**Q. What is Nth Highest Salary Example in MySQL and SQL Server ?  
(nth means highest or 2nd highest or 3rd highest or anything.)**   
---------------------------------------------------------------  
**Q. Write a SQL query to get the nth highest salary from the Employee table.**  
+----+--------+  
| Id | Salary |  
+----+--------+  
| 1  | 100    |  
| 2  | 200    |  
| 3  | 300    |  
+----+--------+  
  
For example, given the above Employee table, the nth highest salary where n = 2 is 200. If there is no nth highest salary, then the query should return null.  
**1. Accepted Solution**

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

DECLARE M INT;

SET M = N - 1;

RETURN (

# WRITE our MySQL query statement below.

SELECT DISTINCT Salary FROM Employee ORDER BY Salary DESC LIMIT M, 1

);

END

**2. Alternate Solution**

This is another solution to find Nth highest salary problem,

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

RETURN (

  # WRITE our MySQL query statement below.

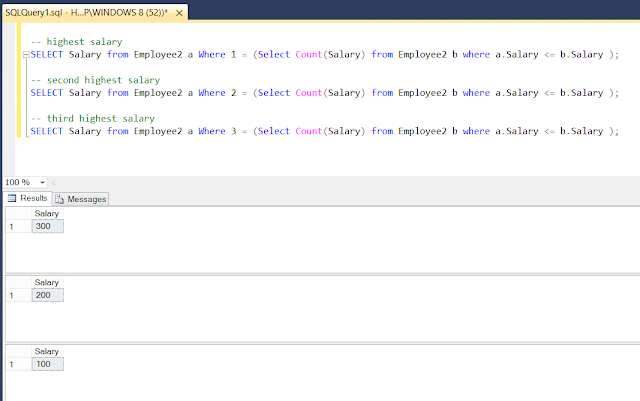
SELECT Salary FROM Employee a

WHERE N = (SELECT COUNT(Salary) FROM Employee b

WHERE a.Salary &lt;= b.Salary)

    );

 END



**Q. Reasons why using  SELECT \* in SQL Query is a Bad Practice**

**Ans**.

**1) Unnecessary IO**

By using SELECT \* we can be returning unnecessary data that will just be ignored but fetching that data is not free of cost. This result in some wasteful IO cycles at DB end, since we will be reading all of that data off the pages, then perhaps we could have read the data from index pages. This can make our query a little bit slow as well. See The Complete SQL BootCamp learn more.



**2) Increased network traffic**

SELECT \* return more data than required to the client which in turn will use more networkbandwidth. This increase in network bandwidth also means that data will take a longer time to reach the client application which could be SSMS or our Java application server.

**3) More application memory**

due to this increase in data, our application may require more memory just to hold unnecessary data which it will not be used but coming from Microsoft SQL Server or any other database we are connecting to.

**4) Dependency on Order of Columns on ResultSet**

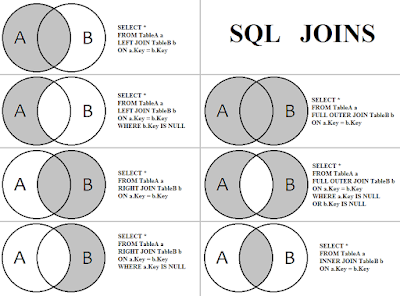
When we use the SELECT \* query in our application and have any dependency on the order of column, which we should not, the ordering of the result set will change if we add a new column or change the order of columns.

**5**) **Breaking Views while adding new columns to a table**

When we use SELECT \* in views then we create subtle bugs if a new column has been added and the old one is removed from the table. Why? because our view will not break but start returning an incorrect result.  
 To avoid that, we should always use WITH SCHEMA BINDING with views. This will also prevent we from using SELECT \* in views.

**6) Conflict in JOIN Query**

When we use SELECT \* in the JOIN query, we can introduce complications when multiple tables have columns with the same name like status, active, name, etc.  
  
On a simple query, this might be fine but when we try to order by one of these columns or use the query in a CTE or derived table, we will need to make some adjustments.

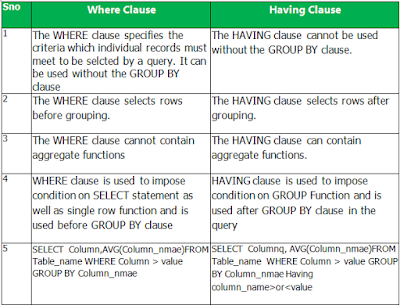


**7)  Copying data from one table to other**

When we use SELECT \* into INSERT .. SELECT statement, which is a common way to copy data from one table to another, we could potentially copy incorrect data into the incorrect column if the order of column is not same between two tables.  
 Some programmers think that using SELECT \* vs SELECT 1 in our  EXISTS code is faster because query parser had to do extra work to validate the static value.  
 That might have been true long ago but nowadays parser has become smart enough to know that within an EXISTS clause, the SELECT list is completely irrelevant.  
 It's always better to use the explicit column list in the SELECT query than a \* wildcard. It not only improves the performance but also makes our code more explicit. It also helps we to create maintainable code, which will not break when we add new columns to our table especially if we have views that refer to the original table.  
  
**Q. Difference between WHERE and HAVING clause in SQL?**

**Ans** . The main difference between WHERE and HAVING clause comes when used together with GROUP BY clause, In that case WHERE is used to filter rows before grouping and HAVING is used to exclude records after grouping. This is the most important difference and if we remember this, it will help we write better SQL queries. I am sure we have used WHERE clause because its one of the most common clauses in SQL along with SELECT and used to specify filtering criterion or condition. We can even use WHERE clause without HAVING or GROUP BY, as we have seen it many. times  
  
On the other hand, **HAVING can only be used if grouping has been performed using the GROUP BY clause in the SQL query**. Another worth noting thing about WHERE and HAVING clause is that **WHERE clause cannot contain aggregate function like COUNT(), SUM(), MAX(), MIN(), etc but  HAVING clause may contain aggregate functions.**  
  
Another worth noting the difference between WHERE and HAVING clause is that **WHERE is used to impose filtering criterion on a SELECT, UPDATE, DELETE statement as well as single row function** and **used before group by** clause but **HAVING is always used after group by clause.**  
Difference between WHERE vs HAVING in SQL

1) **WHERE clause is processed right after FROM clause** in the logical order of query processing, which means it is processed before GROUP BY clause while **HAVING clause is executed after groups are created.**   
2) **If used in GROUP BY, We can refer any column from a table** in WHERE clause but we can only use columns which are not grouped or aggregated.  
3) If we use HAVING clause without group by, it can also refer any column but the index will not be used as opposed to WHERE clause. For example, the following have the same result set, however "where" will use the id index and having will do a table scan  
  
select \* from table where id = 1  
select \* from table having id = 1  
  
4) We can use an aggregate function to filter rows with HAVING clause. Because HAVING clause is processed after the rows have been grouped, we can refer to an aggregate function in the logical expression. For example, the following query will display only courses which have more than 10 students :  
  
SELECT Course, COUNT(Course) as NumOfStudent from Training **GROUP BY Course** **HAVING** COUNT(COURSE)> 10  
  
5) Another key difference between WHERE and HAVING clause is that WHERE will use Index and HAVING will not, for example following two queries will produce an identical result but WHERE will use Index and HAVING will do a table scan  
  
SELECT \* FROM Course WHERE Id = 101;  
SELECT \* FROM Course HAVING Id = 102;  
  
6) Since WHERE clause is evaluated before groups are formed, it evaluates for per row. On the other hand, the HAVING clause is evaluated after groups are formed hence it evaluates for per group.



**Q. When to use WHERE and HAVING clause?**

**Ans**. Though both are used to exclude rows from the result set, we should use WHERE clause to filter rows before grouping and use the HAVING clause to filter rows after grouping. In other words, WHERE can be used to filter on table columns while HAVING can be used to filter on aggregate function like count, sum, avg, min, and max.  
  
If filtering can be done without aggregate function then we must do it on WHERE clause because it improves performance because counting and sorting will be done on a much smaller set. If we filter same rows after grouping, we unnecessarily bear the cost of sorting, which is not used.  
For example, the following statement is syntactically correct and produce the same result, but the second one is more efficient than the first one because it filters rows before grouping :  
  
SELECT Job, City, State, Count(Employee) from ... HAVING...  
SELECT ..... from .. WHERE ...  
  
Just remember that WHERE is used to filter rows before grouping while HAVING is used to filter rows after grouping. We can also use AGGREGATE function along with HAVING clause for filtering.

**Q.What is difference between View and Materialized View in Database or SQL?**

**Ans**.

Views allow a level of separation than an original table in terms of access rights but it always fetches updated data.

**View in database**

Views are a logical virtual table created by “select query” but the result is not stored anywhere in the disk and every time we need to fire the query when we need data, so always we get updated or latest data from original tables.  
  
The performance of the view depends on our select query. If we want to improve the performance of view we should avoid using join statements in our query or if we need multiple joins between table always try to use the index-based column for joining as we know index-based columns are faster than a non-index based column.View also allows storing the definition of the query in the database itself.

**Materialized View in database**

Materialized views are also the logical view of our data-driven by the select query but the result of the query will get stored in the table or disk, also the definition of the query will also store in the database.  
  
When we see the performance of Materialized view it is better than normal View because the data of materialized view will be stored in table and table may be indexed so faster for joining also joining is done at the time of materialized views refresh time so no need to every time fire join statement as in case of view.  
  
Difference between View vs Materialized View in database

1) The first difference between View and materialized view is that In Views query result is not stored in the disk or database but Materialized view allow to store the query result in disk or table.  
2) Another difference between View vs materialized view is that, when we create a view using any table, rowid of view is same as the original table but in case of Materialized view rowid is different. See Oracle PL/SQL Programming: Covers Versions Through Oracle Database 12c for more details on materialized view in Oracle.  
  
3) One more difference between View and materialized view in the database is that In case of View we always get latest data but in case of Materialized view we need to refresh the view for getting latest data.  
4) Performance of View is less than Materialized view.  
5) This is continuation of first difference between View and Materialized View, In case of view its only the logical view of table no separate copy of table but in case of Materialized view we get physically separate copy of table  
6) Last difference between View vs Materialized View is that In case of Materialized view we need an extra trigger or some automatic method so that we can keep MV refreshed, this is not required for views in the database.

When to Use View vs Materialized View in SQL

Mostly in an application, we use views because they are more feasible, only logical representation of table data no extra space needed.  
 We easily get a replica of data and we can perform our operation on that data without affecting actual table data but when we see a performance which is crucial for a large application they use materialized view where Query Response time matters.  
  
So, Materialized views are used mostly with data warehousing or business intelligence application.

**Q. Difference between Primary key vs Candidate Key in table - SQL database**

**Ans**. By definition primary key is a column or collection of columns, which uniquely defines a row in a table. Candidate keys are keys that can be a primary key and also able to uniquely identify any row in the table.  
In simple terms, we may have a couple of Candidate keys and we have chosen one of them as a primary key.  
This selection part is the most important skill in database design. Since only the primary key can have a clustered index in a table while unique keys can have a Nonclustered index, it's important to choose the right column or collection of columns as a primary key.  
Often I select a column which is most frequently used in Where clause of SELECT query

Similarities between Candidate Key vs Primary Key

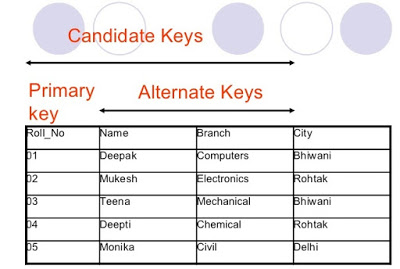
 1) Both Primary and Candidate keys can uniquely identify records in a table on the database.

 2) Both Primary and Candidate keys have constraints UNIQUE and NOT NULL.

 3) Primary key or Candidate keys can be either single column or combination of multiple columns in a table.  
  
Difference between a Candidate key and primary key in SQL

1) There can be multiple Candidate keys in a table in relation to database e.g. Oracle, MySQL, Sybase, or MSSQL but only one primary key is permitted.

2) An example of Primary key and Candidate key can be ID and SSN number in an Employee table, Since both can identify each employee uniquely they are candidate key and anyone can become the primary key.  
  
Now if we have to choose between them as primary key, I will go ID as a primary key because SSN is sensitive information and may not be allowed/not safe to use as String in queries as frequently as ID.  
The second reason of choosing ID over SSN as a primary key can be the use of ID as a primary tracking ID within the organization and its frequent use all over the place.  
Once we choose a primary key, All candidate keys are like unique keys.



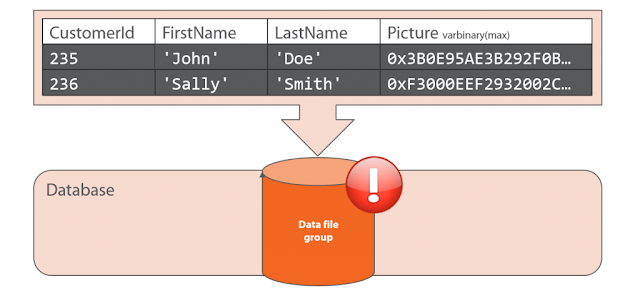
In this table Roll\_No, Name, and Branch are candidate keys because they have unique values but in reality, Name can be duplicate and Branch can also become duplicate, but Roll\_No will be unique, hence it is chosen as a primary key.

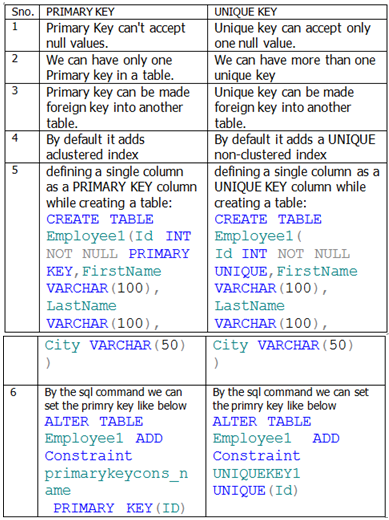
**Q. What is difference between primary key and unique key in table - ?**

**Ans**. The primary key and unique key are two important concepts in a relational database and used to uniquely identify a row in a table.   
 The key thing to remember is that they help to keep data integrity in check. The unique key ensures that no duplicate values are inserted in the database, while a primary and foreign key is responsible for maintaining referential integrity.  
 They also used to link two tables when we are joining them. Btw, there is a slight difference in unique key and unique key constraint, former is a concept while later is an implementation that ensures that no duplicate key is inserted into our tables.

Difference between primary key and unique key in SQL

1) Unique key in a table can be null, at-least one but the primary key cannot be null in any table in a relational database like MySQL, Oracle etc.  
2) The primary key can be a combination of more than one unique keys in the same table.  
3) There can be only one primary key per table in relation database e.g. MySQL, Oracle or Sybase but there can be more than one unique key per table.  
4) The unique key is represented using unique constraint while a primary key is created using primary key constraint in any table and it's automatically got unique constraint.



5) Many database engines automatically put clustered index on the primary key and since we can only have one clustered index per table, it's not available to any other unique key at same time. This is true for both Oracle and MySQL Server.   


**Q. Difference between Self and Equi Join in SQL - INNER Join example MySQL**

**Ans**. The main difference between Self Join and Equi Join is that In Self Join we join one table to itself rather than joining two tables. Both Self Join and Equi Join are types of INNER Join in SQL, but there is a subtle difference between the two. Any INNER Join with equal as join predicate is known as Equi Join. SQL Joins are the fundamental concept of SQL similar to correlated and noncorrelated subqueries or using group by clause and a good understanding of various types of SQL join is the must for any programmer.  
 By the way, If we have written INNER join using where clause than using a comparison operator as = will be known as an equijoin. Equi joins or Self-join is not a formal JOIN or part of the syntax, instead, they are a just popular way to refer certain join examples.

One of the best examples of Self Join, I have seen in an SQL query Interview questions is "How do we find all Employees who are Managers in Employee table", which is commonly asked along with another popular question how to find the second highest salary of employee or questions related to joining three tables in one SQL query.

Self Join Example in SQL

In order to solve this query let's first see schema and data of the Employee table.

mysql> select \* from employee;  
+--------+----------+---------+--------+--------+  
| emp\_id | emp\_name | dept\_id | salary | mgr\_id |  
+--------+----------+---------+--------+--------+  
|    103 | Jack     |       2 |   1400 |    104 |  
|    104 | John     |       2 |   1450 |    104 |  
|    105 | Johnny   |       3 |   1050 |    104 |  
|    108 | Alan     |       3 |   1150 |    104 |  
|    106 | Virat    |       4 |    850 |    105 |  
|    107 | Vina     |       4 |    700 |    105 |  
|    109 | joya     |       4 |    700 |    105 |  
+--------+----------+---------+--------+--------+  
7 rows in set (0.00 sec)

In above  table all employees who are managers has there emp\_id as mgr\_id in other employees and by using SELF JOIN  i.e. join two instances of employee table and comparing, we can find all employees who are managers. Here is the SELECT query example using self-join:

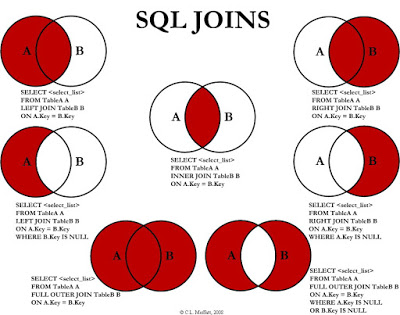
Mysql-

select distinct e.emp\_id, e.emp\_name from employee e

join employee m on e.emp\_id=m.mgr\_id;  
+--------+----------+  
| emp\_id | emp\_name |  
+--------+----------+  
|    104 | John     |  
|    105 | Johnny   |  
+--------+----------+  
2 rows in set (0.00 sec)

In this example of self join, we have joined employee table to itself by using two table aliases e and m. We have also used distinct keyword to remove duplicates here. We can also say this is an example of EQUI JOIN because in join predicate we have used = or equal condition. In fact, this one is an example of INNER Join, SELF Join, and EQUI Join at the same time.

If we are confused about syntax of all possible join in SQL, here is a handy diagram:



**Self Join vs Equi Join**

In the short major difference between Self Join and Equi Join in SQL is that Self Join requires only one table while most of Equi join is condition used in join predicate. Since Equi Join is based on the condition for comparison, it can occur in any INNER, OUTER or SELF join in SQL.

**Q. How to use ALTER COMMAND in SQL ?**

**Ans** . ALTER command is mainly **used to add, modify and drop columns, indexes, and constraints on the table in relational database** e.g. MySQL, Oracle, Sybase and SQL Server.  Though ALTER is not part of classical CRUD operation but it’s one of the important **DDL commands**. One of the most frequent uses of ALTER command in SQL is adding and removing indexes to improve the performance of SQL SELECT queries.  
  
Example 1 : How to add column in an existing table using ALTER clause in SQL:

Suppose we have one table called student with field roll\_number, stu\_name, class, subject and we want to add one more column called marks-obtained, how will do using ALTER Command.

**ALTER TABLE Student ADD marks\_obtained Number (3);**

When this field is added to the table it will contain NULL Value by default.

Example 2 :  How to modify existing column using ALTER command in SQL:

Suppose now we want to modify above added field mark\_obtained from number to decimal data type we can use modify column along with alter command.

**ALTER TABLE Student ALTER COLUMN marks\_obtained DECIMAL (5, 2);**

Example 3 :  How to drop existing column from table using ALTER command in SQL:

Now if we want to drop some existing column from the table we use drop along with ALTER command

**ALTER TABLE Student DROP column marks\_obtained;**

Example 4. How to drop Primary Key Constraint using ALTER command in database :

Suppose there is a PRIMARY KEY CONSTRAINT named 'pk\_roll\_num' for the column roll\_number of the Student table and we want to drop this constraint

**ALTER TABLE Student DROP CONSTRAINT 'pk\_roll\_num’;**

Example 5. How to add primary key constraints using ALTER command in SQL:

Now if we want to add primary key constraint in table Student named 'pk\_roll\_num' for the column roll\_number' . We do like this

**ALTER TABLE Student ADD Constraint 'pk\_roll\_num PRIMARY KEY(roll\_number)**

Example 6. How to drop foreign key Constraint using Alter command:

Suppose there is a foreign key constraint named 'fk\_roll\_num' for the column roll\_number' of the 'Marks' table and we want to drop this constraint

**ALTER TABLE Marks DROP CONSTRAINT 'fk\_roll\_num’;**

Example 7 : How to add foreign key Constraints using alter command

Now if we want to add foreign key constraint in table Marks named 'fk\_roll\_num' for the column roll\_number' .we do like this

**ALTER TABLE Marks ADD Constraint 'fk\_roll\_num FOREIGN KEY (roll\_number) REFERENCES Students (roll\_number);**

Example 8 : How to add unique key Constraints using ALTER command

Now if we want to add Unique key constraint in table Student named ‘unique\_roll\_no ' for the column roll\_number' .we do like this

**ALTER TABLE Student ADD CONSTRAINT unique\_roll\_no UNIQUE (roll\_Number);**

Example 9 : How to drop unique Key Constraint using ALTER command

Now if we want to drop unique key constraint in table Student named ‘unique\_roll\_no ' for the column roll\_number' .we do like this

**ALTER TABLE Student DROP CONSTRAINT unique\_roll\_no);**

Example 10 : How to add check Constraint using ALTER command in SQL

Basic syntax for adding check constraint in table is like this

**ALTER TABLE table\_name ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);**

**Q. Difference between table scan, index scan, and index seek in SQL Server Database**

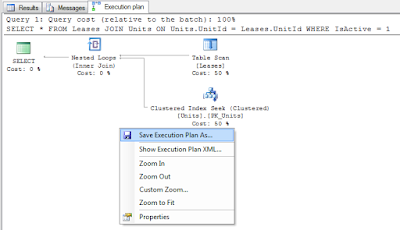
**Ans**. Hello guys, a good understanding of how the index works and how to use them to improve our SQL query performance is very important while working in a database and SQL and that's why we will find many questions based upon indexes on Programming Job interviews. One of such frequently asked SQL question is the real difference between table scan, index scan, an index seek? which one is faster and why? How does the database chooses which scan or seek to use? and How we can optimize the performance of our  SQL SELECT queries by using this knowledge. In general, there are only two ways in which our query engine retrieves the data, using a table scan or by using an index.  
Which method is used for our particular query depends upon what indexes are available in that table, what columns we are requesting in our query, the kind of joins we are doing, and the size of our tables.  
If we have a clear understanding of how the index works and how SQL engine retrieves data from the disk then we can quickly identify performance problems and solve them. That's where most of the SQL developers, especially the Java application developer who write queries and design database lack.  
Btw, if we are not familiar with what is an index and how to create and drop an index, then I suggest we first go through The Complete SQL Bootcamp Become an expert at SQL course to learn those basics.

**Q. Difference between table scan, index scan, and index seek**

**Ans**. In this article, we'll go through each three i.e. table scan, index scan, and index seek and try to understand how database process a particular query hence a basic understanding of database, SQL and index are required.

1. What is a Table Scan in a database?

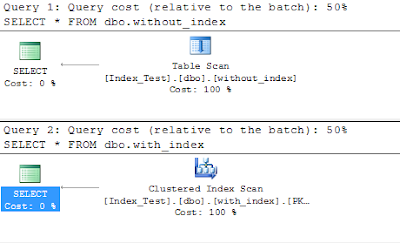
A table scan is a pretty straightforward process. When our query engine performs a table scan it starts from the physical beginning of the table and goes through every row in the table. If a row matches the criterion then it includes that into the result set.  
  
We might have heard nasty things about table scans but in truth, it's the fastest way to retrieve data especially if our table is quite small. It starts being bad when our table starts growing. We can imagine doing a full table scan in a table with 4 million rows and a full table scan in a table with just 100 rows.  
  
In a small table, a query engine can load all data in just one shot but in a large table, it's not possible, which means more IO and more time to process those data.  
  
Normally, a full table scan is used when our query doesn't have a WHERE clause, I mean, we want more or less every record from a table like the following query will use a full table scan:  
  
SELECT \* from Employee;  
  
Btw, If our query is taking too long in a large table then most likely it using either table scan or index scan. We can see that by enabling an execution plan like by doing Ctrl + A in Microsoft SQL Server Management Studio. If we are new to SQL Server, I highly recommend we to join the Microsoft SQL for Beginners to learn T-SQL and SQL Server Management Studio better.

[](https://click.linksynergy.com/deeplink?id=JVFxdTr9V80&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Fmicrosoft-sql-for-beginners%2F)

2. What is the Index Scan in a database?

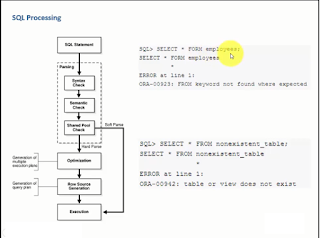
If our table has a clustered index and we are firing a query which needs all or most of the rows i.e. query without WHERE or HAVING clause, then it uses an index scan. It works similar to the table scan, during the query optimization process, the query optimizer takes a look at the available index and chooses the best one, based on information provided in our joins and where clause, along with the statistical information database keeps.  
  
Once the right index is chosen, the SQL Query processor or engine navigates the tree structure to the point of data that matches our criteria and again extracts only the records it needs. See SQL Performance Explained by Markus Winand to learn more about how indexes work in different databases.

The main difference between a full table scan and an index scan is that because data is sorted in the index tree, the query engine knows when it has reached the end of the current it is looking for. It can then send the query, or move on to the next range of data as necessary.  
  
For example, the following query, same as above will use Index scan if we have a clustered index in our table:  
  
SELECT \* From Employee;  
  
This is slightly faster than the table scan but considerably slower than an index seek which we'll see in the next section.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fadvanced-sql-server-performance-tuning%2F)

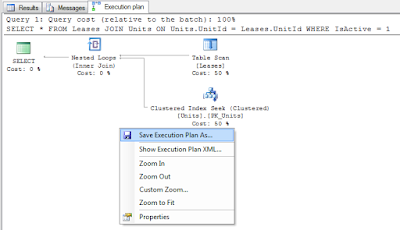
3. What is Index Seek in SQL?

When our search criterion matches an index well enough that the index can navigate directly to a particular point in our data, that's called an index seek. It is the fastest way to retrieve data in a database. The index seeks are also a great sign that our indexes are being properly used.  
  
This happens when we specify a condition in WHERE clause like searching an employee by id or name if we have a respective index.  
  
For example, the following query will use an index seek, we can also confirm that by checking the execution plan of this query when we run this on SQL server:  
  
SELECT \* from Employee where EmployeeId=3;  
  
In this case, the Query Optimizer can use an index to directly go to the third employee and retrieve the data. If we look at the execution plan shown below, we can see that it uses an index seek using the index created on EmployeeId.  
  
If we want to learn more about how the SQL engine process requests I suggest we go through the SQL Tuning course by Amarnath Reddy from Udemy/ No we will not only learn more about SQL performance tuning but also learn SQL performance tuning tools which can help we with the task. I highly recommend this course to anyone who wants to improve their SQL and Database performance skills

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fsql-tuning%2F)

**Q. Difference between table scan, index scan, and index seek in SQL**

Based upon our understanding of indexes, we can now deduce the following points to summarize the difference between table scan, index scan, and index seek in a database:  
1) A table scan and an index scan are used when we need to retrieve all data like 90% to 100% while index seek is used when we need to retrieve data based upon some conditions like 10% of data.  
2) If our query doesn't have WHERE clause and our table doesn't have a clustered index then a full table scan is used, if it does have a clustered index then index scan is used.  
3) index scan is faster than a table scan because they look at sorted data and query optimizer know when to stop and look for another range.  
4) index seek is the fastest way to retrieve data and it comes into picture when our search criterion is very specific. Normally, when we have WHERE clause in our query and we are using a column which also has an index, then index seek is used to retrieve data as shown in the following query:  
  
select \* from Employee where Id= 3;  
  
We can confirm that by actually looking at the execution plan for our query.  
  
In MSSQL management studio, we can see the execution plan by clicking Ctrl + A and then running our query. If we want to learn more SQL Server Performance Tuning Part 1 by Raphel Ashgar is a good starting point.

[](https://click.linksynergy.com/deeplink?id=JVFxdTr9V80&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Fsql-server-performance-tuning-part-1%2F)

That's all about the difference between table scan, index scan and index seek in a database. As I told, there are only two ways to retrieve data in a database either by using a table scan or by using an index. The later is faster in case of large tables. The choice of the index depends upon multiple things like the WHERE clause and joins in our table, the columns we are requesting, the size of tables, etc.  
  
If we feel that our query is slow, we must check the execution plan to confirm whether its using index seeks or index scan or table scan. Then we can optimize our query by introducing the right index or tunning our query.

**Q. Difference between ISNULL() and COALESCE() function in SQL?**

**Ans**. Even though both ISNULL() and COALESCE() function provides alternate values to NULL in T-SQL and Microsoft SQL Server e.g. replacing NULL values with empty String, there are some key differences between them, which is often the topic of SQL Server interview. In this article, we will not only learn the answer to this question but also learn how to use COALESCE and ISNULL function properly. One the main difference between them is that  COALESCE() is a standard SQL function but ISNULL() is a Microsoft SQL Server specific, which means it's not guaranteed to be supported by other database vendors like Oracle, MySQL or PostgreSQL.  
  
But, perhaps the most important difference between them is that COALESCE is more flexible and powerful than ISNULL().  
  
With ISNULL(), we can only provide one alternate value but with COALESCE we can provide more than one e.g. if col1 IS NULL then take value from column2, if that is NULL then take the default value.  
  
Btw, that's not the only differences, there are three key differences between ISNULL() and COALESCE() which we will explore in this article. Btw, if we are new to SQL Server, then I first suggest we go through a good introductory course like Microsoft SQL for Beginners on Udemy. That will help we to better prepare for the interview as well instead of just going through random articles.

ISNULL() vs COALESCE()

There are three major differences between these two function besides being ANSI standard or not:  
  
1) COALESCE correctly promotes its arguments to the highest data type in the expression list, but ISNULL doesn't.  
  
2) COALESCE is more flexible and allows we to provide multiple columns and default values but ISNULL can only work with two values.  
  
3) In the case of ISNULL, the alternate value takes the length of the first value but this doesn't happen in the case of COALESCE. Which means the type of COALESCE expression is determined by the returned element whereas the return type of ISNULL is determined by the first input.  
  
4) When we use them in a SELECT INTO query then both will produce a NON-NULL value in result table if the attribute has NON NULL constraint but if it doesn't then COALESCE will create an attribute which allows NULL and ISNULL will create which doesn't allow NULLs  
  
Now let's understand each point in little more detail.

1.  COALESCE promotes its argument to the higher data type.

As I have said before that, COALESCE correctly promotes its arguments to the highest data type in the expression list, while ISNULL just looks the data type of the first argument and makes everything of that type. Let's see an SQL query to understand this point:

SELECT 19 / ISNULL(CONVERT(INT,NULL), 3.00);

Output

6

SELECT 19 / COALESCE(CONVERT(INT,NULL), 3.00)

Output

6.333333

In first SQL query, we have used ISNULL and first data type is INT but because of its NULL, it also converts 3.00 to INT and performed integer arithmetic, but COALESCE correctly promotes the 19 to FLOAT and performed floating point arithmetic. See Microsoft SQL for Beginners to learn more about floating point arithmetic and data types in SQL Server.

2. COALESCE allows multiple values but ISNULL allows only one value

We can provide COALESCE multiple values to use in case target is NULL. For example, in the following query, we have provided four options to COALESCE

DECLARE @x VARCHAR(10)

DECLARE @y VARCHAR(10)

DECLARE @z VARCHAR(10)

DECLARE @a VARCHAR(10)

SELECT @a = 'SQL'

--This will return SQL

SELECT COALESCE(@x,@y,@z,@a)

Output

SQL

With ISNULL, we can only provide two values e.g.

SELECT ISNULL(@x,@y); --NULL

SELECT ISNULL(@x,@a); --SQL

This flexibility allows we to replace complex case statement with simple coalesce function call on SQL Server stored procedure and functions.  See Querying Microsoft SQL Server 2012  to learn more about it.

3. Length of Result

In the case of COALESCE data type of the result value determines the type of COALESCE expression but in the case of ISNULL, it's the type of the first argument. For example, see the following T-SQL Query:

DECLARE

@a AS VARCHAR(4) = NULL,

@b AS VARCHAR(10) = '1234567890';

SELECT COALESCE(@a, @b) AS [COALESCE], ISNULL(@a, @b) AS [ISNULL];

Output

COALESCE ISNULL

1234567890 1234

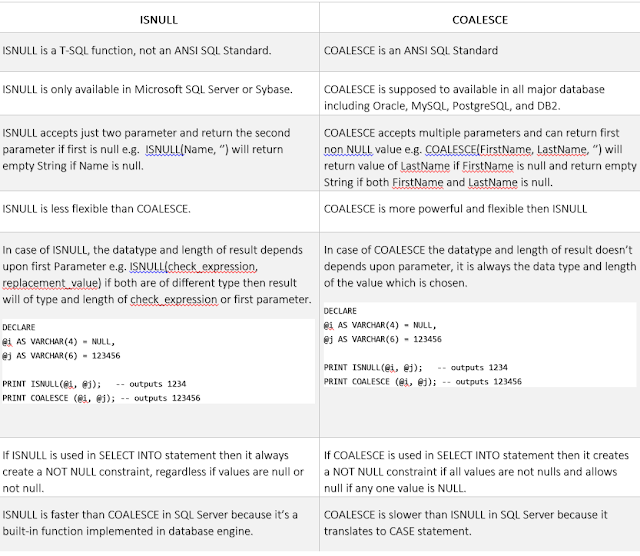
We can see that in the case of COALESCE() the result has type and length VARCHAR(10) but in the case of ISNULL() is the length of first value i.e. length is 4 character. Another worth noting thing is the use of square bracket e.g. [ISNULL], we do this when we use any keyword or function as literal i.e. variable name or column name.  If we want to learn more, I suggest we read Microsoft SQL Server 2012 T-SQL Fundamentals, one of the best books to understand SQL Server.

4. The behavior of COALESCE and ISNULL when used in SELECT INTO

One more difference between COALESCE and ISNULL comes when we are using them in SELECT INTO clause. If we don't know we can create a table by copying data and schema from another table by using SELECT INTO clause.  
  
If we are using something like:  
  COALESCE(column1, 0) as new\_column  
vs  
ISNULL(column1, 0) as new\_column.  
  
Then, both expression will produce a NOT NULL attribute in result table if the source is defined as NOT NULL, but in case source attribute allows NULLs then COALESCE will create an attribute which allows NULL and ISNULL will create which doesn't allow NULLs.  
  
Btw, if we are preparing for Microsoft SQL Server certification like Microsoft SQL Server 2012/2014 Certification Training Exam 70-461 then also this concept will help we a lot. There is sure shot chance that we will get at least one question from this concept.

And if we really want to do well then we can also take a look at this list of free 70-461: Querying Microsoft SQL Server Practice questions and exam dumps from David Mayer.

Finally, here is a nice summary of the differences between COALESCE and ISNULL function in Microsoft SQL Server.

[](https://javarevisited.blogspot.com/2018/05/top-5-sql-and-database-courses-to-learn-online.html)

That's all about the difference between ISNULL and COALESCE in SQL Server. Generally, it's recommended to stick to standard features unless there is some flexibility or major performance we get by using a non-standard feature.  
  
Since ISNULL is actually more limited than COALESCE, so there is no reason to use ISNULL over COALESCE, unless we find ISNULL more readable than COALESCE, like many beginners.  
  
Btw, we must remember these key differences between ISNULL and COALESCE if we are refactoring code and replacing ISNULL with COALESCE in our SQL Script.

**Q. What is difference between Clustered and Non-Clustered Indexes in SQL - Database**

**Clustered vs Nonclustered Indexes in SQL**  
**Ans.** The difference between Clustered and Nonclustered index in a relational database is one of the most popular SQL interview questions almost as popular as the difference between truncate and delete,  primary key vs unique key and correlated vs noncorrelated subqueries. Indexes are a very important concept, it makes our queries run fast and if we compare a SELECT query which uses an indexed column to one who doesn't we will see a big difference in performance. There can be two kinds of indexes in relational database Clustered and Nonclustered indexes. A clustered index determines the physical sorting order of rows in a table similar to entries on yellow pages which are sorted in alphabetical order.  
  
Suppose we have a table Employee, which contains emp\_id as primary key than a clustered index which is created on a primary key will sort the Employee table as per emp\_id. That was a brief introduction of What is clustered index in SQL.  
  
On another hand, the Non-Clustered index involves one extra step which points to the physical location of the record. In this SQL Interview question, we will see some more differences between clustered and nonclustered index in point format.  
  
Btw, if we are very new to SQL and doesn't even know what is an index, what is the real use of an index in a table, and how to create and drop index then we should first go through The Complete SQL Bootcamp Become an expert at SQL! course from Udemy. It's one of the best courses to learn SQL fundamentals in a quick time.  
  
Most of the things we discuss here will make more sense if we have a basic understanding of what is n index and how they work.

Difference between Clustered and Nonclustered Indexes in SQL

In the last section, we have seen What is Clustered and Nonclustered index and How they resolve the physical location of the record. Let's see some more difference between them to answer this SQL interview question :  
1) A Clustered Index physically sort all rows while Nonclustered Index doesn't.  
2) In SQL, one table can only have one Clustered Index but there is no such restriction on the NonClustered Index.  
3) In many relational databases, the Clustered Index is automatically created on the primary key column.  
4) Clustered Index can improve the performance of data retrieval while non-clustered index should be created on columns that are used in the join, where, and order by clause. We can further see the SQL Performance Explained by Markus Winand to learn more about how to index work in different databases e.g. MySQL, SQL Server, and Oracle.

That's all on the difference between Clustered vs Nonclustered Index in SQL. An index is very critical to query performance but too many indexes can actually harm the performance of INSERT and UPDATE queries because these require rearrangement of indexes which can be time-consuming based on the size of table or number of records on the table.  
  
Always create Indexes on columns that are frequently used in a SELECT query because there is a huge performance difference between an indexed and non-indexed column in SQL queries, though we should be careful with creating too many indexes because they will make the update and delete statement slow.  
  
An index is maintained on separate places and costs to update the index tree is directly proportional to the number of indexes we have. If we have index in all columns of the database then obviously, we have to re-arrange most of those index trees whenever we update or delete a row.  
  
Anyway, here are some key differences between a clustered index and a non-clustered index in the tabular format:

