in the node of a B-tree
- This technique, greatly reduces the number of entries that can be packed into a node of a B-tree, thereby contributing to the increase in the number of levels in the B-tree, hence increasing the search time of a record
- B+ tree eliminates the above drawback by storing data pointers only at the leaf nodes of the tree.
- Thus, the structure of leaf nodes of a B+ tree is quite different from the structure of internal nodes of the B tree.

SQL Data Manipulation Language (DML)

Physical Data Organization

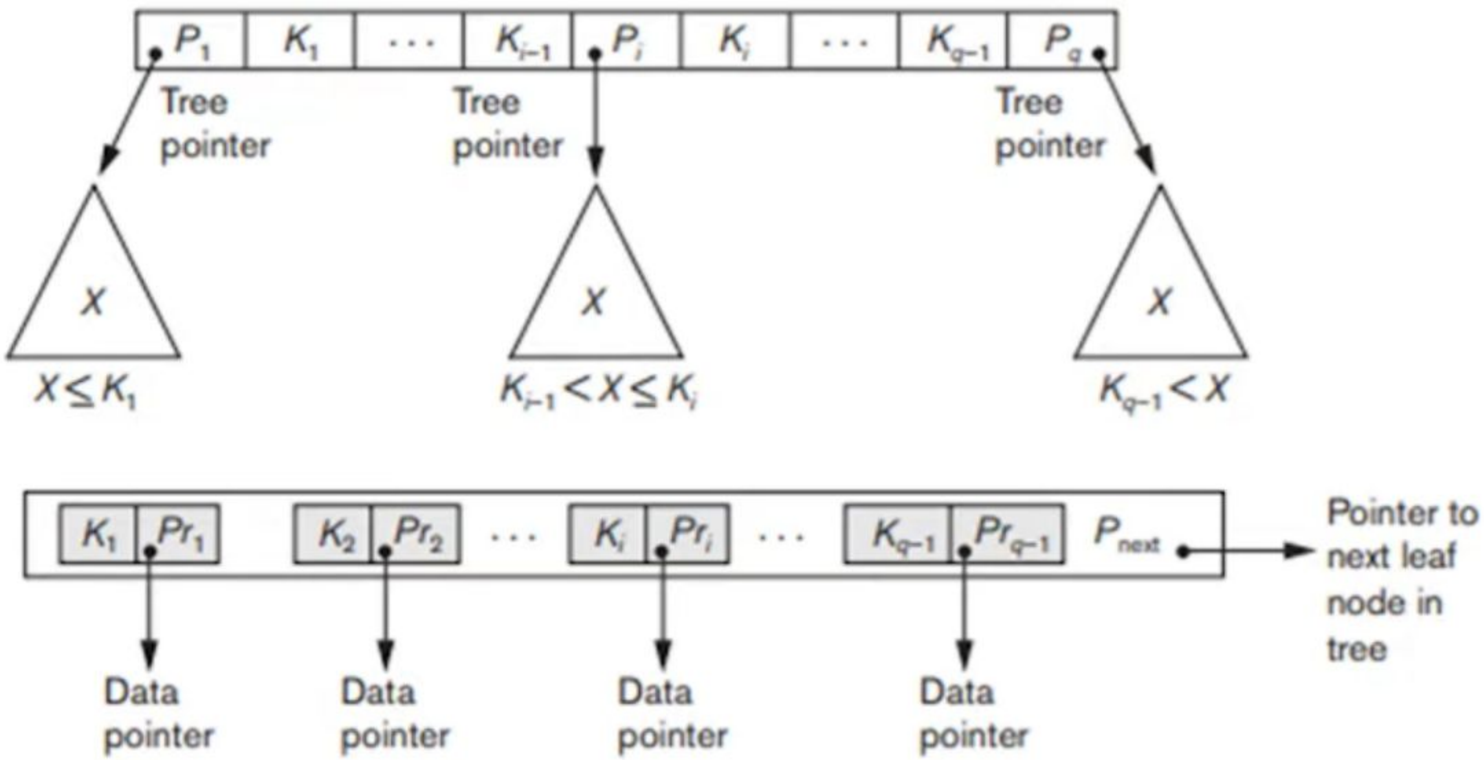
B+ Trees

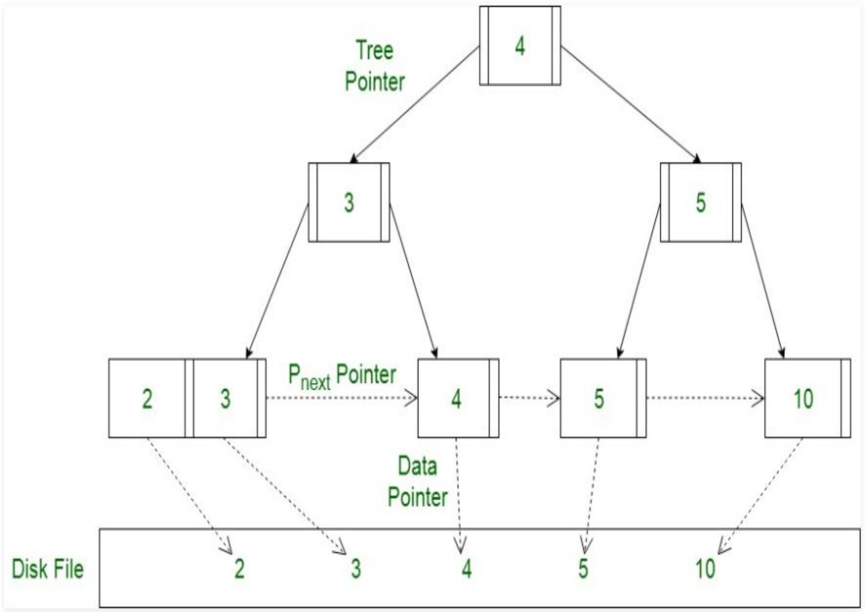
- A B+ Tree is a balanced binary search tree
- It stores data pointers only at the leaf nodes of the tree
- The leaf nodes must necessarily store all the key values along with their corresponding data pointers to the disk file block in order to access them
- Leaf nodes are linked to provide ordered access to the records
