Unit-5

Concurrency control concept comes under the Transaction in database management system (DBMS). It is a procedure in DBMS which helps us for the management of two simultaneous processes to execute without conflicts between each other, these conflicts occur in multi user systems.

Concurrency can simply be said to be executing multiple transactions at a time. It is required to increase time efficiency. If many transactions try to access the same data, then inconsistency arises. Concurrency control required to maintain consistency data.

In a multi-user system, multiple users can access and use the same database at one time, which is known as the concurrent execution of the database. It means that the same database is executed simultaneously on a multi-user system by different users.

While working on the database transactions, there occurs the requirement of using the database by multiple users for performing different operations, and in that case, concurrent execution of the database is performed.

The thing is that the simultaneous execution that is performed should be done in an interleaved manner, and no operation should affect the other executing operations, thus maintaining the consistency of the database. Thus, on making the concurrent execution of the transaction operations, there occur several challenging problems that need to be solved.

Concurrency control: Concurrency control concept comes under the Transaction in database management system (DBMS). It is a procedure in DBMS which helps us for the management of two simultaneous processes to execute without conflicts between each other, these conflicts occur in multi user systems.

Concurrency can simply be said to be executing multiple transactions at a time. It is required to increase time efficiency. If many transactions try to access the same data, then inconsistency arises. Concurrency control required to maintain consistency data.

For example, if we take ATM machines and do not use concurrency, multiple persons cannot draw money at a time in different places. This is where we need concurrency.

Advantages:

The advantages of concurrency control are as follows −

- Waiting time will be decreased.
- Response time will decrease.
- Resource utilization will increase.
- System performance & Efficiency is increased.

Control concurrency

The simultaneous execution of transactions over shared databases can create several data integrity and consistency problems.

For example, if too many people are logging in the ATM machines, serial updates and synchronization in the bank servers should happen whenever the transaction is done, if not it gives wrong information and wrong data in the database.

Main problems in using Concurrency

The problems which arise while using concurrency are as follows −

- **Updates will be lost** − One transaction does some changes and another transaction deletes that change. One transaction nullifies the updates of another transaction.
- **Uncommitted Dependency or dirty read problem** − On variable has updated in one transaction, at the same time another transaction has started and deleted the value of the variable there the variable is not getting updated or committed that has been done on the first transaction this gives us false values or the previous values of the variables this is a major problem.
- **Inconsistent retrievals** − One transaction is updating multiple different variables, another transaction is in a process to update those variables, and the problem occurs is inconsistency of the same variable in different instances.

Concurrency control techniques

The concurrency control techniques are as follows −

Locking

Lock guaranties exclusive use of data items to a current transaction. It first accesses the data items by acquiring a lock, after completion of the transaction it releases the lock.

2PL (Two Phase Locking Protocol):

Types of Locks

The types of locks are as follows −

- Shared Lock [Transaction can read only the data item values]
- Exclusive Lock [Used for both read and write data item values]

Shared lock:

- It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction.
- It can be shared between the transactions because when the transaction holds a lock, then it can't update the data on the data item.

Exclusive lock:

- In the exclusive lock, the data item can be both reads as well as written by the transaction.
- This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

Time Stamping

- The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.
- The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.
- The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.
- Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at 009 times. T1 has the higher priority, so it executes first as it is entered the system first.
- The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

Basic Timestamp ordering protocol works as follows:

1. Check the following condition whenever a transaction Ti issues a **Read (X)** operation:

- If W_TS(X) >TS(Ti) then the operation is rejected.
- If W_TS(X) <= TS(Ti) then the operation is executed.
- Timestamps of all the data items are updated.

2. Check the following condition whenever a transaction Ti issues a **Write(X)** operation:

- If TS(Ti) < R_TS(X) then the operation is rejected.
- If TS(Ti) < W_TS(X) then the operation is rejected and Ti is rolled back otherwise the operation is executed.

