BCA501N

# Unit-III

<u>File Organization</u>

File Organization

- o The **File** is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.

- o File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.

- o File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.

- o The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.

- o Files of fixed length records are easier to implement than the files of variable length records.

Objective of file organization

- o It contains an optimal selection of records, i.e., records can be selected as fast as possible.

- o To perform insert, delete or update transaction on the records should be quick and easy.

- o The duplicate records cannot be induced as a result of insert, update or delete.

- o For the minimal cost of storage, records should be stored efficiently.

Types of file organization:

File organization contains various methods. These particular methods have pros and cons on the basis of access or selection. In the file organization, the programmer decides the best-suited file organization method according to his requirement. Types of file organization are as follows:
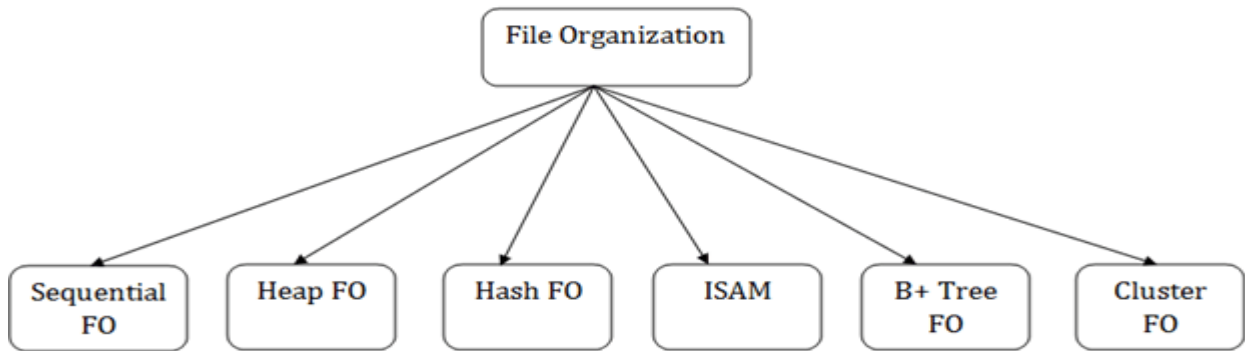
<u>Sequential File Organization</u>

This method is the easiest method for file organization. In this method, files are stored sequentially. This method can be implemented in two ways:
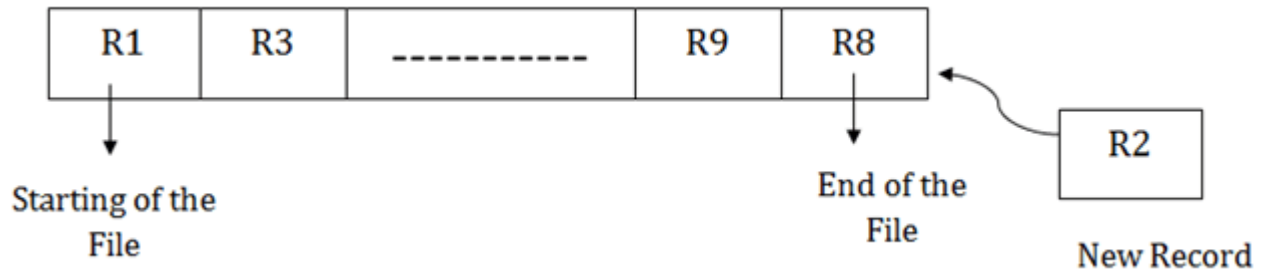
1. Pile File Method:

- o It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables.

- o In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.
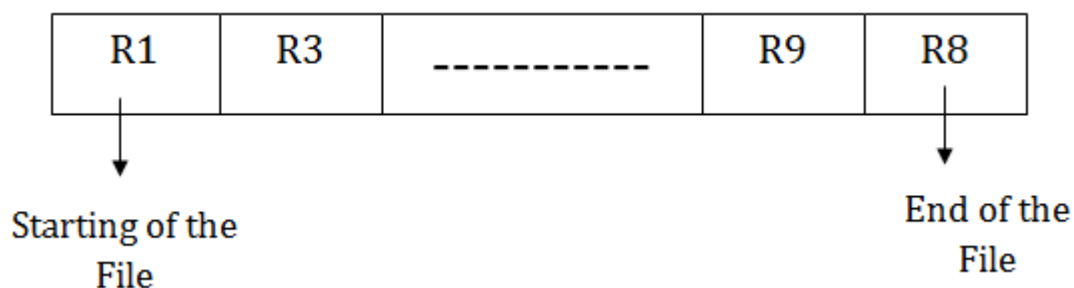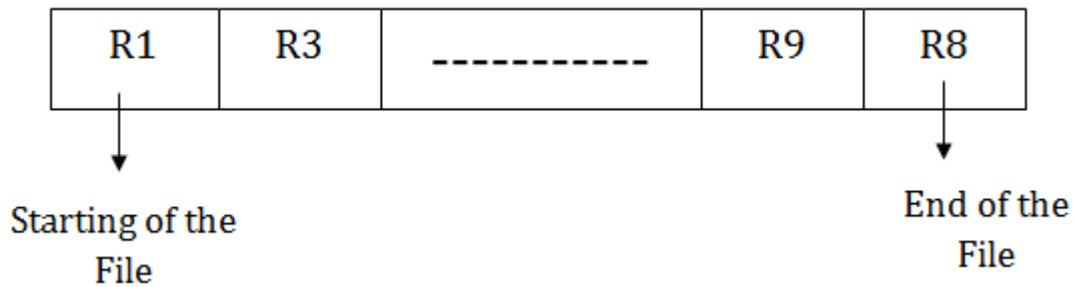

Insertion of the new record:

Suppose we have four records R1, R3 and so on upto R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2 in the sequence, then it will be placed at the end of the file. Here, records are nothing but a row in any table.

| R1 | R3 | ----------- | R9 | R8 |

Starting of the File

End of the File

R2

New Record

2. Sorted File Method:

o   In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order. Sorting of records is based on any primary key or any other key.

o   In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.

| R1 | R3 | ----------- | R9 | R8 |

Starting of the File

End of the File

Insertion of the new record:

Suppose there is a preexisting sorted sequence of four records R1, R3 and so on upto R6 and R7. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence.
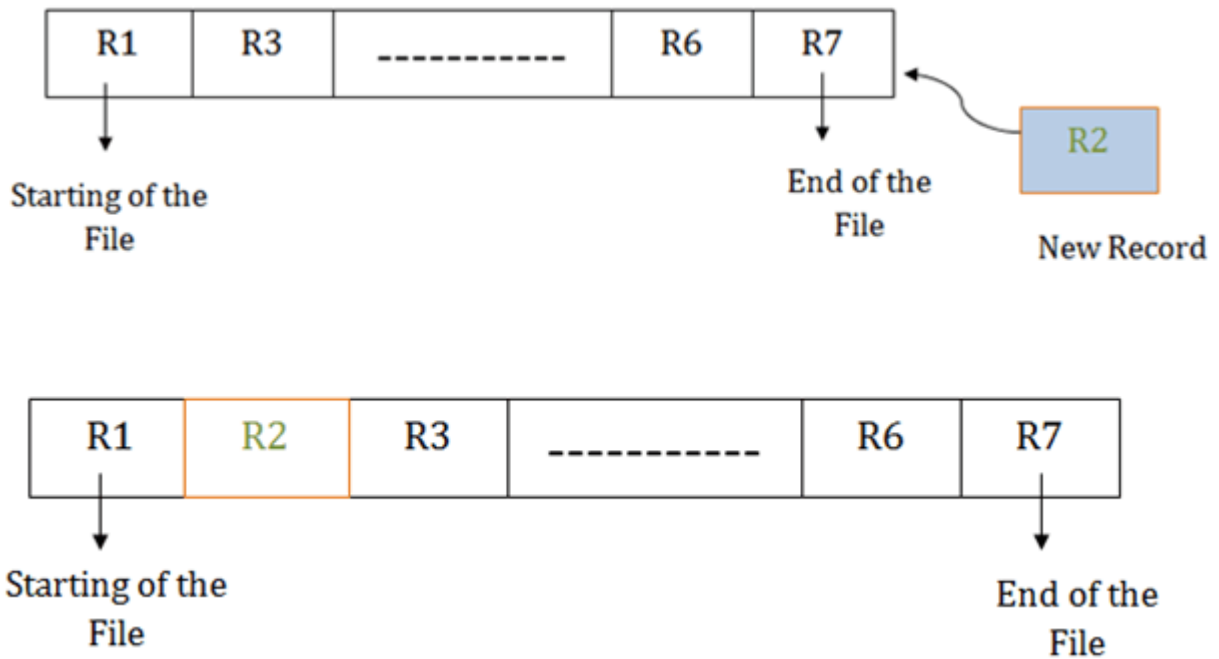
Pros of sequential file organization

   o   It contains a fast and efficient method for the huge amount of data.

   o   In this method, files can be easily stored in cheaper storage mechanism like magnetic tapes.

   o   It is simple in design. It requires no much effort to store the data.

   o   This method is used when most of the records have to be accessed like grade calculation of a student, generating the salary slip, etc.

   o   This method is used for report generation or statistical calculations.

Cons of sequential file organization

   o   It will waste time as we cannot jump on a particular record that is required but we have to move sequentially which takes our time.

   o   Sorted file method takes more time and space for sorting the records.

Indexed sequential access method (ISAM)

ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key. An index value is generated for each primary key and mapped with the record. This index contains the address of the record in the file.

Data Records			Data Blocks in memory
R1	AA6DK		DS46G
R2	BS8KA		XS5GF
R5	SA7VD		BS8KA
R7	DS46G		DH4FD
R8	XS5GF		AA6DK
⋮	⋮		⋮
R9	DH4FD		SA7VD

If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

Pros of ISAM:

- o In this method, each record has the address of its data block, searching a record in a huge database is quick and easy.

- o This method supports range retrieval and partial retrieval of records. Since the index is based on the primary key values, we can retrieve the data for the given range of value. In the same way, the partial value can also be easily searched, i.e., the student name starting with 'JA' can be easily searched.

Cons of ISAM

- o This method requires extra space in the disk to store the index value.

- o When the new records are inserted, then these files have to be reconstructed to maintain the sequence.

- o When the record is deleted, then the space used by it needs to be released. Otherwise, the performance of the database will slow down.

Introduction of B-Tree

B Tree

B Tree is a specialized m-way tree that can be widely used for disk access. A B-Tree of order m can have at most m-1 keys and m children. One of the main reasons of using B tree is its capability to store large number of keys in a single node and large key values by keeping the height of the tree relatively small.

A B tree of order m contains all the properties of an M way tree. In addition, it contains the following properties.

1. Every node in a B-Tree contains at most m children.

2. Every node in a B-Tree except the root node and the leaf node contain at least m/2 children.

3. The root nodes must have at least 2 nodes.

4. All leaf nodes must be at the same level.

It is not necessary that, all the nodes contain the same number of children but, each node must have m/2 number of nodes.

A B tree of order 4 is shown in the following image.
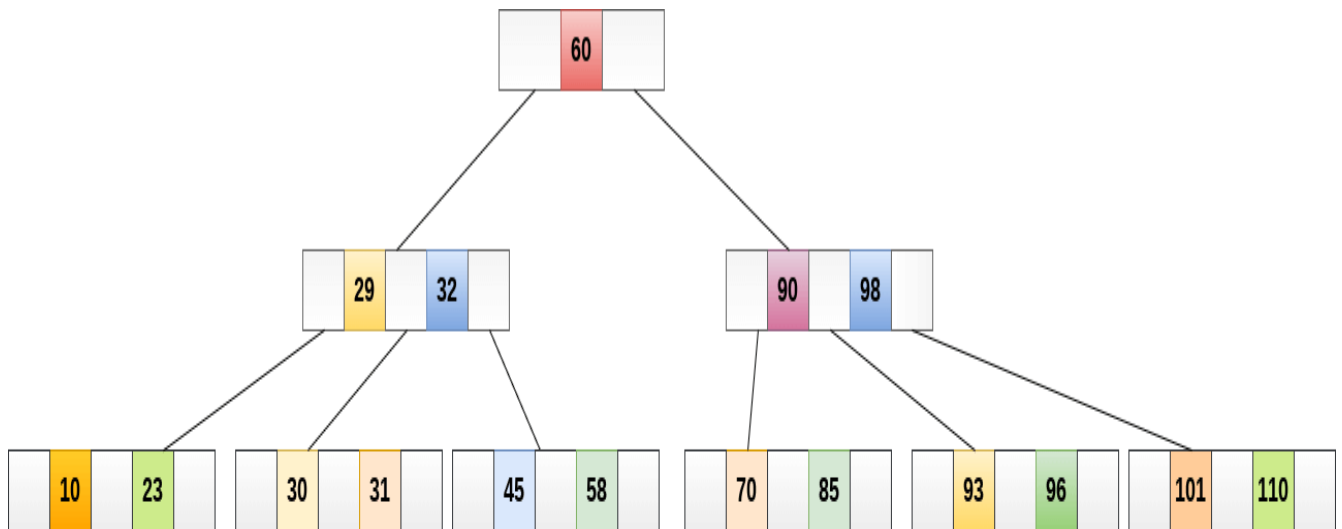
While performing some operations on B Tree, any property of B Tree may violate such as number of minimum children a node can have. To maintain the properties of B Tree, the tree may split or join.

Operations

Searching :

Searching in B Trees is similar to that in Binary search tree. For example, if we search for an item 49 in the following B Tree. The process will something like following :

1. Compare item 49 with root node 78. since 49 < 78 hence, move to its left sub-tree.

2. Since, 40<49<56, traverse right sub-tree of 40.

3. 49>45, move to right. Compare 49.

4. match found, return.

Searching in a B tree depends upon the height of the tree. The search algorithm takes O(log n) time to search any element in a B tree.
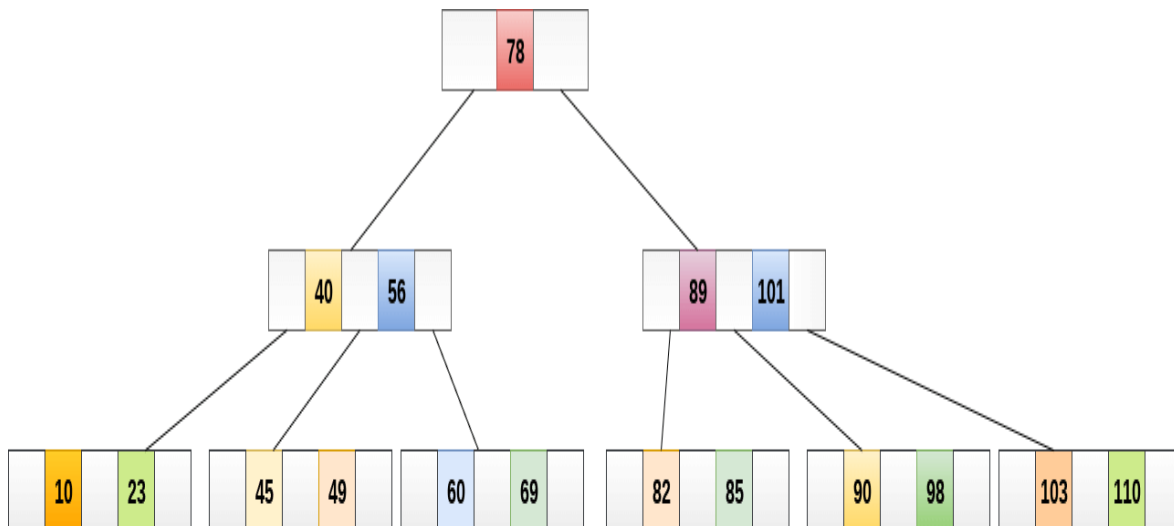
Inserting

Insertions are done at the leaf node level. The following algorithm needs to be followed in order to insert an item into B Tree.

1. Traverse the B Tree in order to find the appropriate leaf node at which the node can be inserted.

2. If the leaf node contain less than m-1 keys then insert the element in the increasing order.

3. Else, if the leaf node contains m-1 keys, then follow the following steps.

   o   Insert the new element in the increasing order of elements.

   o   Split the node into the two nodes at the median.

   o   Push the median element upto its parent node.

   o   If the parent node also contain m-1 number of keys, then split it too by following the same steps.
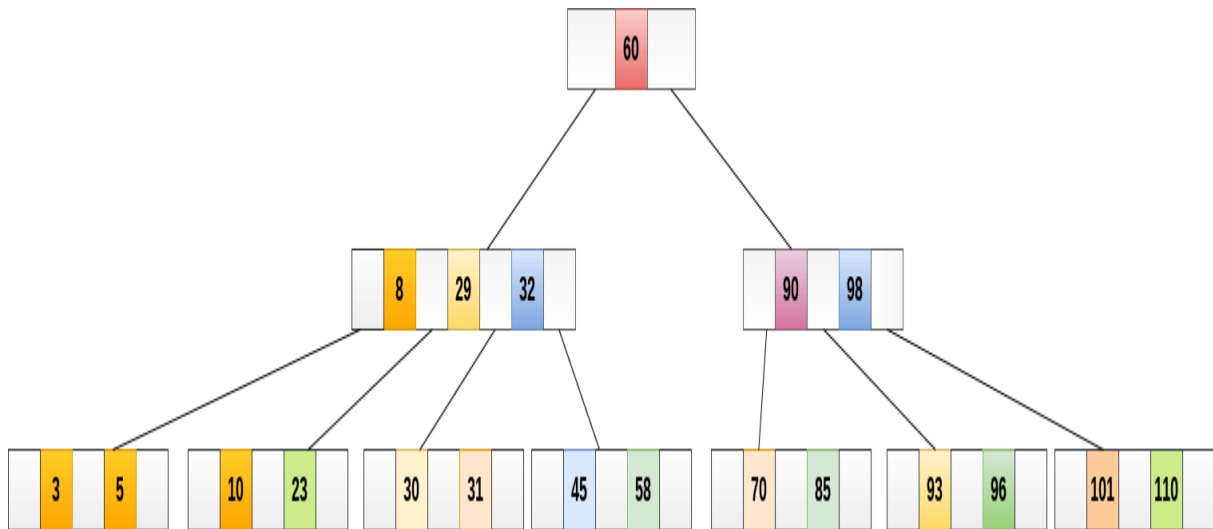
Example:

Insert the node 8 into the B Tree of order 5 shown in the following image.
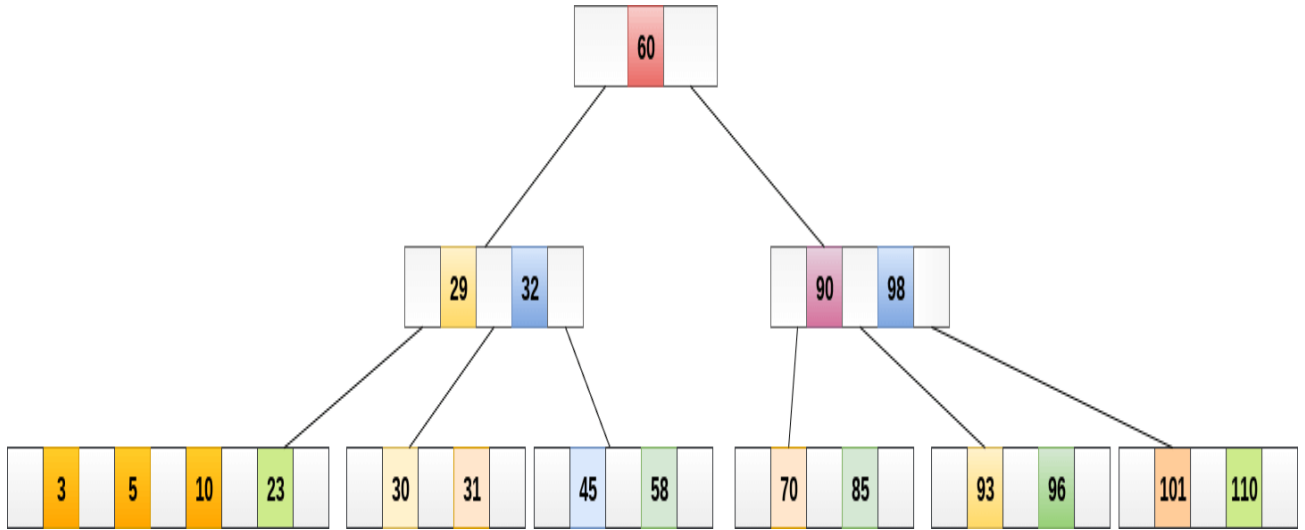
8 will be inserted to the right of 5, therefore insert 8.
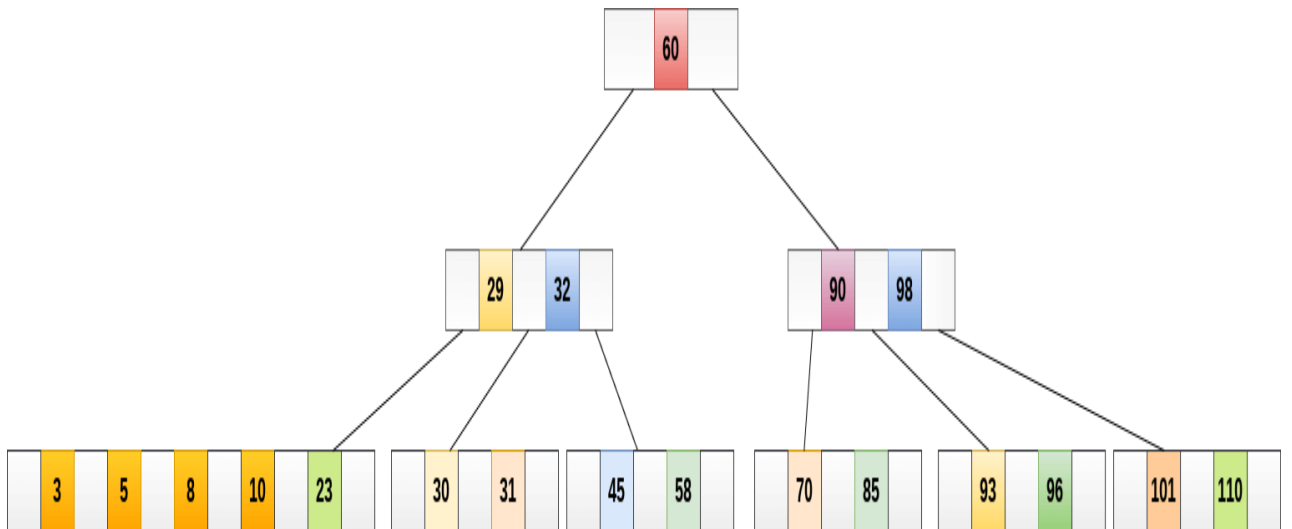
The node, now contain 5 keys which is greater than (5 -1 = 4) keys. Therefore split the node from the median i.e. 8 and push it up to its parent node shown as follows.

Deletion

Deletion is also performed at the leaf nodes. The node which is to be deleted can either be a leaf node or an internal node. Following algorithm needs to be followed in order to delete a node from a B tree.

1. Locate the leaf node.

2. If there are more than m/2 keys in the leaf node then delete the desired key from the node.

3. If the leaf node doesn't contain m/2 keys then complete the keys by taking the element from right or left sibling.

   o If the left sibling contains more than m/2 elements then push its largest element up to its parent and move the intervening element down to the node where the key is deleted.

   o If the right sibling contains more than m/2 elements then push its smallest element up to the parent and move intervening element down to the node where the key is deleted.

4. If neither of the sibling contain more than m/2 elements then create a new leaf node by joining two leaf nodes and the intervening element of the parent node.

5. If parent is left with less than m/2 nodes then, apply the above process on the parent too.

If the node which is to be deleted is an internal node, then replace the node with its in-order successor or predecessor. Since, successor or predecessor will always be on the leaf node hence, the process will be similar as the node is being deleted from the leaf node.

Example 1

Delete the node 53 from the B Tree of order 5 shown in the following figure.
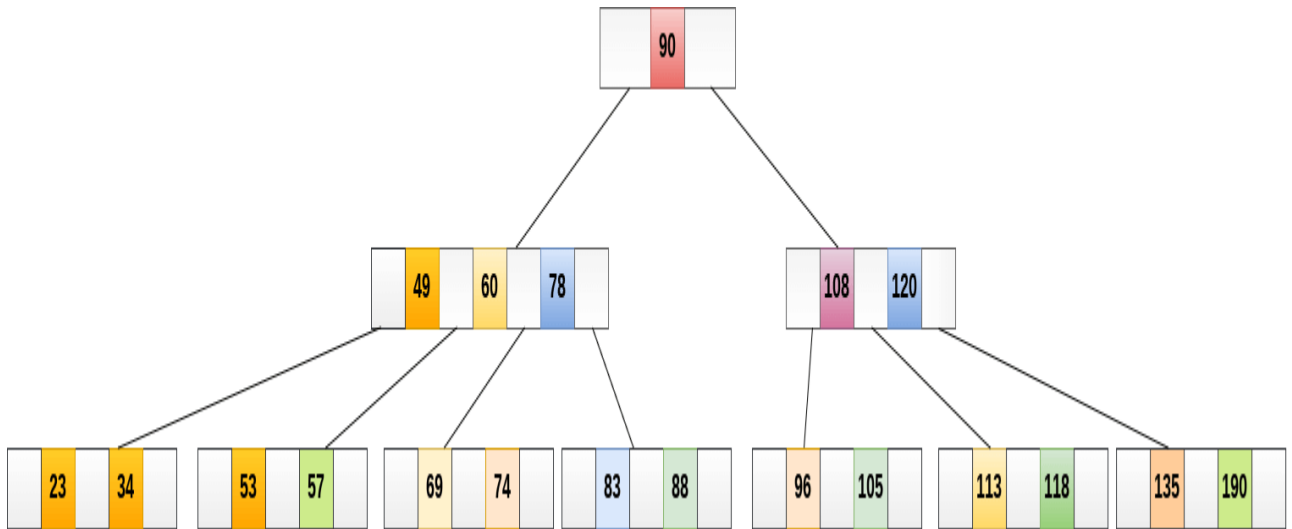
53 is present in the right child of element 49. Delete it.
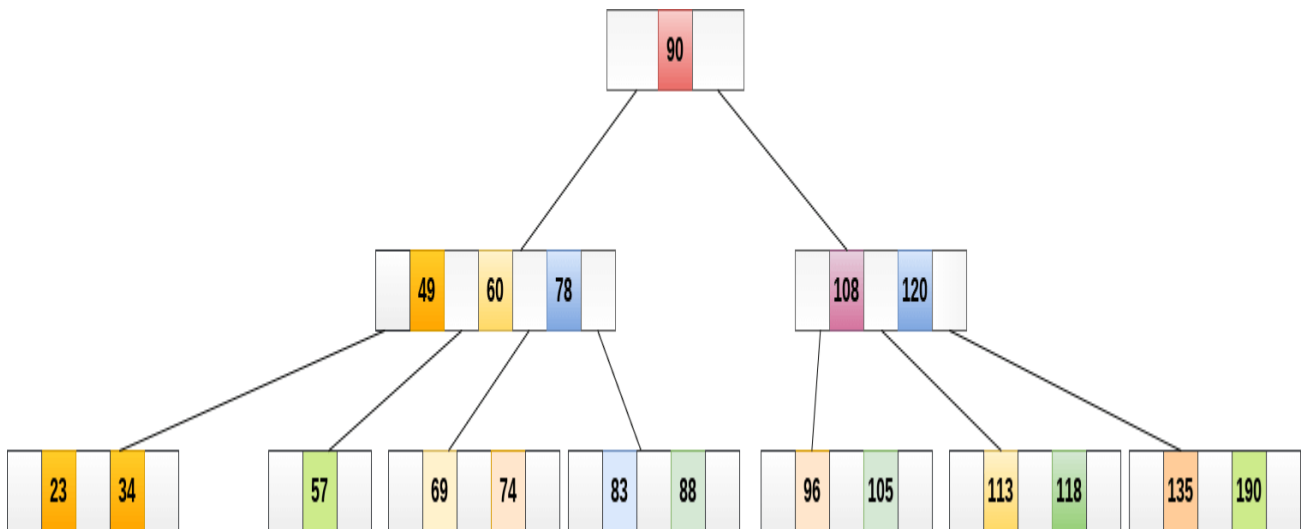
Now, 57 is the only element which is left in the node, the minimum number of elements that must be present in a B tree of order 5, is 2. it is less than that, the elements in its left and right sub-tree are also not sufficient therefore, merge it with the left sibling and intervening element of parent i.e. 49.

The final B tree is shown as follows.
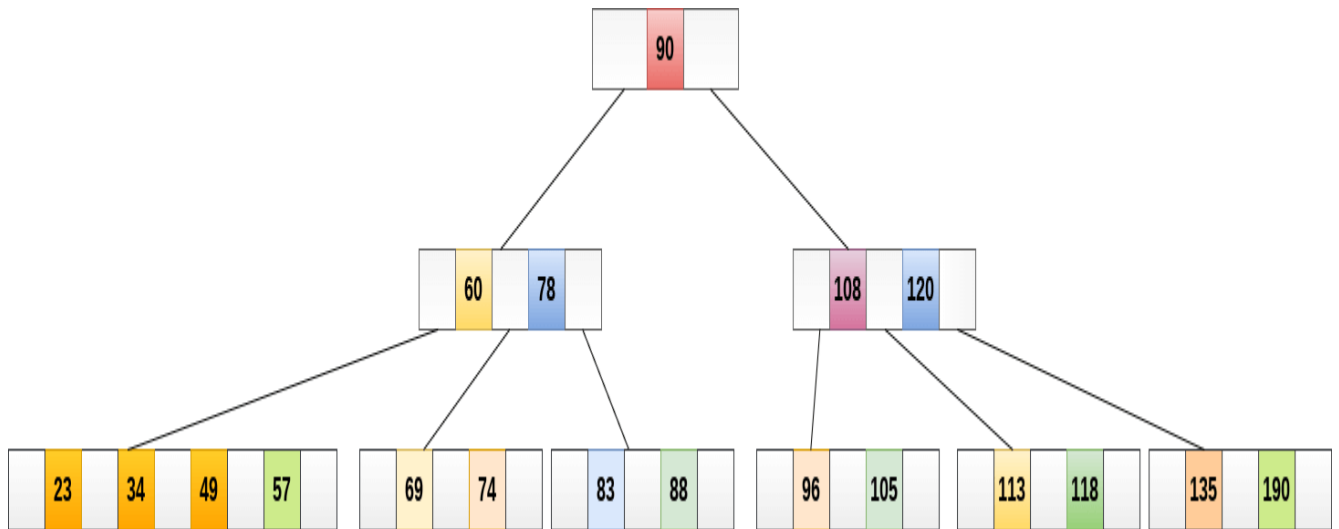
Application of B tree

B tree is used to index the data and provides fast access to the actual data stored on the disks since, the access to value stored in a large database that is stored on a disk is a very time consuming process.

Searching an un-indexed and unsorted database containing n key values needs O(n) running time in worst case. However, if we use B Tree to index this database, it will be searched in O(log n) time in worst case.

Introduction of B+ Tree

B+ Tree

o   The B+ tree is a balanced binary search tree. It follows a multi-level index format.

o   In the B+ tree, leaf nodes denote actual data pointers. B+ tree ensures that all leaf nodes remain at the same height.

o   In the B+ tree, the leaf nodes are linked using a link list. Therefore, a B+ tree can support random access as well as sequential access.

Structure of B+ Tree

o   In the B+ tree, every leaf node is at equal distance from the root node. The B+ tree is of the order n where n is fixed for every B+ tree.

o   It contains an internal node and leaf node.
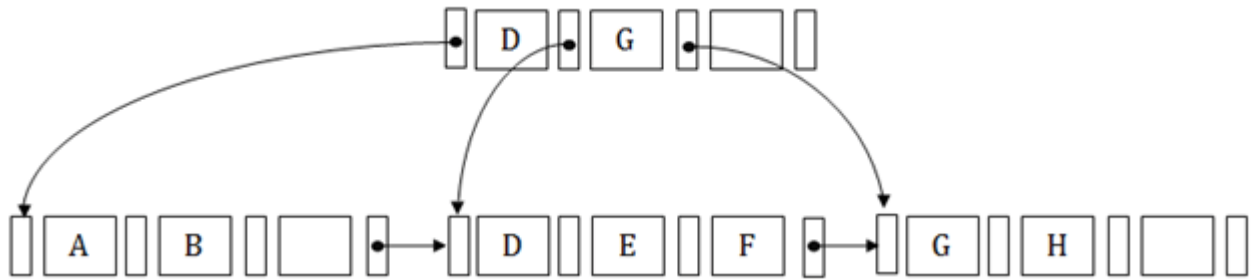
Internal node

- o An internal node of the B+ tree can contain at least n/2 record pointers except the root node.

- o At most, an internal node of the tree contains n pointers.

Leaf node

- o The leaf node of the B+ tree can contain at least n/2 record pointers and n/2 key values.

- o At most, a leaf node contains n record pointer and n key values.

- o Every leaf node of the B+ tree contains one block pointer P to point to next leaf node.

Searching a record in B+ Tree

Suppose we have to search 55 in the below B+ tree structure. First, we will fetch for the intermediary node which will direct to the leaf node that can contain a record for 55.

So, in the intermediary node, we will find a branch between 50 and 75 nodes. Then at the end, we will be redirected to the third leaf node. Here DBMS will perform a sequential search to find 55.
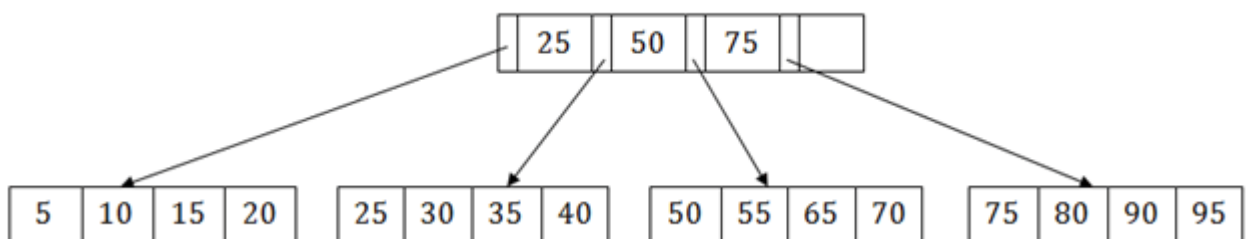
B+ Tree Insertion

Suppose we want to insert a record 60 in the below structure. It will go to the 3rd leaf node after 55. It is a balanced tree, and a leaf node of this tree is already full, so we cannot insert 60 there.

In this case, we have to split the leaf node, so that it can be inserted into tree without affecting the fill factor, balance and order.
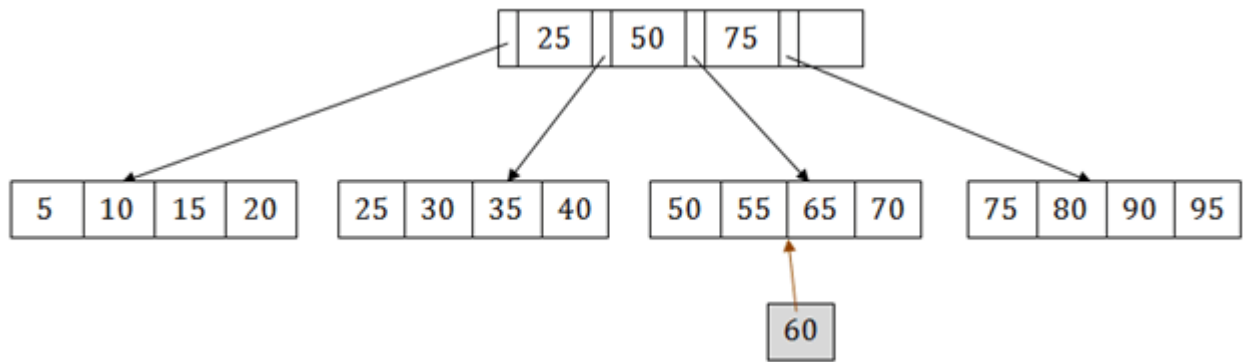
The 3rd leaf node has the values (50, 55, 60, 65, 70) and its current root node is 50. We will split the leaf node of the tree in the middle so that its balance is not altered. So we can group (50, 55) and (60, 65, 70) into 2 leaf nodes.