- The leaf nodes therefore form the first level of the index with the internal nodes forming the other levels of a multilevel index
- Some of the key values of the leaf nodes may also appear in internal nodes

SQL Data Manipulation Language (DML)

Physical Data Organization

B+ Trees





SQL Data Manipulation Language (DML)

Physical Data Organization

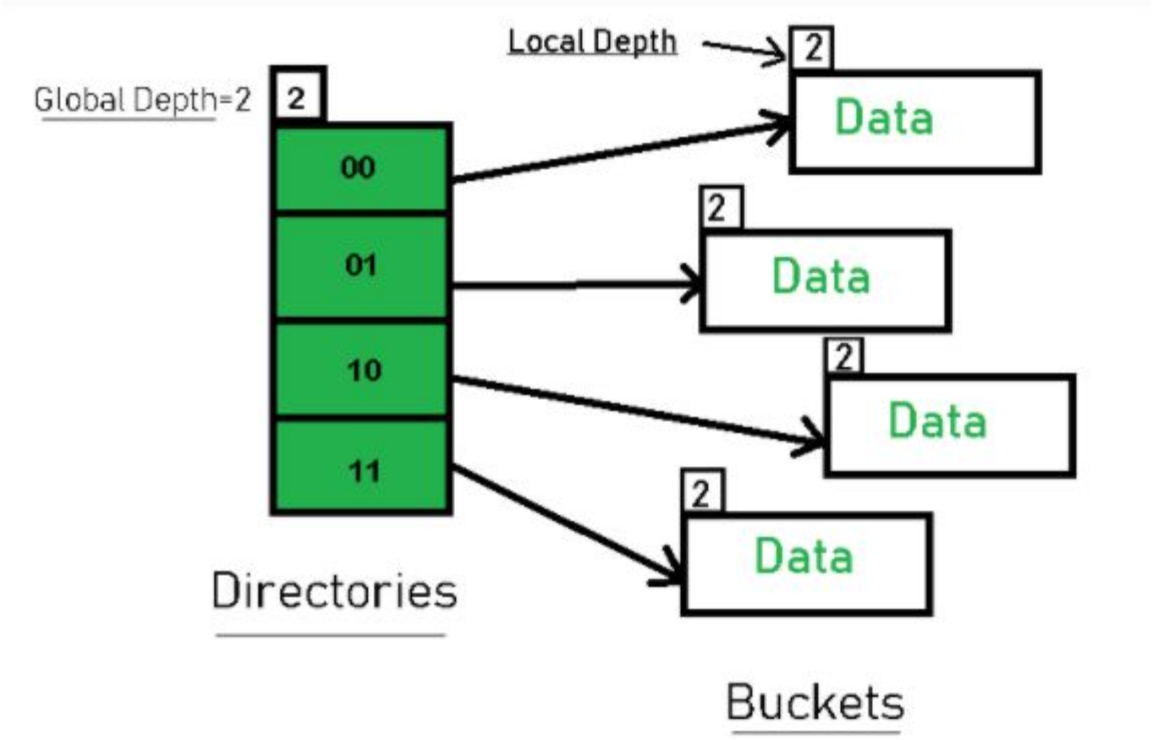
Extendible Hashing

- Extendible Hashing is a dynamic hashing method wherein directories, and buckets are used to hash data
- It is an aggressively flexible method in which the hash function also experiences dynamic changes
- The main features in this hashing technique are Directories and Buckets
- Directories: The directories store addresses of the buckets in pointers
- An id is assigned to each directory which may change each time when Directory Expansion takes place
- Buckets: The buckets are used to hash the actual data

SQL Data Manipulation Language (DML)

Physical Data Organization

Extendible Hashing



SQL Data Manipulation Language (DML)

Physical Data Organization

Extendible Hashing

- In extendible hashing, an array of 2^d bucket addresses is maintained where d is called the global depth of the directory
- Global depth is the number of bits in the directory id
- A local depth d’ stored with each bucket specifies the number of bits on which the bucket contents are based

SQL Data Manipulation Language (DML)

Physical Data Organization

Extendible Hashing

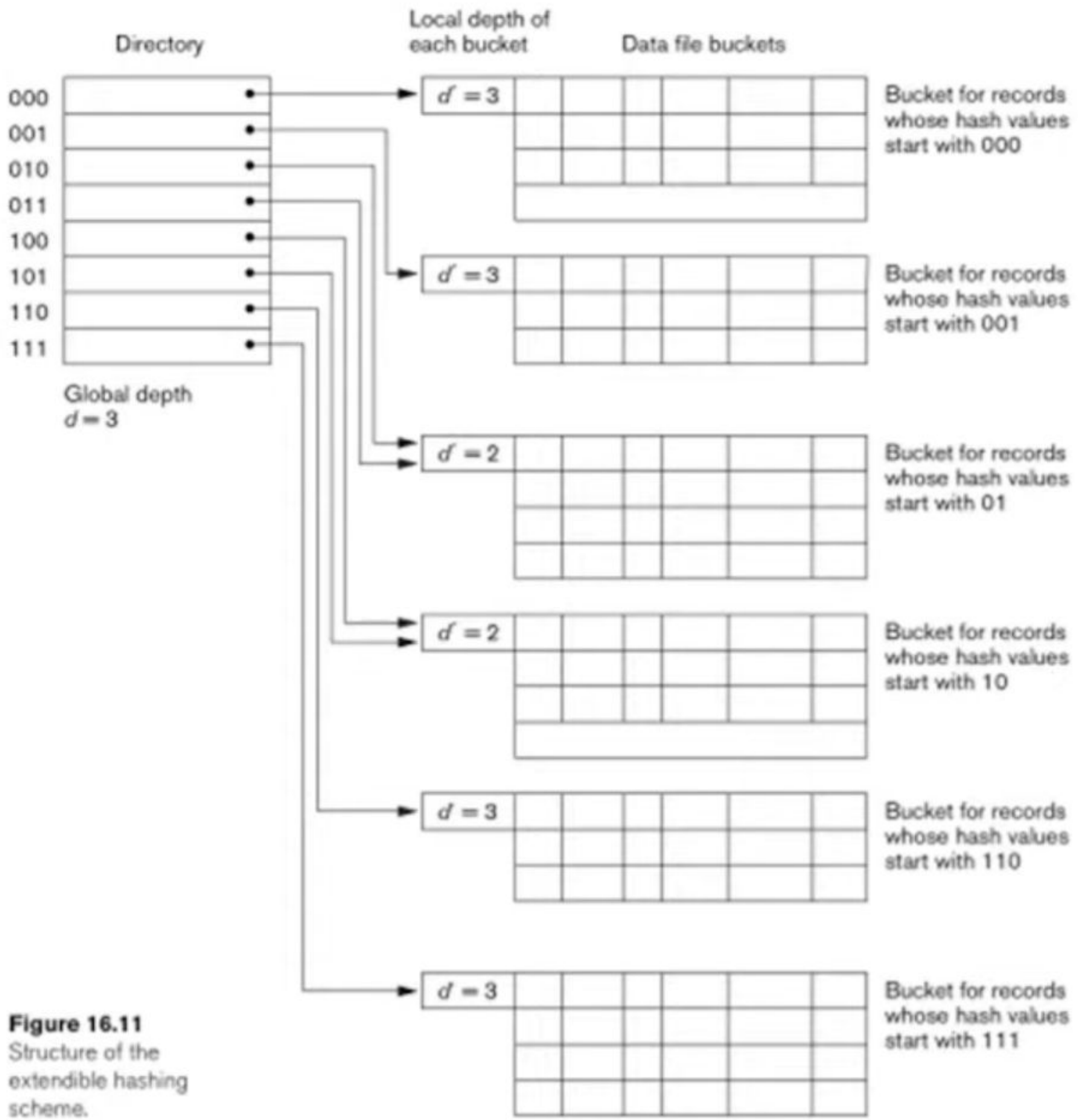


Figure 16.11
Structure of the extendible hashing scheme.

SQL Data Manipulation Language (DML)

Physical Data Organization

Indexing on Multiple Keys

- Here we use combination of two or more coloumns which are frequently queried to get index
- For example, SELECT * FROM EMPLOYEE WHERE DEPT_ID = 20 AND SALARY = 5000;
- Here we need both DEPT_ID as well as SALARY as index
- DEPT_ID and SALARY are clubbed into one index and are stored in the files
- Then it filers both at a short and returns the result