Multi version Technique Based on Timestamp Ordering

- In this method, several versions X1, X2, ..., Xk of each data item X are maintained. For each version, the value of version Xi and the following two timestamps are kept:
-
- read_TS(Xi). The read timestamp of Xi is the largest of all the timestamps of transactions that have successfully read version Xi.
- write_TS(Xi). The write timestamp of Xi is the timestamp of the transaction that wrote the value of version Xi.
-
- Whenever a transaction T is allowed to execute a write_item(X) operation, a new version Xk+1 of item X is created, with both the write_TS(Xk+1) and the read_TS(Xk+1) set to TS(T). Correspondingly, when a transaction T is allowed to read the value of version Xi, the value of read_TS(Xi) is set to the larger of the current read_TS(Xi) and TS(T).
- To ensure serializability, the following rules are used:
-
- If transaction T issues a write_item(X) operation, and version i of X has the highest write_TS(Xi) of all versions of X that is also less than or equal to TS(T), and read_TS(Xi) > TS(T), then abort and roll back transaction T; otherwise, create a new version Xj of X with read_TS(Xj) = write_TS(Xj) = TS(T).
-
- If transaction T issues a read_item(X) operation, find the version i of X that has the highest write_TS(Xi) of all versions of X that is also less than or equal to TS(T); then return the value of Xi to transaction T, and set the value of read_TS(Xi) to the larger of TS(T) and the current read_TS(Xi).
-
- As we can see in case 2, a read_item(X) is always successful, since it finds the appropriate version Xi to read based on the write_TS of the various existing versions of X. In case 1, however, transaction T may be aborted and rolled back. This happens if T attempts to write a version of X that should have been read by another transaction T whose timestamp is read_TS(Xi); however, T has already read version Xi, which was written by the transaction with timestamp equal to write_TS(Xi). If this conflict occurs, T is rolled back; otherwise, a new version of X, written by transaction T, is created. Notice that if T is rolled back, cascading rollback may occur.

Hence, to ensure recoverability, a transaction T should not be allowed to commit until after all the transactions that have written some version that T has read have committed.

Validation Based Protocol:

is also called Optimistic Concurrency Control Technique. This protocol is used in DBMS (Database Management System) for avoiding concurrency in transactions. It is called optimistic because of the assumption it makes, i.e. very less interference occurs, therefore, there is no need for checking while the transaction is executed.
In this technique, no checking is done while the transaction is been executed. Until the transaction end is reached updates in the transaction are not applied directly to the database. All updates are applied to local copies of data items kept for the transaction. At the end of transaction execution, while execution of the transaction, a validation phase checks whether any of transaction updates violate serializability. If there is no violation of serializability the transaction is committed and the database is updated; or else, the transaction is updated and then restarted.
Optimistic Concurrency Control is a three-phase protocol. The three phases for validation based protocol:

1. Read Phase:
   Values of committed data items from the database can be read by a transaction. Updates are only applied to local data versions.

2. Validation Phase:
   Checking is performed to make sure that there is no violation of serializability when the transaction updates are applied to the database.

3. Write Phase:
   On the success of the validation phase, the transaction updates are applied to the database, otherwise, the updates are discarded and the transaction is slowed down.

The idea behind optimistic concurrency is to do all the checks at once; hence transaction execution proceeds with a minimum of overhead until the validation phase is reached. If there is not much interference among transactions most of them will have successful validation, otherwise, results will be discarded and restarted later. These circumstances are not much favourable for optimization techniques, since, the assumption of less interference is not satisfied.

Validation based protocol is useful for rare conflicts. Since only local copies of data are included in rollbacks, cascading rollbacks are avoided. This method is not favourable for longer transactions because they are more likely to have conflicts and might be repeatedly rolled back due to conflicts with short transactions.

In order to perform the Validation test, each transaction should go through the various phases as described above. Then, we must know about the following three time-stamps that we assigned to transaction Ti, to check its validity:
1. Start(Ti): It is the time when Ti started its execution.
2. Validation(Ti): It is the time when Ti just finished its read phase and begin its validation phase.
3. Finish(Ti): the time when Ti end it's all writing operations in the database under write-phase.

Two more terms that we need to know are:

1. Write_set: of a transaction contains all the write operations that Ti performs.
2. Read_set: of a transaction contains all the read operations that Ti performs.

In the Validation phase for transaction Ti the protocol inspect that Ti doesn't overlap or intervene with any other transactions currently in their validation phase or in committed. The validation phase for Ti checks that for all transaction Tj one of the following below conditions must hold to being validated or pass validation phase:

1. Finish(Tj)<Starts(Ti), since Tj finishes its execution means completes its write-phase before Ti started its execution(read-phase). Then the serializability indeed maintained.
2. Ti begins its write phase after Tj completes its write phase, and the read_set of Ti should be disjoint with write_set of Tj.
3. Tj completes its read phase before Ti completes its read phase and both read_set and write_set of Ti are disjoint with the write_set of Tj.