If these two has to be leaf nodes, the intermediate node cannot branch from 50. It should have 60 added to it, and then we can have pointers to a new leaf node.
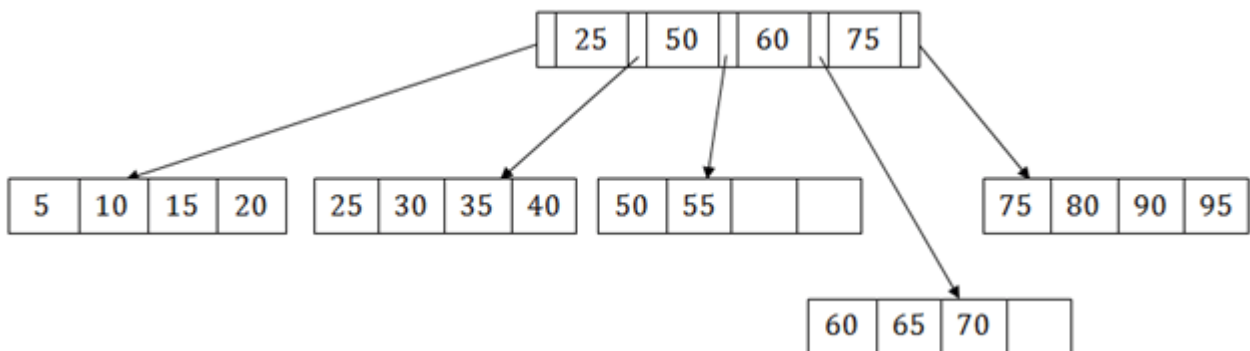
This is how we can insert an entry when there is overflow. In a normal scenario, it is very easy to find the node where it fits and then place it in that leaf node.

B+ Tree Deletion

Suppose we want to delete 60 from the above example. In this case, we have to remove 60 from the intermediate node as well as from the 4th leaf node too. If we remove it from the intermediate node, then the tree will not satisfy the rule of the B+ tree. So we need to modify it to have a balanced tree.

After deleting node 60 from above B+ tree and re-arranging the nodes, it will show as follows:
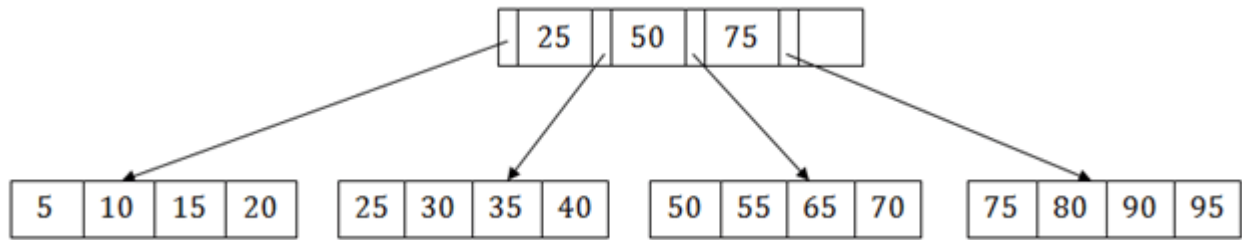
<u>Hashing</u>

In a huge database structure, it is very inefficient to search all the index values and reach the desired data. Hashing technique is used to calculate the direct location of a data record on the disk without using index structure.

In this technique, data is stored at the data blocks whose address is generated by using the hashing function. The memory location where these records are stored is known as data bucket or data blocks.

In this, a hash function can choose any of the column value to generate the address. Most of the time, the hash function uses the primary key to generate the address of the data block. A hash function is a simple mathematical function to any complex mathematical function. We can even consider the primary key itself as the address of the data block. That means each row whose address will be the same as a primary key stored in the data block.

Data Buckets in Memory

Data Records		Data Buckets Address	Actual data in Memory		
98	→	98	100	John	Delhi
103					
106		102			
104		103	103	John	US
		104	104	Kathrin	UP
		105	105	Honey	China
102		106	106	Jackson	Delhi
		120			
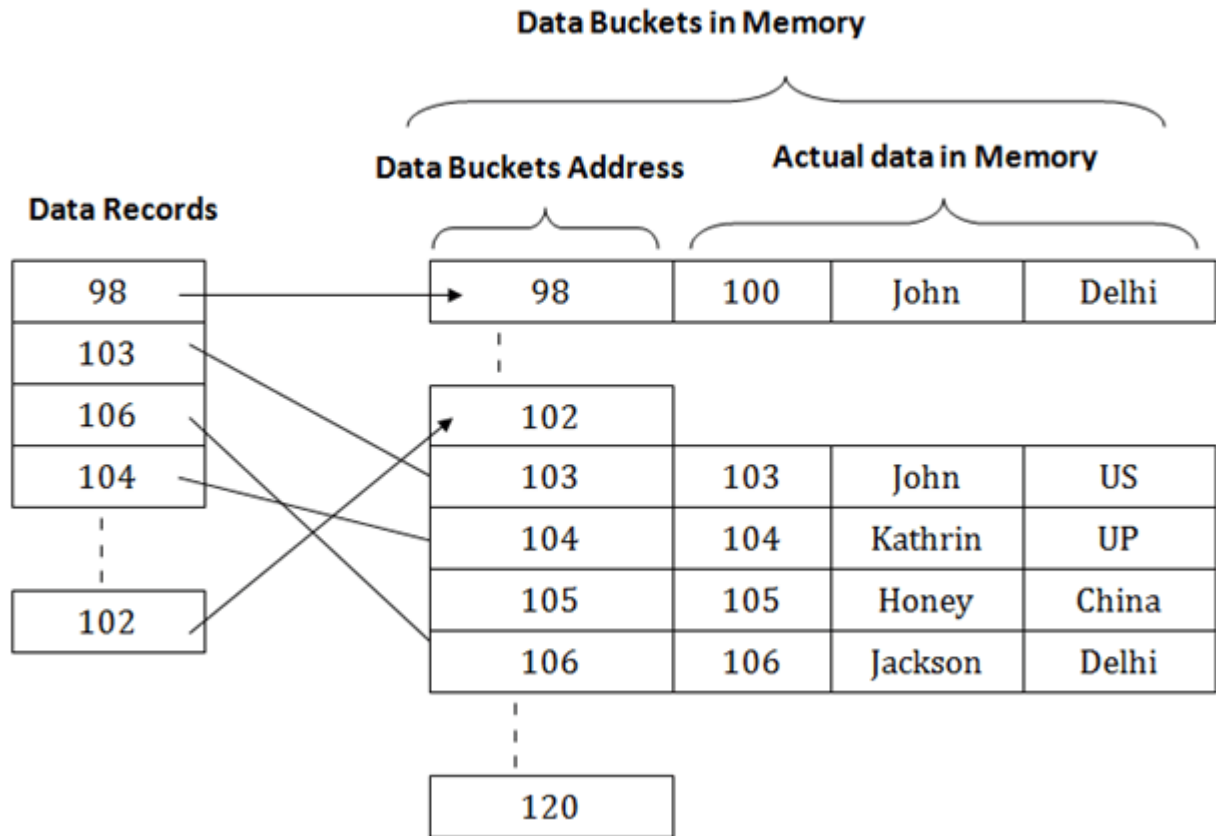
The above diagram shows data block addresses same as primary key value. This hash function can also be a simple mathematical function like exponential, mod, cos, sin, etc. Suppose we have mod (5) hash function to determine the address of the data block. In this case, it applies mod (5) hash function on the primary keys and generates 3, 3, 1, 4 and 2 respectively, and records are stored in those data block addresses.

Data Buckets in Memory

Data Records		Data Buckets Address	Actual Data in Memory					
98		1	106	James	Delhi			
104		2	102	Harry	US			
106		3	98	Alia	UK	—	103	Amy	UK
		4	104	Jackson	China			
102		5						
		6						
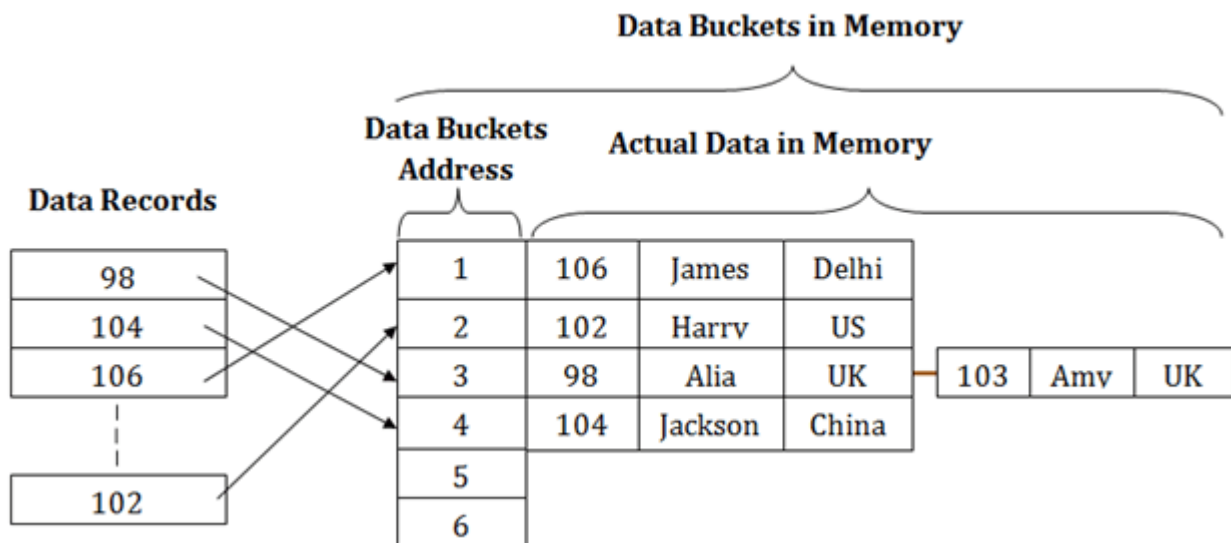
Types of Hashing:
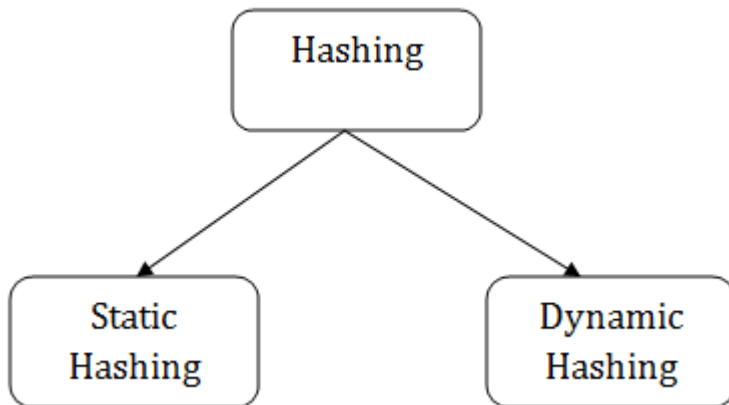
Static Hashing

In static hashing, the resultant data bucket address will always be the same. That means if we generate an address for EMP_ID =103 using the hash function mod (5) then it will always result in same bucket address 3. Here, there will be no change in the bucket address.

Hence in this static hashing, the number of data buckets in memory remains constant throughout. In this example, we will have five data buckets in the memory used to store the data.
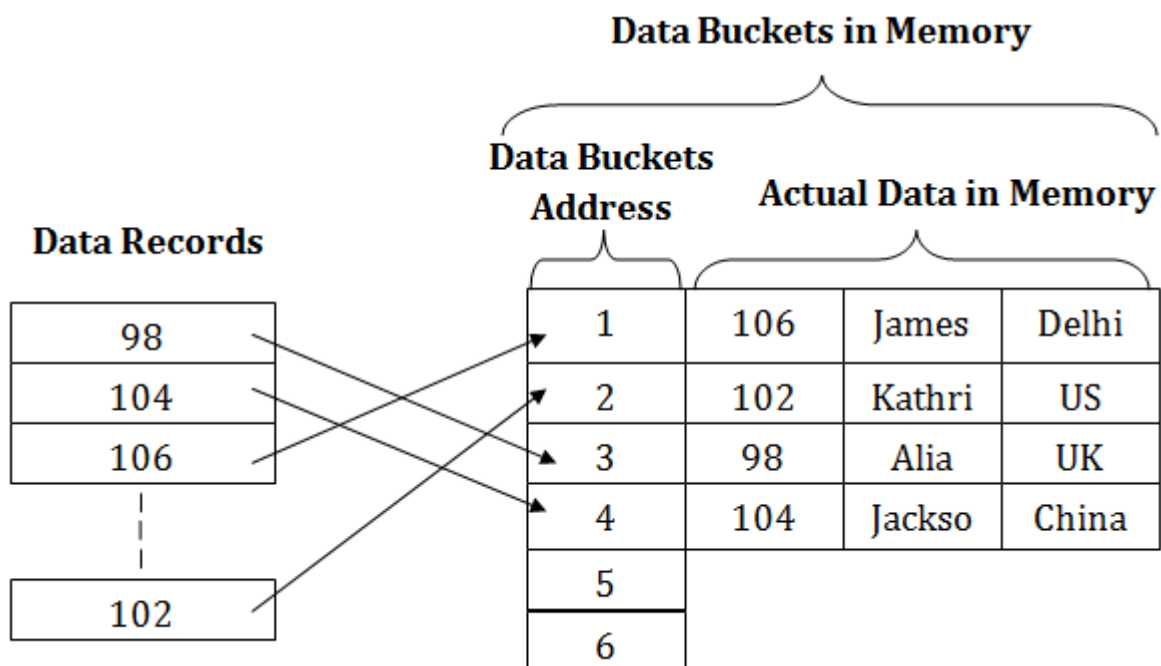
Operations of Static Hashing

○ **Searching a record**

When a record needs to be searched, then the same hash function retrieves the address of the bucket where the data is stored.

- **Insert a Record**

When a new record is inserted into the table, then we will generate an address for a new record based on the hash key and record is stored in that location.

- **Delete a Record**

To delete a record, we will first fetch the record which is supposed to be deleted. Then we will delete the records for that address in memory.

- **Update a Record**

To update a record, we will first search it using a hash function, and then the data record is updated.

If we want to insert some new record into the file but the address of a data bucket generated by the hash function is not empty, or data already exists in that address. This situation in the static hashing is known as **bucket overflow**. This is a critical situation in this method.

To overcome this situation, there are various methods. Some commonly used methods are as follows:

1. Open Hashing

When a hash function generates an address at which data is already stored, then the next bucket will be allocated to it. This mechanism is called as **Linear Probing**.

For example: suppose R3 is a new address which needs to be inserted, the hash function generates address as 112 for R3. But the generated address is already full. So the system searches next available data bucket, 113 and assigns R3 to it.
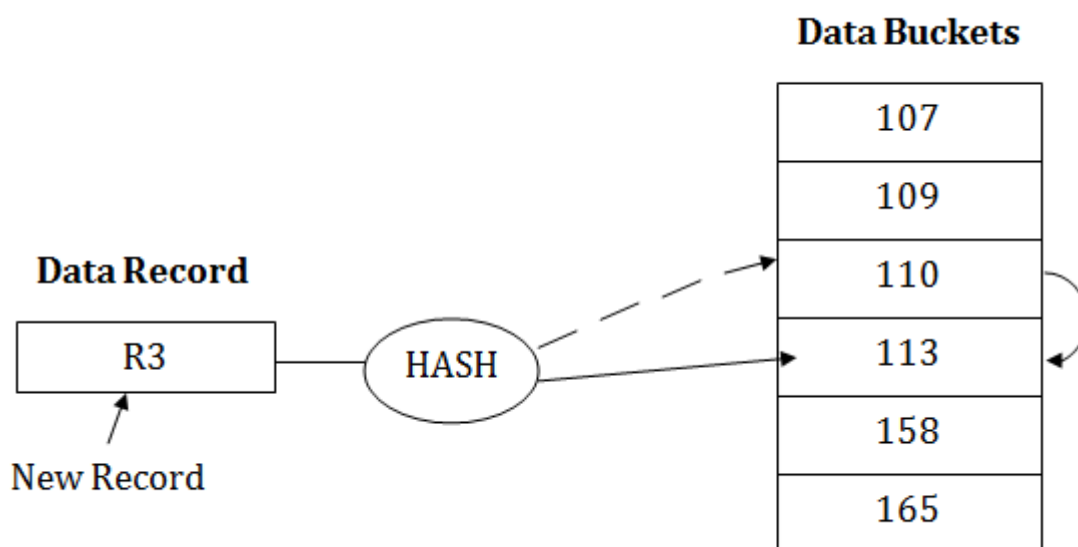
2. Close Hashing

When buckets are full, then a new data bucket is allocated for the same hash result and is linked after the previous one. This mechanism is known as **Overflow chaining**.

For example: Suppose R3 is a new address which needs to be inserted into the table, the hash function generates address as 110 for it. But this bucket is full to store the new data. In this case, a new bucket is inserted at the end of 110 buckets and is linked to it.
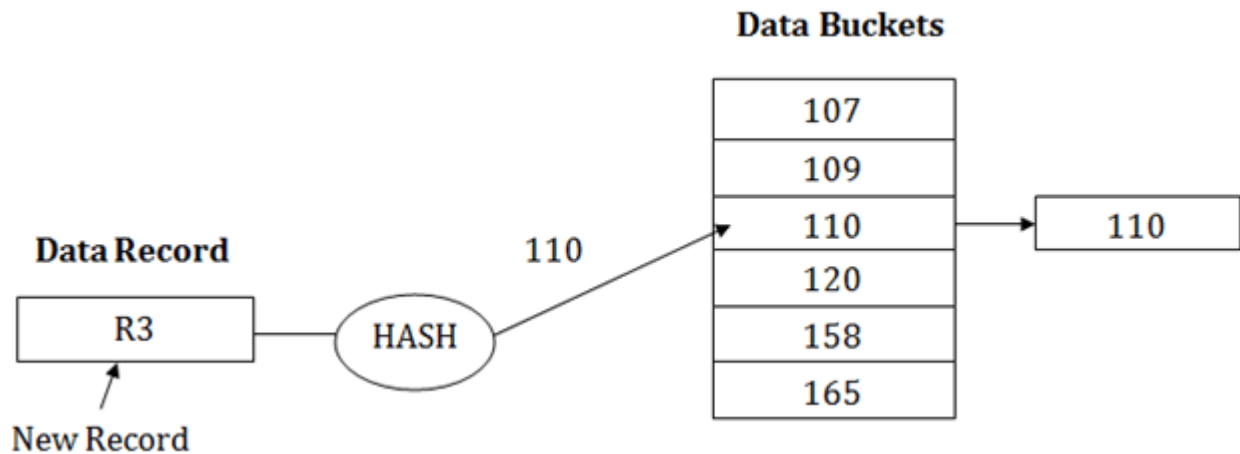
Dynamic Hashing

- o The dynamic hashing method is used to overcome the problems of static hashing like bucket overflow.

- o In this method, data buckets grow or shrink as the records increases or decreases. This method is also known as Extendable hashing method.

- o This method makes hashing dynamic, i.e., it allows insertion or deletion without resulting in poor performance.

How to search a key

- o First, calculate the hash address of the key.

- o Check how many bits are used in the directory, and these bits are called as i.

- o Take the least significant i bits of the hash address. This gives an index of the directory.

- o Now using the index, go to the directory and find bucket address where the record might be.

How to insert a new record

- o Firstly, you have to follow the same procedure for retrieval, ending up in some bucket.

- o   If there is still space in that bucket, then place the record in it.

- o   If the bucket is full, then we will split the bucket and redistribute the records.

For example:

Consider the following grouping of keys into buckets, depending on the prefix of their hash address:

Key	Hash address
1	11010
2	00000
3	11110
4	00000
5	01001
6	10101
7	10111

The last two bits of 2 and 4 are 00. So it will go into bucket B0. The last two bits of 5 and 6 are 01, so it will go into bucket B1. The last two bits of 1 and 3 are 10, so it will go into bucket B2. The last two bits of 7 are 11, so it will go into B3.

Data Buckets

Data Records
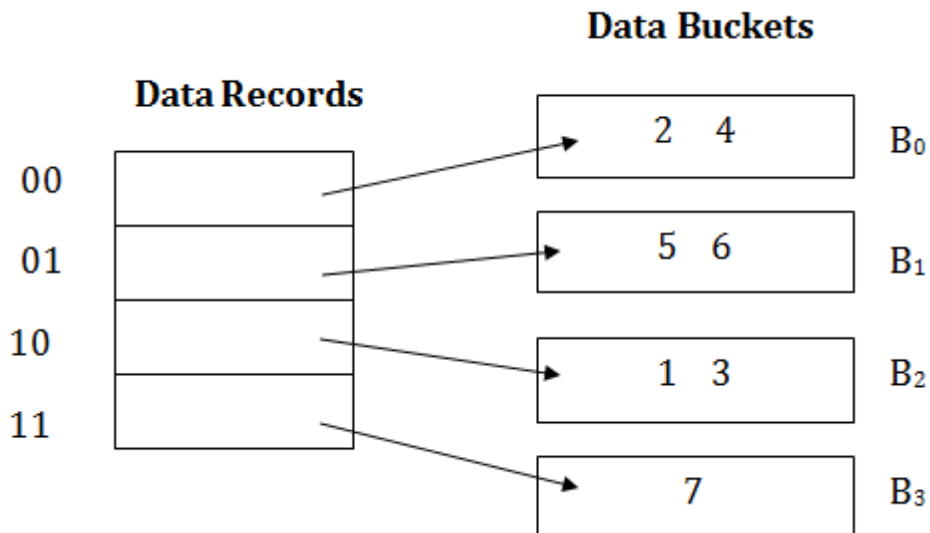
Insert key 9 with hash address 10001 into the above structure:

- o   Since key 9 has hash address 10001, it must go into the first bucket. But bucket B1 is full, so it will get split.

- o   The splitting will separate 5, 9 from 6 since last three bits of 5, 9 are 001, so it will go into bucket B1, and the last three bits of 6 are 101, so it will go into bucket B5.

- Keys 2 and 4 are still in B0. The record in B0 pointed by the 000 and 100 entry because last two bits of both the entry are 00.

- Keys 1 and 3 are still in B2. The record in B2 pointed by the 010 and 110 entry because last two bits of both the entry are 10.

- Key 7 are still in B3. The record in B3 pointed by the 111 and 011 entry because last two bits of both the entry are 11.
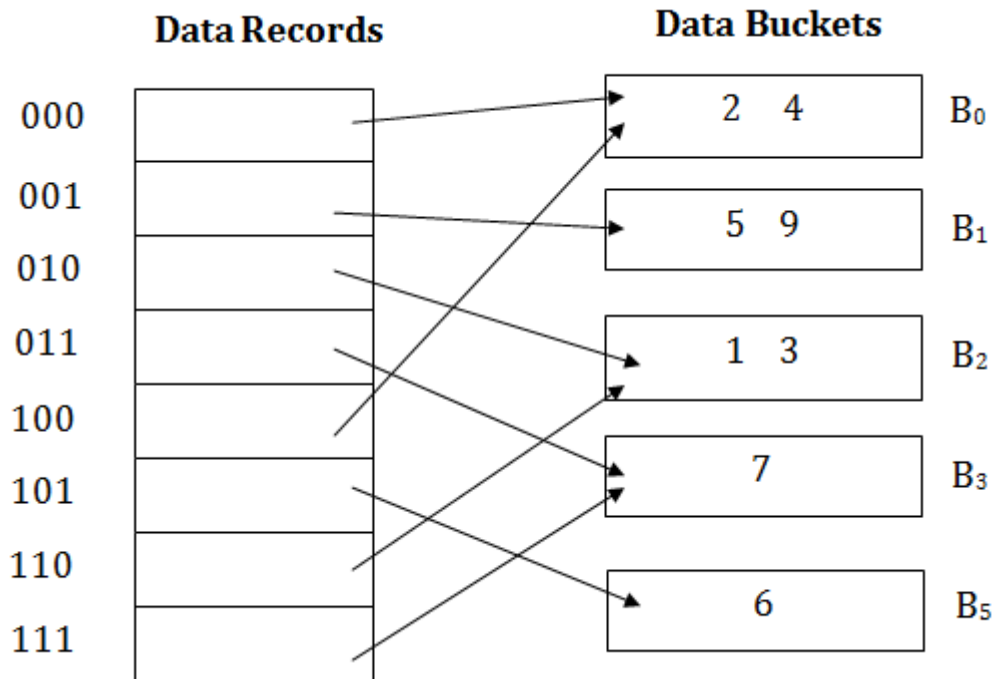
Advantages of dynamic hashing

- In this method, the performance does not decrease as the data grows in the system. It simply increases the size of memory to accommodate the data.

- In this method, memory is well utilized as it grows and shrinks with the data. There will not be any unused memory lying.

- This method is good for the dynamic database where data grows and shrinks frequently.

Disadvantages of dynamic hashing

- In this method, if the data size increases then the bucket size is also increased. These addresses of data will be maintained in the bucket address table. This is because the data address will keep changing as buckets grow and shrink. If there is a huge increase in data, maintaining the bucket address table becomes tedious.

- In this case, the bucket overflow situation will also occur. But it might take little time to reach this situation than static hashing.

Unit–IV

Relational Data Model

Relational Model concept

Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.

Domain: It contains a set of atomic values that an attribute can take.

Attribute: It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai)

Relational instance: In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.

Relational schema: A relational schema contains the name of the relation and name of all columns or attributes.

Relational key: In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

Example: STUDENT Relation

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

- o In the given table, NAME, ROLL_NO, PHONE_NO, ADDRESS, and AGE are the attributes.

- o The instance of schema STUDENT has 5 tuples.

- o t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

Properties of Relations

- o Name of the relation is distinct from all other relations.

- o Each relation cell contains exactly one atomic (single) value

- o Each attribute contains a distinct name

- o Attribute domain has no significance

- o tuple has no duplicate value

- o Order of tuple can have a different sequence

Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

Types of Relational operation
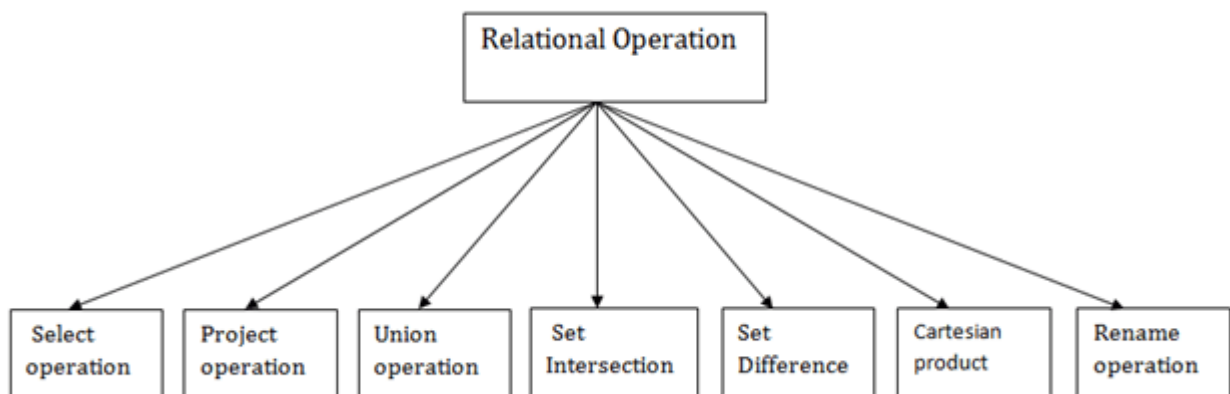
1. Select Operation:

- o The select operation selects tuples that satisfy a given predicate.

- o It is denoted by sigma (σ).

1. **Notation**: $\sigma p(r)$

Where:

σ is used for selection prediction
r is used for relation

p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like $=, \neq, \geq, <, >, \leq$.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.

- It is denoted by ∏.

  Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.

- It eliminates the duplicate tuples. It is denoted by ∪.

Notation: R ∪ S

A union operation must hold the following condition:

- R and S must have the attribute of the same number.

- Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15

Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:

∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

o Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.

o It is denoted by intersection ∩.

Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

Input:

∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

o Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.

o It is denoted by intersection minus (-).

Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes

Willians
Curry

6. Cartesian product

o   The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.

o   It is denoted by X.

Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

ρ(STUDENT1, STUDENT)

Integrity Constraints

- o   Integrity constraints are a set of rules. It is used to maintain the quality of information.

- o   Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.

- o   Thus, integrity constraint is used to guard against accidental damage to the database.

   Types of Integrity Constraint
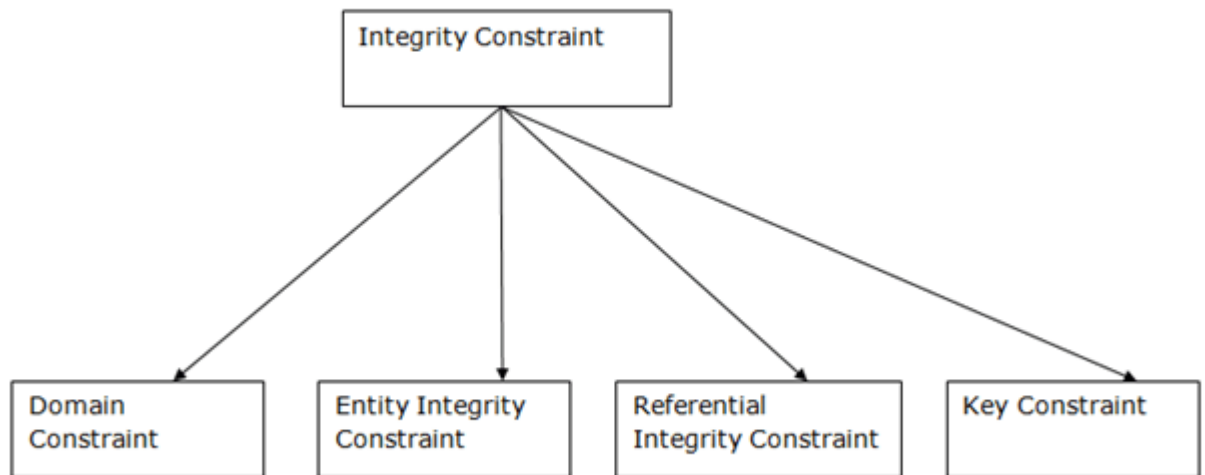
1. Domain constraints

○ Domain constraints can be defined as the definition of a valid set of values for an attribute.

○ The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

○ The entity integrity constraint states that primary key value can't be null.

○ This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.

o A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

o A referential integrity constraint is specified between two tables.

o In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:

(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

Foreign key

Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined

Relationships

(Table 2)

Primary Key

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida
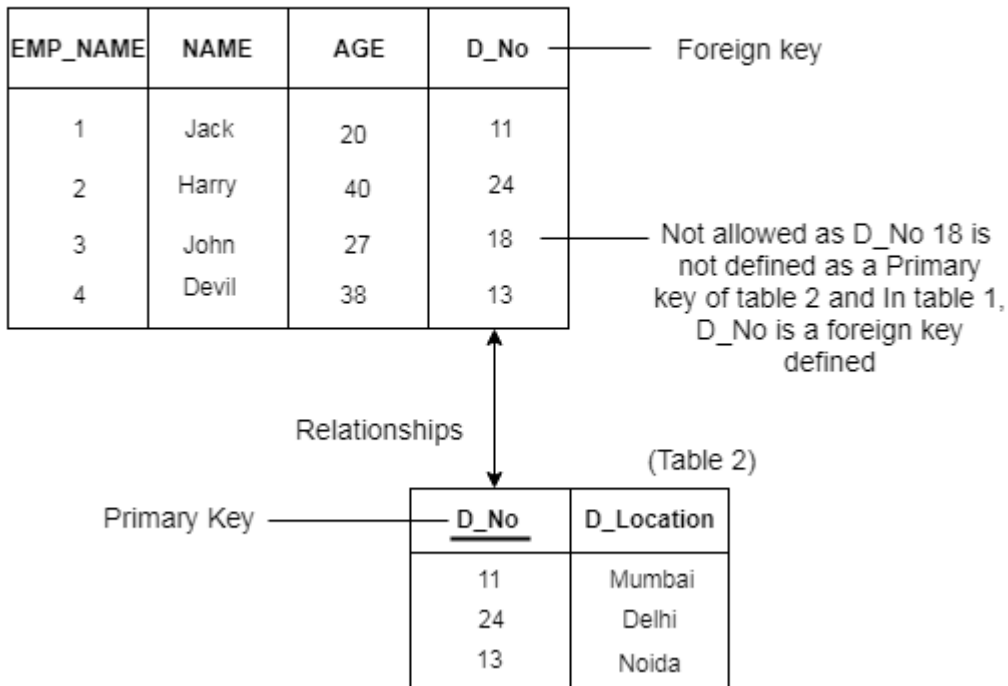
4. Key constraints

o   Keys are the entity set that is used to identify an entity within its entity set uniquely.

o   An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Relational Calculus

- Relational calculus is a non-procedural query language. In the non-procedural query language, the user is concerned with the details of how to obtain the end results.

- The relational calculus tells what to do but never explains how to do.

Types of Relational calculus:
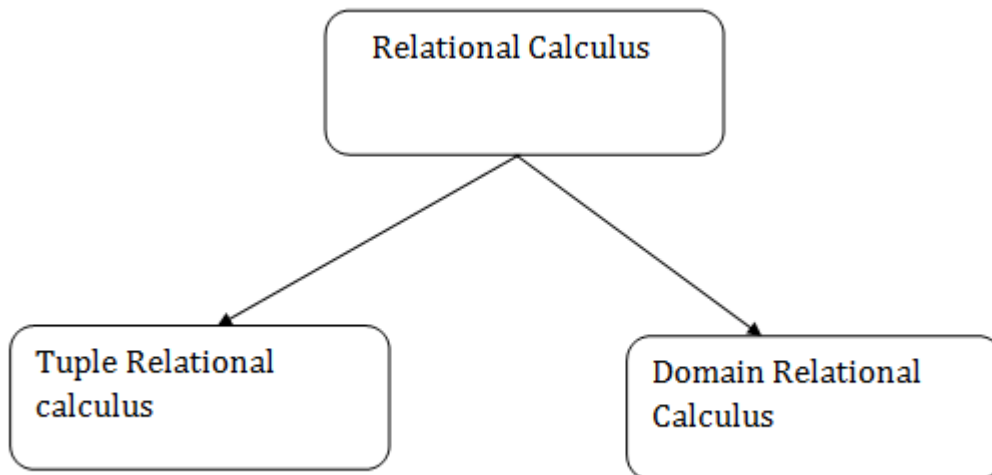
1. Tuple Relational Calculus (TRC)

- The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation.

- The result of the relation can have one or more tuples.

Notation:

1. {T | P (T)}   or {T | Condition (T)}

Where

T is the resulting tuples

P(T) is the condition used to fetch T.

For example:

1. { T.name | Author(T) AND T.article = 'database' }

OUTPUT: This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and Universal Quantifiers (∀).

For example:

1. { R| ∃T ∈ Authors(T.article='database' AND R.name=T.name)}

Output: This query will yield the same result as the previous one.

2. Domain Relational Calculus (DRC)

o The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes.

o Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ∧ (and), ∨ (or) and ¬ (not).

o It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable.

Notation:

1. { a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

Where

a1, a2 are attributes
P stands for formula built by inner attributes

For example:

1. {< article, page, subject > | ∈ javatpoint ∧ subject = 'database'}

Output: This query will yield the article, page, and subject from the relational javatpoint, where the subject is a database.

SQL

o SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).

o It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.

o All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.

o SQL allows users to query the database in a number of ways, using English-like statements.

Rules:

SQL follows the following rules:

- Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.

- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text line.

- Using the SQL statements, you can perform most of the actions in a database.

- SQL depends on tuple relational calculus and relational algebra.

SQL process:

- When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that how to interpret the task.

- In the process, various components are included. These components can be optimization Engine, Query engine, Query dispatcher, classic, etc.

- All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files.

Characteristics of SQL

- SQL is easy to learn.
- SQL is used to access data from relational database management systems.
- SQL can execute queries against the database.
- SQL is used to describe the data.
- SQL is used to define the data in the database and manipulate it when needed.
- SQL is used to create and drop the database and table.
- SQL is used to create a view, stored procedure, function in a database.
- SQL allows users to set permissions on tables, procedures, and views.

Advantages of SQL

There are the following advantages of SQL:

High speed

Using the SQL queries, the user can quickly and efficiently retrieve a large amount of records from a database.

No coding needed

In the standard SQL, it is very easy to manage the database system. It doesn't require a substantial amount of code to manage the database system.

SQL Datatype

o SQL Datatype is used to define the values that a column can contain.

o Every column is required to have a name and data type in the database table.

Datatype of SQL:
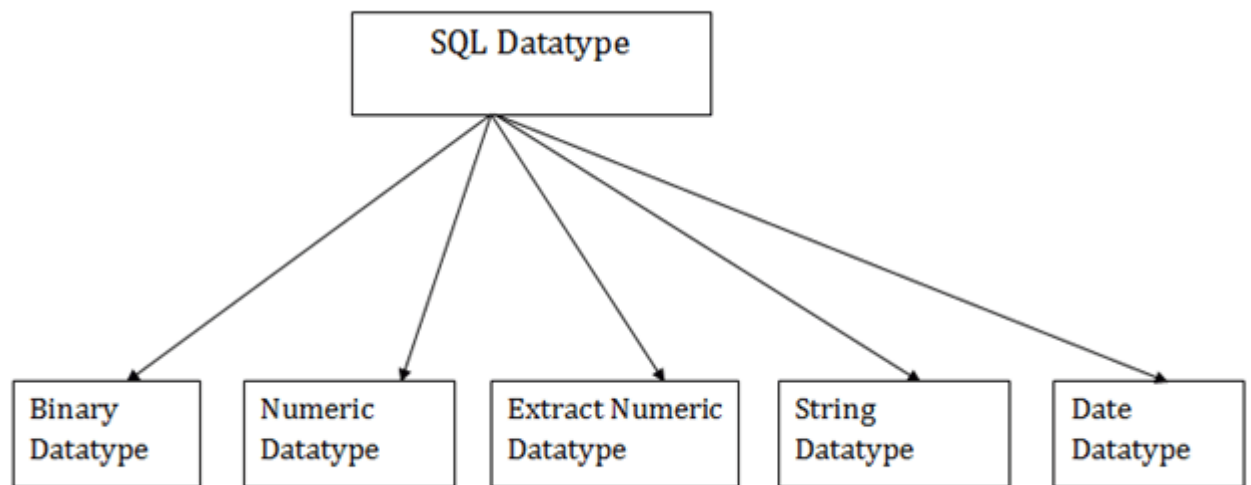
1. Binary Datatypes

There are Three types of binary Datatypes which are given below:

Data Type	Description
binary	It has a maximum length of 8000 bytes. It contains fixed-length binary data.
varbinary	It has a maximum length of 8000 bytes. It contains variable-length binary data.
image	It has a maximum length of 2,147,483,647 bytes. It contains variable-length binary data.

2. Approximate Numeric Datatype :

The subtypes are given below:

Data type	From	To	Description
float	-1.79E + 308	1.79E + 308	It is used to specify a floating-point value e.g. 6.2, 2.9 etc.
real	-3.40e + 38	3.40E + 38	It specifies a single precision floating point number

3. Exact Numeric Datatype

The subtypes are given below:

Data type	Description
int	It is used to specify an integer value.
smallint	It is used to specify small integer value.
bit	It has the number of bits to store.
decimal	It specifies a numeric value that can have a decimal number.
numeric	It is used to specify a numeric value.

4. Character String Datatype

The subtypes are given below:

Data type	Description
char	It has a maximum length of 8000 characters. It contains Fixed-length non-unicode characters.
varchar	It has a maximum length of 8000 characters. It contains variable-length non-unicode characters.
text	It has a maximum length of 2,147,483,647 characters. It contains variable-length

	non-unicode characters.

5. Date and time Datatypes

The subtypes are given below:

Datatype	Description
date	It is used to store the year, month, and days value.
time	It is used to store the hour, minute, and second values.
timestamp	It stores the year, month, day, hour, minute, and the second value.

SQL Commands

o   SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.

o   SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.

SOL Command

```
                            SOL Command
                                 │
    ┌──────────┬──────────┬──────────┬──────────┬──────────┐
    ▼          ▼          ▼          ▼          ▼
  DDL        DML        DCL        TCL        DQL

 ─ Create   ─ Insert   ─ Grant    ─ Commit   ─ Select

 ─ Drop     ─ Update   ─ Revoke   ─ Rollback

 ─ Alter    ─ Delete              ─ Save
                                    point
 ─ Truncate
```
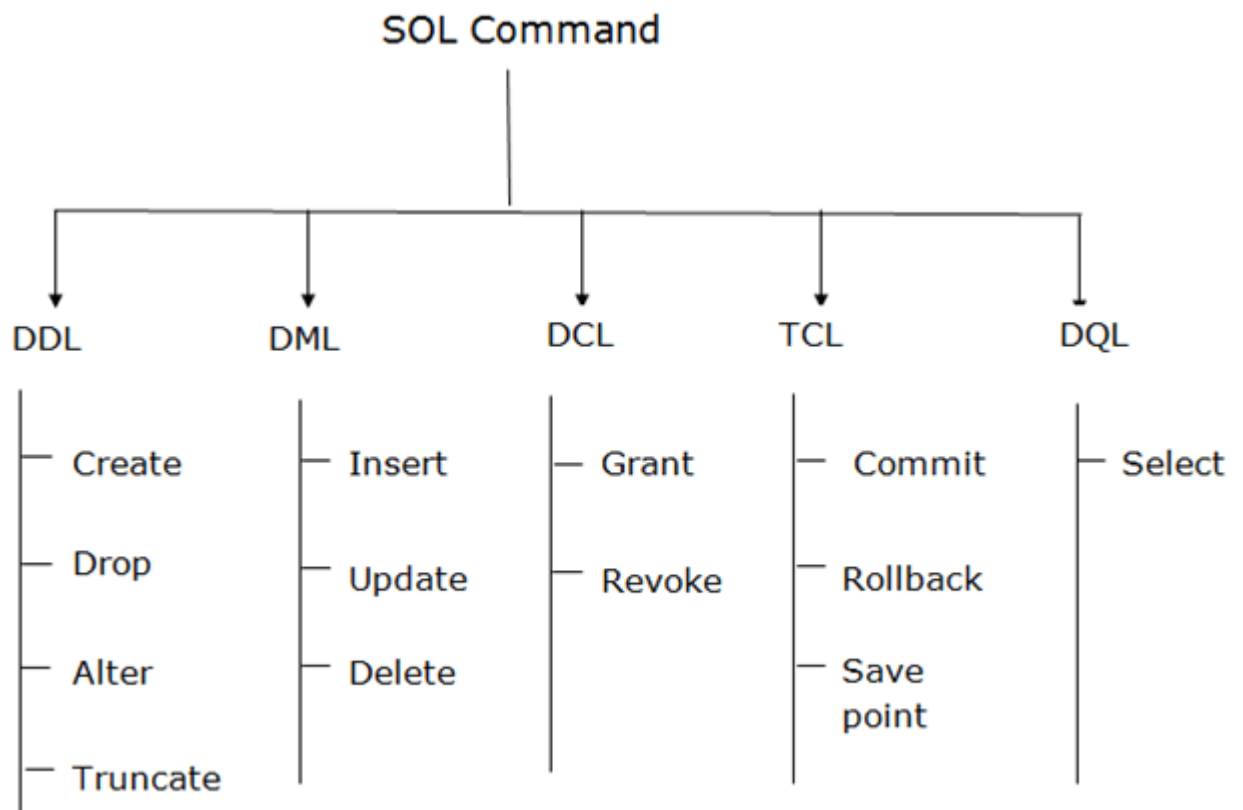
1. Data Definition Language (DDL)

o DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

o All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

o CREATE

o ALTER

o DROP

o TRUNCATE

a. CREATE It is used to create a new table in the database.

Syntax:

CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Example:

    CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100),
    DOB DATE);

b. DROP: It is used to delete both the structure and record stored in the table.

Syntax

    DROP TABLE ;

Example

    DROP TABLE EMPLOYEE;

c. ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

    ALTER TABLE table_name ADD column_name COLUMN-definition;

To modify existing column in the table:

    ALTER TABLE MODIFY(COLUMN DEFINITION....);

EXAMPLE

    ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));

    ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

d. TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax:

    TRUNCATE TABLE table_name;

Example:

    TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language

o   DML commands are used to modify the database. It is responsible for all form of changes in the database.

o   The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

o   INSERT

o   UPDATE

o   DELETE

a. INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE_NAME (col1, col2, col3,.... col N) VALUES (value1, value 2, value3, ... valueN);

Or

INSERT INTO TABLE_NAME VALUES (value1, value2, value3, ... valueN);

For example:

INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");

b. UPDATE: This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [W HERE CONDITION]

For example:

UPDATE students   SET User_Name = 'Sonoo'   WHERE Student_Id = '3'

c. DELETE: It is used to remove one or more row from a table.

Syntax:

DELETE FROM table_name [WHERE condition];

For example:

DELETE FROM javatpoint WHERE Author="Sonoo";

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

o   Grant

o   Revoke

a. Grant: It is used to give user access privileges to a database.

Example

GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

b. Revoke: It is used to take back permissions from the user.

Example

   REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

4. Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

o   COMMIT

o   ROLLBACK

o   SAVEPOINT

a. Commit: Commit command is used to save all the transactions to the database.

Syntax:

   COMMIT;

Example:

   DELETE FROM CUSTOMERS WHERE AGE = 25; COMMIT;

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

   ROLLBACK;

Example:

   DELETE FROM CUSTOMERS WHERE AGE = 25; ROLLBACK;

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

   SAVEPOINT SAVEPOINT_NAME;

5. Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

o   SELECT

a. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

   SELECT expressions   FROM TABLES   WHERE conditions;

For example:

   SELECT emp_name   FROM employee WHERE age > 20;

SQL Operator

There are various types of SQL operator:
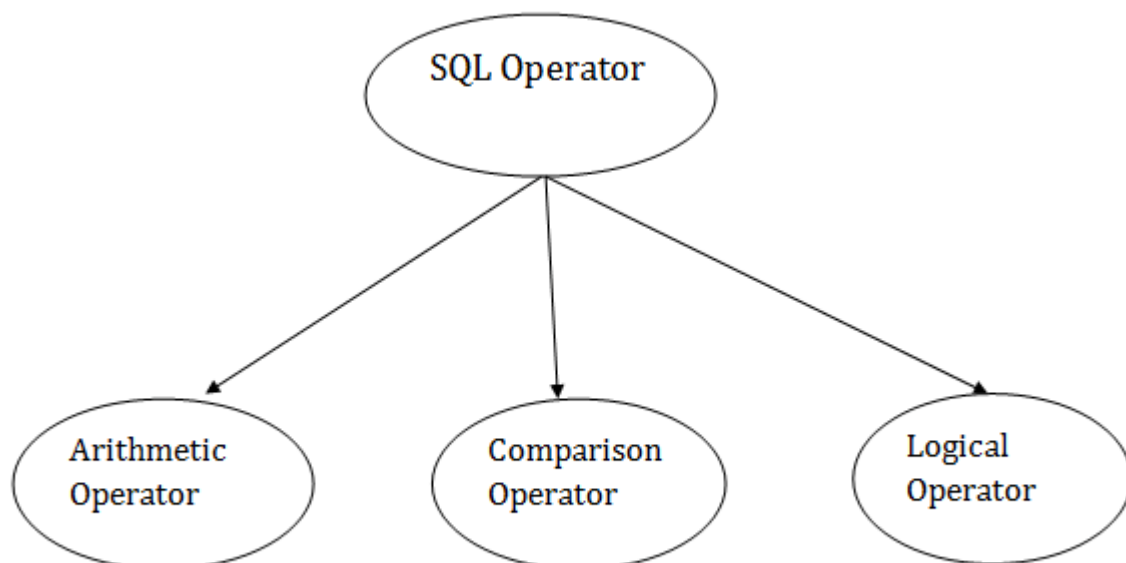
SQL Arithmetic Operators

Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.

Operator	Description	Example
+	It adds the value of both operands.	a+b will give 30
-	It is used to subtract the right-hand operand from the left-hand operand.	a-b will give 10
*	It is used to multiply the value of both operands.	a*b will give 200
/	It is used to divide the left-hand operand by the right-hand operand.	a/b will give 2
%	It is used to divide the left-hand operand by the right-hand operand and returns reminder.	a%b will give 0

SQL Comparison Operators:

Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.

Operator	Description	Example
=	It checks if two operands values are equal or not, if the values are queal then condition becomes true.	(a=b) is not true
!=	It checks if two operands values are equal or not, if values are not equal, then condition becomes true.	(a!=b) is true
<>	It checks if two operands values are equal or not, if values are not equal then condition becomes true.	(a<>b) is true
>	It checks if the left operand value is greater than right operand value, if yes then condition becomes true.	(a>b) is not true
<	It checks if the left operand value is less than right operand value, if yes then condition becomes true.	(a<b) is true
>=	It checks if the left operand value is greater than or equal to	(a>=b) is not true

	the right operand value, if yes then condition becomes true.	
<=	It checks if the left operand value is less than or equal to the right operand value, if yes then condition becomes true.	(a<=b) is true
!<	It checks if the left operand value is not less than the right operand value, if yes then condition becomes true.	(a!=b) is not true
!>	It checks if the left operand value is not greater than the right operand value, if yes then condition becomes true.	(a!>b) is true

SQL Logical Operators

There is the list of logical operator used in SQL:

Operator	Description
ALL	It compares a value to all values in another value set.
AND	It allows the existence of multiple conditions in an SQL statement.
ANY	It compares the values in the list according to the condition.
BETWEEN	It is used to search for values that are within a set of values.
IN	It compares a value to that specified list value.
NOT	It reverses the meaning of any logical operator.
OR	It combines multiple conditions in SQL statements.
EXISTS	It is used to search for the presence of a row in a specified table.
LIKE	It compares a value to similar values using wildcard operator.

SQL Table

- o SQL Table is a collection of data which is organized in terms of rows and columns. In DBMS, the table is known as relation and row as a tuple.

o Table is a simple form of data storage. A table is also considered as a convenient representation of relations.

Let's see an example of the **EMPLOYEE** table:

EMP_ID	EMP_NAME	CITY	PHONE_NO
1	Kristen	Washington	7289201223
2	Anna	Franklin	9378282882
3	Jackson	Bristol	9264783838
4	Kellan	California	7254728346
5	Ashley	Hawaii	9638482678

In the above table, "EMPLOYEE" is the table name, "EMP_ID", "EMP_NAME", "CITY", "PHONE_NO" are the column names. The combination of data of multiple columns forms a row, e.g., 1, "Kristen", "Washington" and 7289201223 are the data of one row.

Operation on Table

1. Create table
2. Drop table
3. Delete table
4. Rename table

SQL Create Table

SQL create table is used to create a table in the database. To define the table, you should define the name of the table and also define its columns and column's data type.

Syntax

1. create table "table_name"
2. ("column1" "data type",
3. "column2" "data type",
4. "column3" "data type",
5. ...

6. "columnN" "data type");

Example

1. SQL> CREATE TABLE EMPLOYEE (

2. EMP_ID INT   NOT NULL,

3. EMP_NAME VARCHAR (25) NOT NULL,

4. PHONE_NO INT   NOT NULL,

5. ADDRESS CHAR (30),

6. PRIMARY KEY (ID)

7.);

If you create the table successfully, you can verify the table by looking at the message by the SQL server. Else you can use DESC command as follows:

SQL> DESC EMPLOYEE;

Field	Type	Null	Key	Default	Extra
EMP_ID	int(11)	NO	PRI	NULL	
EMP_NAME	varchar(25)	NO		NULL	
PHONE_NO	NO	int(11)		NULL	
ADDRESS	YES			NULL	char(30)

Now you have an EMPLOYEE table in the database, and you can use the stored information related to the employees.

Drop table

A SQL drop table is used to delete a table definition and all the data from a table. When this command is executed, all the information available in the table is lost forever, so you have to very careful while using this command.

Syntax

DROP TABLE "table_name";

Firstly, you need to verify the **EMPLOYEE** table using the following command:

SQL> DESC EMPLOYEE;

Field	Type	Null	Key	Default	Extra
EMP_ID	int(11)	NO	PRI	NULL	
EMP_NAME	varchar(25)	NO		NULL	
PHONE_NO	NO	int(11)		NULL	
ADDRESS	YES			NULL	char(30)

This table shows that EMPLOYEE table is available in the database, so we can drop it as follows:

SQL>DROP TABLE EMPLOYEE;

As this shows that the table is dropped, so it doesn't display it.

SQL DELETE table

In SQL, DELETE statement is used to delete rows from a table. We can use WHERE condition to delete a specific row from a table. If you want to delete all the records from the table, then you don't need to use the WHERE clause.

Syntax

DELETE FROM table_name WHERE condition;

Example

Suppose, the EMPLOYEE table having the following records:

EMP_ID	EMP_NAME	CITY	PHONE_NO	SALARY
1	Kristen	Chicago	9737287378	150000

2	Russell	Austin	9262738271	200000
3	Denzel	Boston	7353662627	100000
4	Angelina	Denver	9232673822	600000
5	Robert	Washington	9367238263	350000
6	Christian	Los angels	7253847382	260000

The following query will DELETE an employee whose ID is 2.

SQL> DELETE FROM EMPLOYEE

WHERE EMP_ID = 3;

Now, the EMPLOYEE table would have the following records.

EMP_ID	EMP_NAME	CITY	PHONE_NO	SALARY
1	Kristen	Chicago	9737287378	150000
2	Russell	Austin	9262738271	200000
4	Angelina	Denver	9232673822	600000
5	Robert	Washington	9367238263	350000
6	Christian	Los angels	7253847382	260000

If you don't specify the WHERE condition, it will remove all the rows from the table.

DELETE FROM EMPLOYEE;

Now, the EMPLOYEE table would not have any records.

SQL SELECT Statement

In SQL, the SELECT statement is used to query or retrieve data from a table in the database. The returns data is stored in a table, and the result table is known as result-set.

Syntax

SELECT column1, column2, ... FROM table_name;

Here, the expression is the field name of the table that you want to select data from.

Use the following syntax to select all the fields available in the table:

SELECT * FROM table_name;

Example:

EMPLOYEE

EMP_ID	EMP_NAME	CITY	PHONE_NO	SALARY
1	Kristen	Chicago	9737287378	150000
2	Russell	Austin	9262738271	200000
3	Angelina	Denver	9232673822	600000
4	Robert	Washington	9367238263	350000
5	Christian	Los angels	7253847382	260000

To fetch the EMP_ID of all the employees, use the following query:

1. SELECT EMP_ID FROM EMPLOYEE;

Output

EMP_ID
1
2
3

4	
5	

To fetch the EMP_NAME and SALARY, use the following query:

SELECT EMP_NAME, SALARY FROM EMPLOYEE;

EMP_NAME	SALARY
Kristen	150000
Russell	200000
Angelina	600000
Robert	350000
Christian	260000

SQL INSERT Statement

The SQL INSERT statement is used to insert a single or multiple data in a table. In SQL, You can insert the data in two ways:

1. Without specifying column name

2. By specifying column name

Sample Table

EMPLOYEE

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42

4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36

1. Without specifying column name

If you want to specify all column values, you can specify or ignore the column values.

Syntax

INSERT INTO TABLE_NAME

VALUES (value1, value2, value 3, ... Value N);

Query

INSERT INTO EMPLOYEE VALUES (6, 'Marry', 'Canada', 600000, 48);

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

2. By specifying column name

To insert partial column values, you must have to specify the column names.

Syntax

INSERT INTO TABLE_NAME

[(col1, col2, col3,.... col N)]

VALUES (value1, value2, value 3, ... Value N);

Query

INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, AGE) VALUES (7, 'Jack', 40)
;

Output: After executing this query, the table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48
7	Jack	null	null	40

SQL Update Statement

The SQL UPDATE statement is used to modify the data that is already in the database. The condition in the WHERE clause decides that which row is to be updated.

Syntax

UPDATE table_name
SET column1 = value1, column2 = value2, ... WHERE condition;

Sample Table

EMPLOYEE

EMP_ID	EMP_NAME	CITY	SALARY	AGE

1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

Updating single record

Update the column EMP_NAME and set the value to 'Emma' in the row where SALARY is 500000.

Syntax

UPDATE table_name   SET column_name = value WHERE condition;

Query

UPDATE EMPLOYEE   SET EMP_NAME = 'Emma' WHERE SALARY = 500000;

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Emma	Washington	500000	29
5	Russell	Los angels	200000	36

6	Marry	Canada	600000	48

Updating multiple records

If you want to update multiple columns, you should separate each field assigned with a comma. In the EMPLOYEE table, update the column EMP_NAME to 'Kevin' and CITY to 'Boston' where EMP_ID is 5.

Syntax

UPDATE table_name

SET column_name = value1, column_name2 = value2

WHERE condition;

Query

UPDATE EMPLOYEE

SET EMP_NAME = 'Kevin', City = 'Boston'

WHERE EMP_ID = 5;

Output

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Kevin	Boston	200000	36
6	Marry	Canada	600000	48

Without use of WHERE clause

If you want to update all row from a table, then you don't need to use the WHERE clause. In the EMPLOYEE table, update the column EMP_NAME as 'Harry'.

Syntax

UPDATE table_name

SET column_name = value1;

Query

UPDATE EMPLOYEE

SET EMP_NAME = 'Harry';

Output

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Harry	Chicago	200000	30
2	Harry	Austin	300000	26
3	Harry	Denver	100000	42
4	Harry	Washington	500000	29
5	Harry	Los angels	200000	36
6	Harry	Canada	600000	48

SQL DELETE Statement

The SQL DELETE statement is used to delete rows from a table. Generally, DELETE statement removes one or more records form a table.

Syntax

DELETE FROM table_name WHERE some_condition;

Sample Table

EMPLOYEE

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

Deleting Single Record

Delete the row from the table EMPLOYEE where EMP_NAME = 'Kristen'. This will delete only the fourth row.

Query

DELETE FROM EMPLOYEE

WHERE EMP_NAME = 'Kristen';

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

Deleting Multiple Record

Delete the row from the EMPLOYEE table where AGE is 30. This will delete two rows(first and third row).

Query

DELETE FROM EMPLOYEE WHERE AGE= 30;

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

Delete all of the records

Delete all the row from the EMPLOYEE table. After this, no records left to display. The EMPLOYEE table will become empty.

Syntax

DELETE * FROM table_name;

or

DELETE FROM table_name;

Query

DELETE FROM EMPLOYEE;

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE

Views in SQL

- Views in SQL are considered as a virtual table. A view also contains rows and columns.

- To create the view, we can select the fields from one or more tables present in the database.

- A view can either have specific rows based on certain condition or all the rows of a table.

Sample table:

Student_Detail

STU_ID	NAME	ADDRESS
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

Student_Marks

STU_ID	NAME	MARKS	AGE
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

1. Creating view

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

Syntax:

CREATE VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE condition;

2. Creating View from a single table

In this example, we create a View named DetailsView from the table Student_Detail.

Query:

CREATE VIEW DetailsView AS

SELECT NAME, ADDRESS

FROM Student_Details

WHERE STU_ID < 4;

Just like table query, we can query the view to view the data.

SELECT * FROM DetailsView;

Output:

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

3. Creating View from multiple tables

View from multiple tables can be created by simply include multiple tables in the SELECT statement.

In the given example, a view is created named MarksView from two tables Student_Detail and Student_Marks.

Query:

CREATE VIEW MarksView AS

SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS

FROM Student_Detail, Student_Mark

WHERE Student_Detail.NAME = Student_Marks.NAME;

To display data of View MarksView:

SELECT * FROM MarksView;

NAME	ADDRESS	MARKS
Stephan	Delhi	97
Kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

4. Deleting View

A view can be deleted using the Drop View statement.

Syntax

1. DROP VIEW view_name;

Example:

If we want to delete the View **MarksView**, we can do this as:

1. DROP VIEW MarksView;

SQL Index

- o Indexes are special lookup tables. It is used to retrieve data from the database very fast.

- o An Index is used to speed up select queries and where clauses. But it shows down the data input with insert and update statements. Indexes can be created or dropped without affecting the data.

- o An index in a database is just like an index in the back of a book.

- o **For example:** When you reference all pages in a book that discusses a certain topic, you first have to refer to the index, which alphabetically lists all the topics and then referred to one or more specific page numbers.

1. Create Index statement

It is used to create an index on a table. It allows duplicate value.

Syntax

CREATE INDEX index_name

ON table_name (column1, column2, ...);

Example

CREATE INDEX idx_name

ON Persons (LastName, FirstName);

2. Unique Index statement

It is used to create a unique index on a table. It does not allow duplicate value.

Syntax

CREATE UNIQUE INDEX index_name

ON table_name (column1, column2, ...);

Example

1. CREATE UNIQUE INDEX websites_idx
2. ON websites (site_name);

3. Drop Index Statement

It is used to delete an index in a table.

Syntax

DROP INDEX index_name;

Example

DROP INDEX websites_idx;

References:

[1] https://www.geeksforgeeks.org/introduction-of-b-tree-2/

[2] https://www.javatpoint.com/dbms-sequential-file-organization