Q: What is the difference between inner and outer join? Explain with example.

Inner join

Inner join is the most common type of Join which is used to combine the rows from two tables and create a result set containing only such records that are present in both the tables based on the joining condition (predicate).

Inner join returns rows when there is at least one match in both tables

If none of the record matches between two tables, then INNER JOIN will return a NULL set. Below is an example of INNER JOIN and the resulting set.

Outer Join

Outer Join can be full outer or single outer

Outer Join, on the other hand, will return matching rows from both tables as well as any unmatched rows from one or both the tables (based on whether it is single outer or full outer join respectively).

Notice in our record set that there is no employee in the department 5 (Logistics). Because of this if we perform inner join, then Department 5 does not appear in the above result. However in the below query we perform an outer join (dept left outer join emp), and we can see this department.

Q: What is the difference between JOIN and UNION?

SQL JOIN allows us to “lookup” records on other table based on the given conditions between two tables. For example, if we have the department ID of each employee, then we can use this department ID of the employee table to join with the department ID of department table to lookup department names.

UNION operation allows us to add 2 similar data sets to create resulting data set that contains all the data from the source data sets. Union does not require any condition for joining. For example, if you have 2 employee tables with same structure, you can UNION them to create one result set that will contain all the employees from both of the tables.

SELECT \* FROM EMP1

UNION

SELECT \* FROM EMP2;

Q: What is the difference between UNION and UNION ALL?

UNION removes duplicate records (where all columns in the results are the same), UNION ALL does not.

There is a performance hit when using UNION vs UNION ALL, since the database server must do additional work to remove the duplicate rows, but usually you do not want the duplicates (especially when developing reports).

UNION Example:

SELECT 'foo' AS bar UNION SELECT 'foo' AS bar

Result:

+-----+

| bar |

+-----+

| foo |

+-----+

1 row in set (0.00 sec)

UNION ALL example:

SELECT 'foo' AS bar UNION ALL SELECT 'foo' AS bar

Result:

+-----+

| bar |

+-----+

| foo |

| foo |

+-----+

2 rows in set (0.00 sec)

Q: What is the difference between WHERE clause and HAVING clause?

WHERE clause introduces a condition on individual rows; HAVING clause introduces a condition on aggregations, i.e. results of selection where a single result, such as count, average, min, max, or sum, has been produced from multiple rows. Your query calls for a second kind of condition (i.e. a condition on an aggregation) hence HAVING works correctly.

As a rule of thumb, use WHERE before GROUP BY and HAVING after GROUP BY. It is a rather primitive rule, but it is useful in more than 90% of the cases.

While you're at it, you may want to re-write your query using ANSI version of the join:

SELECT L.LectID,Fname,Lname

FROM Lecturers L

JOIN Lecturers\_Specialization S ON L.LectID=S.LectID

GROUP BY L.LectID,Fname,Lname

HAVING COUNT(S.Expertise)>=ALL

(SELECT COUNT(Expertise) FROM Lecturers\_Specialization GROUP BY LectID)

This would eliminate WHERE that was used as a theta join condition.

Q: What is the difference among UNION, MINUS and INTERSECT?

UNION combines the results from 2 tables and eliminates duplicate records from the result set.

MINUS operator when used between 2 tables, gives us all the rows from the first table except the rows which are present in the second table.

INTERSECT operator returns us only the matching or common rows between 2 result sets.

To understand these operators, let’s see some examples. We will use two different queries to extract data from our emp table and then we will perform UNION, MINUS and INTERSECT operations on these two sets of data.

UNION

SELECT \* FROM EMPLOYEE WHERE ID = 5

UNION

SELECT \* FROM EMPLOYEE WHERE ID = 6

ID MGR\_ID DEPT\_ID NAME SAL DOJ

5 2 2.0 Anno 80.0 01-Feb-2012

6 2 2.0 Darl 80.0 11-Feb-2012

MINUS

SELECT \* FROM EMPLOYEE

MINUS

SELECT \* FROM EMPLOYEE WHERE ID > 2

ID MGR\_ID DEPT\_ID NAME SAL DOJ

1 2 Hash 100.0 01-Jan-2012

2 1 2 Robo 100.0 01-Jan-2012

INTERSECT

SELECT \* FROM EMPLOYEE WHERE ID IN (2, 3, 5)

INTERSECT

SELECT \* FROM EMPLOYEE WHERE ID IN (1, 2, 4, 5)

ID MGR\_ID DEPT\_ID NAME SAL DOJ

5 2 2 Anno 80.0 01-Feb-2012

2 1 2 Robo 100.0 01-Jan-2012

Q: How to generate row number in SQL Without ROWNUM

Generating a row number – that is a running sequence of numbers for each row is not easy using plain SQL. In fact, the method I am going to show below is not very generic either. This method only works if there is at least one unique column in the table. This method will also work if there is no single unique column, but collection of columns that is unique. Anyway, here is the query:

SELECT name, sal, (SELECT COUNT(\*) FROM EMPLOYEE i WHERE o.name >= i.name) row\_num

FROM EMPLOYEE o

order by row\_num

Q: How to select first 5 records from a table?

This question, often asked in many interviews, does not make any sense to me. The problem here is how do you define which record is first and which is second. Which record is retrieved first from the database is not deterministic. It depends on many uncontrollable factors such as how database works at that moment of execution etc. So the question should really be – “how to select any 5 records from the table?” But whatever it is, here is the solution:

In Oracle,

SELECT \*

FROM EMP

WHERE ROWNUM <= 5;

In SQL Server,

SELECT TOP 5 \* FROM EMP;

Generic solution,

I believe a generic solution can be devised for this problem if and only if there exists at least one distinct column in the table. For example, in our EMP table ID is distinct. We can use that distinct column in the below way to come up with a generic solution of this question that does not require database specific functions such as ROWNUM, TOP etc.

SELECT name

FROM EMPLOYEE o

WHERE (SELECT count(\*) FROM EMPLOYEE i WHERE i.name < o.name) < 5

Q: What is the difference between ROWNUM pseudo column and ROW\_NUMBER() function?

ROWNUM is a pseudo column present in Oracle database returned result set prior to ORDER BY being evaluated. So ORDER BY ROWNUM does not work.

ROW\_NUMBER() is an analytical function which is used in conjunction to OVER() clause wherein we can specify ORDER BY and also PARTITION BY columns.

Suppose if you want to generate the row numbers in the order of ascending employee salaries for example, ROWNUM will not work. But you may use ROW\_NUMBER() OVER() like shown below:

SELECT name, sal, row\_number() over(order by sal desc) rownum\_by\_sal

FROM EMPLOYEE o

Q: What are the differences among ROWNUM, RANK and DENSE\_RANK?

ROW\_NUMBER assigns contiguous, unique numbers from 1.. N to a result set.

RANK does not assign unique numbers—nor does it assign contiguous numbers. If two records tie for second place, no record will be assigned the 3rd rank as no one came in third, according to RANK. See below:

SELECT name, sal, rank() over(order by sal desc) rank\_by\_sal

FROM EMPLOYEE o

name Sal RANK\_BY\_SAL

Hash 100 1

Robo 100 1

Anno 80 3

Darl 80 3

Tomiti 70 5

Pete 70 5

Bhuti 60 7

Meme 60 7

Inno 50 9

Privy 50 9

DENSE\_RANK, like RANK, does not assign unique numbers, but it does assign contiguous numbers. Even though two records tied for second place, there is a third-place record. See below:

SELECT name, sal, dense\_rank() over(order by sal desc) dense\_rank\_by\_sal

FROM EMPLOYEE o

name Sal DENSE\_RANK\_BY\_SAL

Hash 100 1

Robo 100 1

Anno 80 2

Darl 80 2

Tomiti 70 3

Pete 70 3

Bhuti 60 4

Meme 60 4

Inno 50 5

Privy 50 5

Q: Clustered Indexes

Clustered indexes are indexes that uniquely identify the rows in an SQL table.

Every table can have exactly one clustered index.

You can create a clustered index that covers more than one column. For example: create Index index\_name(col1, col2, col.....).

By default, a column with a primary key already has a clustered index.

Non-clustered Indexes

Non-clustered indexes are like simple indexes. They are just used for fast retrieval of data. Not sure to have unique data.

Q: What is the difference between a HAVING CLAUSE and a WHERE CLAUSE?

They specify a search condition for a group or an aggregate. But the difference is that HAVING can be used only with the SELECT statement. HAVING is typically used in a GROUP BY clause. When GROUP BY is not used, HAVING behaves like a WHERE clause. Having Clause is basically used only with the GROUP BY function in a query whereas WHERE Clause is applied to each row before they are part of the GROUP BY function in a query.

Q: What is Stored Procedure?

A stored procedure is a named group of SQL statements that have been previously created and stored in the server database. Stored procedures accept input parameters so that a single procedure can be used over the network by several clients using different input data. And when the procedure is modified, all clients automatically get the new version. Stored procedures reduce network traffic and improve performance. Stored procedures can be used to help ensure the integrity of the database.

Q: What are DMVs?

Dynamic Management Views (DMVs), are functions that give you information on the state of the server. DMVs, for the most part, are used to monitor the health of a server. They really just give you a snapshot of what’s going on inside the server. They let you monitor the health of a server instance, troubleshoot major problems and tune the server to increase performance.

Q: Define a temp table

In a nutshell, a temp table is a temporary storage structure. What does that mean? Basically, you can use a temp table to store data temporarily so you can manipulate and change it before it reaches its destination format.

Q: What’s the difference between a local temp table and a global temp table?

Local tables are accessible to a current user connected to the server. These tables disappear once the user has disconnected from the server. Global temp tables, on the other hand, are available to all users regardless of the connection. These tables stay active until all the global connections are closed.

Q: How do you use transactions?

In general, there are three types of transactions that you can use in the SQL Server environment: BEGIN TRANSACTION, ROLL BACK TRANSACTION and COMMIT TRANSACTION. The gist behind deploying transactions is that they allow you to group multiple SQL commands into a single unit. From there, each transaction begins with a certain task, and ends when all the tasks within the transaction are complete. BEGIN TRANSACTION gets the ball rolling. ROLLBACK TRANSACTION functions a lot like an “undo” command, and COMMIT TRANSACTION completes all of the tasks within that transaction.

Q: What’s the difference between a clustered and a non-clustered index?

A clustered index directly affects the way tabled data is stored on a specific disk. This means that when a clustered index is used, data is stored in sequential rows based on the index column value. This is why a table can only contain a single clustered index. Non-clustered indexes directly affect the way physical data is stored and managed within SQL Server.

Q: What are DBCC commands?

In very basic terms the Database Consistency Checker (DBCC) is used to aid in server maintenance. DBCC commands, many of which are completely undocumented, provide a set of commands that let you perform routing maintenance, status and validation checks. The most common DBCC commands are: DBCC CHECKALLOC (Lets you check disk allocation); DBCC OPENTRAN (Lets you check any open transactions); and DBCC HELP (shows a list of available DBCC commands to aid your server maintenance processes).

Q: Describe the difference between truncate and delete

The difference between these two processes is fairly simple. Truncate means to simply empty out a table. On the other hand, the delete command lets you delete entire rows from within a table, but not all of the data within that table.

Q: What is a view?

A view is simply a virtual table that is made up of elements of multiple physical or “real” tables. Views are most commonly used to join multiple tables together, or control access to any tables existing in background server processes.

Q: What is a Query Execution Plan?

SQL Server has several built-in tools that optimize how queries are executed within their databases. A query execution plan is exactly what it sounds like – a snapshot of how the optimizing tools will execute and deploy specific queries within the database. This service helps you troubleshoot problems with jobs that don’t necessarily execute perfectly.

Q: What is the default port number for SQL Server?

While this is kind of a softball question – if you know anything about SQL Server you should at least know the basic configuration options – it’s an important one to nail in the interview. Basically, when SQL Server is enabled the server instant listens to the TCP port 1433.

Q: What is Cursor?

Cursor is a database object used by applications to manipulate data in a set on a row-by- row basis, instead of the typical SQL commands that operate on all the rows in the set at one time.

In order to work with a cursor we need to perform some steps in the following order:

Declare cursor

Open cursor

Fetch row from the cursor

Process fetched row

Close cursor

Deallocate cursor

http://a4academics.com/interview-questions/53-database-and-sql/397-top-100-database-sql-interview-questions-and-answers-examples-queries?showall=&limitstart=