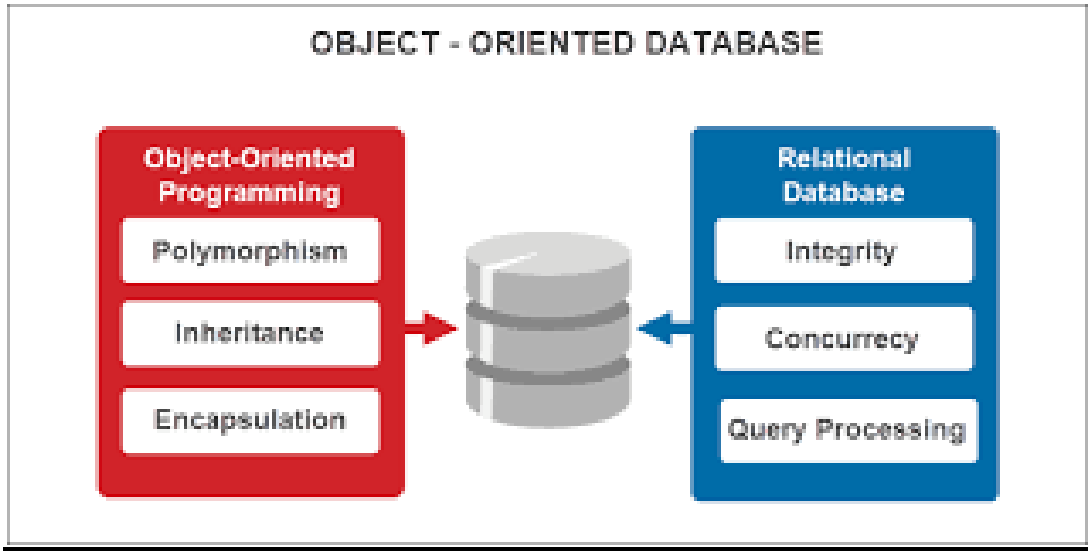
Ex: Here two Transactions Ti and Tj are given, since TS(Tj)<TS(Ti) so the validation phase succeeds in the Schedule-A. It's noteworthy that the final write operations to the database are performed only after the validation of both Ti and Tj. Since Ti reads the old values of x(12) and y(15) while print(x+y) operation unless final write operation take place.

Object-Oriented Database:

Object database management systems (ODBMSs) are based on objects in object-oriented programming (OOP). In OOP, an entity is represented as an object and objects are stored in memory. Objects have members such as fields, properties, and methods. Objects also have a life cycle that includes the creation of an object, use of an object, and deletion of an object. OOP has key characteristics, encapsulation, inheritance, and polymorphism. Today, there are many popular OOP languages such as C++, Java, C#, Ruby, Python, JavaScript, and Perl.

The idea of object databases was originated in 1985 and today has become common for various common OOP languages, such as C++, Java, C#, Smalltalk, and LISP. Common examples are Smalltalk is used in GemStone, LISP is used in Gbase, and COP is used in Vbase.

Object databases are commonly used in applications that require high performance, calculations, and faster results. Some of the common applications that use object databases are real-time systems, architectural & engineering for 3D modeling, telecommunications, and scientific products, molecular science, and astronomy.



Advantages of Object Databases:

ODBMS provide persistent storage to objects. Imagine creating objects in your program and saving them as it is in a database and reading back from the database.

In a typical relational database, the program data is stored in rows and columns. To store and read that data and convert it into program objects in memory requires reading data, loading data into objects, and storing it in memory. Imagine creating a class in your program and saving it as it is in a database, reading back and start using it again.
Object databases bring permanent persistent to objects. Objects can be stored in persistent storage forever.

In typical RDBMS, there is a layer of object-relational mapping that maps database schemas with objects in code. Reading and mapping an object database data to the objects is direct without any API or OR tool. Hence faster data access and better performance.

Some object database can be used in multiple languages. For example, Gemstone database supports C++, Smalltalk and Java programming languages.

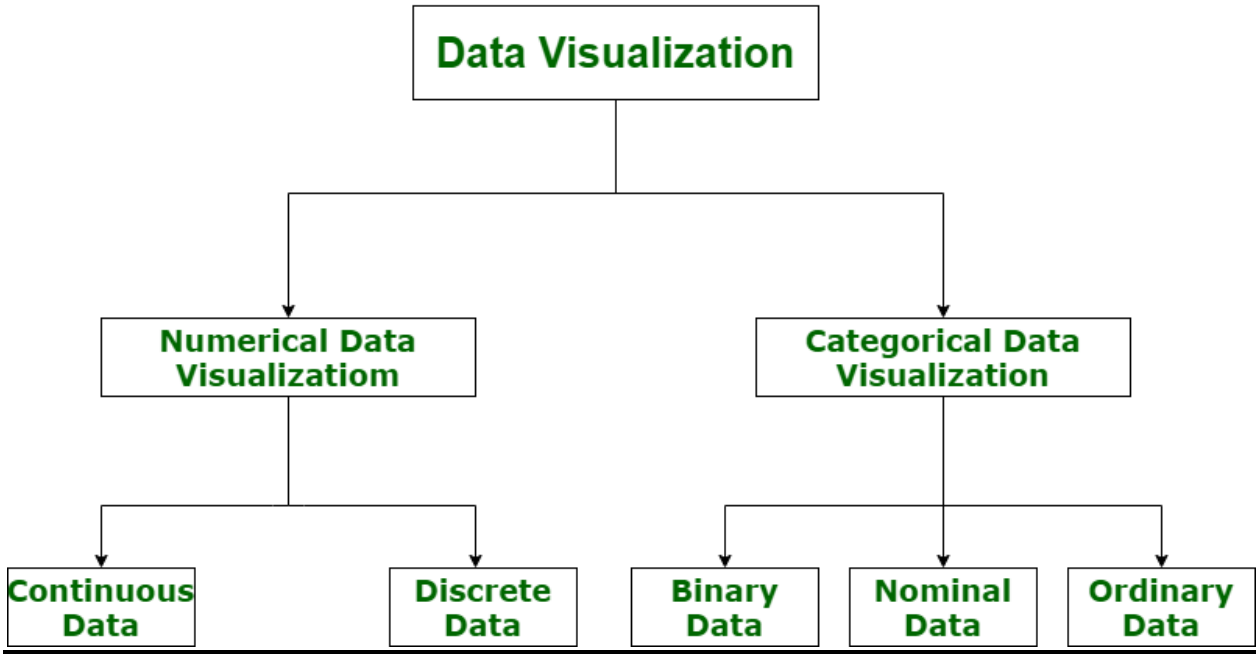
Drawbacks of Object Databases:

- Object databases are not as popular as RDBMS. It is difficult to find object DB developers.
- Not many programming language support object databases.
- RDBMS have SQL as a standard query language. Object databases do not have a standard.
- Object databases are difficult to learn for non-programmers.

Data Visualization: Data visualization is the graphical representation of information and data in a pictorial or graphical format (Example: charts, graphs, and maps). Data visualization tools provide an accessible way to see and understand trends, patterns in data, and outliers. Data visualization tools and technologies are essential to analyzing massive amounts of information and making data-driven decisions. The concept of using pictures is to understand data that has been used for centuries. General types of data visualization are Charts, Tables, Graphs, Maps, and Dashboards.

Categories of Data Visualization

Data visualization is very critical to market research where both numerical and categorical data can be visualized, which helps in an increase in the impact of insights and also helps in reducing the risk of analysis paralysis. So, data visualization is categorized into the following categories:



Advantages of Data Visualization

1. Better Agreement: In business, for numerous periods, it happens that we need to look at the exhibitions of two components or two situations. A conventional methodology is to experience the massive information of both the circumstances and afterward examine it. This will
arly take a great deal of time.

2. A Superior Method: It can tackle the difficulty of placing the information of both perspectives into the pictorial structure. This will unquestionably give a superior comprehension of the circumstances. For instance, Google patterns

assist us with understanding information identified with top ventures or inquiries in pictorial or graphical structures.

3. Simple Sharing of Data: With the representation of the information, organizations present another arrangement of correspondence. Rather than sharing the cumbersome information, sharing the visual data will draw in and pass on across the data which is more absorbable.

4Deals Investigation: With the assistance of information representation, a salesman can, without much of a stretch, comprehend the business chart of items. With information perception instruments like warmth maps, he will have the option to comprehend the causes that are pushing the business numbers up just as the reasons that are debasing the business numbers. Information representation helps in understanding the patterns and furthermore, different variables like sorts of clients keen on purchasing, rehash clients, the impact of topography, and so forth.

5. Discovering Relations between Occasions: A business is influenced by a lot of elements. Finding a relationship between these elements or occasions encourages chiefs to comprehend the issues identified with their business. For instance, the online business market is anything but another thing today. Each time during certain happy seasons, like Christmas or Thanksgiving, the diagrams of online organizations go up. Along these lines, state if an online organization is doing a normal $1 million business in a specific quarter and the business ascends straightaway, at that point they can rapidly discover the occasions compared to it.

6. Investigating Openings and Patterns: With the huge loads of information present, business chiefs can discover the profundity of information in regard to the patterns and openings around them. Utilizing information representation, the specialists can discover examples of the conduct of their clients, subsequently preparing for them to investigate patterns and open doors for business

Mobile Database:

Mobile databases are separate from the main database and can easily be transported to various places. Even though they are not connected to the main database, they can still communicate with the database to share and exchange data.

The mobile database includes the following components −

- The main system database that stores all the data and is linked to the mobile database.
- The mobile database that allows users to view information even while on the move. It shares information with the main database.

- The device that uses the mobile database to access data. This device can be a mobile phone, laptop etc.
- A communication link that allows the transfer of data between the mobile database and the main database.

Advantages of Mobile Databases

Some advantages of mobile databases are −

- The data in a database can be accessed from anywhere using a mobile database. It provides wireless database access.
- The database systems are synchronized using mobile databases and multiple users can access the data with seamless delivery process.
- Mobile databases require very little support and maintenance.
- The mobile database can be synchronized with multiple devices such as mobiles, computer devices, laptops etc.
-

Disadvantages of Mobile Databases

Some disadvantages of mobile databases are −

- The mobile data is less secure than data that is stored in a conventional stationary database. This presents a security hazard.
- The mobile unit that houses a mobile database may frequently lose power because of limited battery. This should not lead to loss of data in database.

XML Database:

XML database is a data persistence software system used for storing the huge amount of information in XML format. It provides a secure place to store XML documents.

You can query your stored data by using XQuery, export and serialize into desired format. XML databases are usually associated with document-oriented databases.

Types of XML databases

There are two types of XML databases.

1. XML-enabled database
2. Native XML database (NXD)

- **Real-time control and monitoring:** Coupled with active database technology, multimedia presentation of information can be very effective means for monitoring and controlling complex tasks Example: Manufacturing operation control.

Web-based database:

The Web-based database management system is one of the essential parts of DBMS and is used to store web application data. A web-based Database management system is used to handle those databases that are having data regarding E-commerce, E-business, blogs, e-mail, and other online applications.

Before acquiring knowledge about the web-based database management system, you need to know some basic terminology about the web. See "What is Web?" which contains more information about it.

At its most simple level, a web database is a set of one or more tables that contain data. Each table has different fields for storing information of various types. These tables can then be linked together in order to manipulate data in useful or interesting ways. In many cases, a table will use a **primary key**, which must be unique for each entry and allows for unambiguous selection of data.

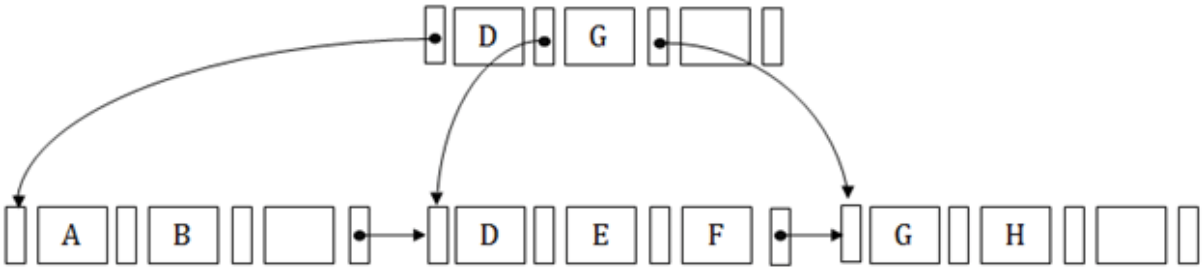
A web database can be used for a range of different purposes. Each field in a table has to have a defined data type. For example, numbers, strings, and dates can all be inserted into a web database. Proper **database design** involves choosing the correct data type for each field in order to reduce memory consumption and increase the speed of access. Although for small databases this often isn't so important, big web databases can grow to millions of entries and need to be well designed to work effectively.

B+ Tree:

- The B+ tree is a balanced binary search tree. It follows a multi-level index format.
- In the B+ tree, leaf nodes denote actual data pointers. B+ tree ensures that all leaf nodes remain at the same height.
- 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

- 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.
- It contains an internal node and leaf node.



Internal node

- An internal node of the B+ tree can contain at least n/2 record pointers except the root node.
- At most, an internal node of the tree contains n pointers.

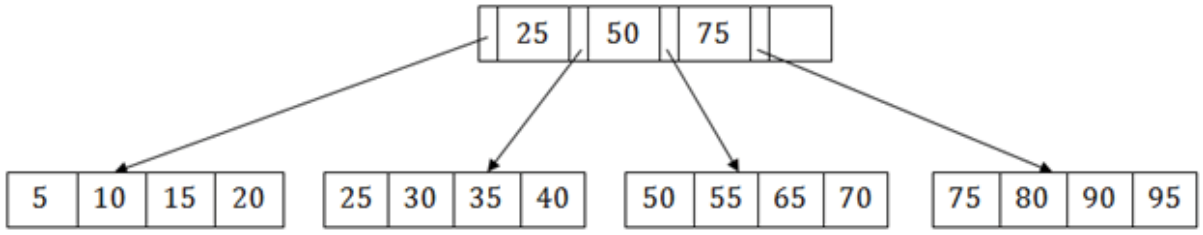
Leaf node

- The leaf node of the B+ tree can contain at least n/2 record pointers and n/2 key values.
- At most, a leaf node contains n record pointer and n key values.
- 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.

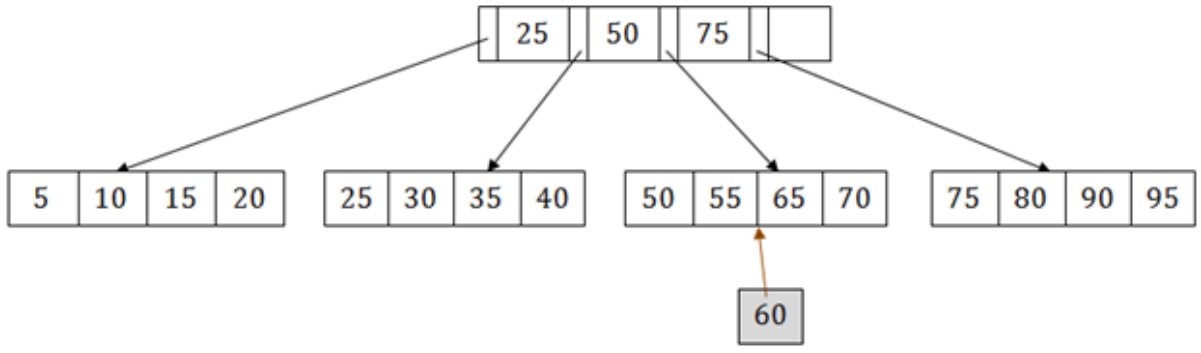
So, in the intermediary node, we will find a branch between 50 and 75 nodes. Then at the end, we will be redirected to the third leaf node. Here DBMS will perform a sequential search to find 55.



B+ Tree Insertion

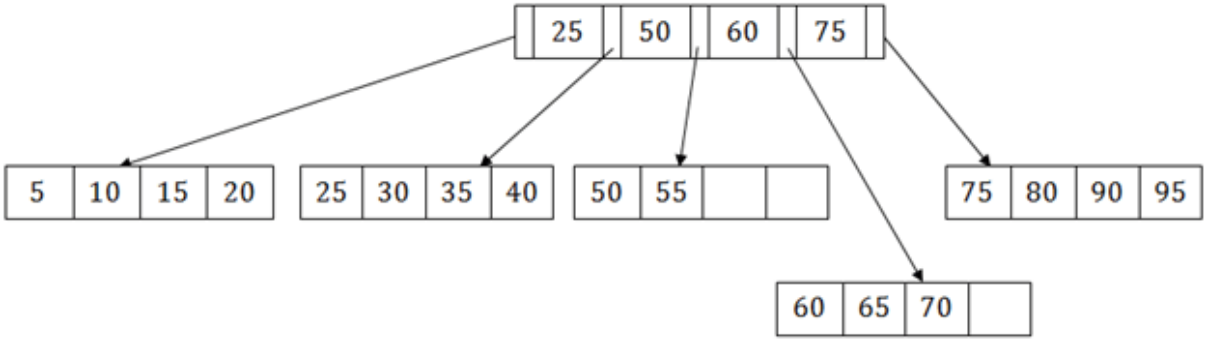
Suppose we want to insert a record 60 in the below structure. It will go to the 3rd leaf node after 55. It is a balanced tree, and a leaf node of this tree is already full, so we cannot insert 60 there.

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: