SQL SERVER Interview Questions & Answers - SET 2 (40 Questions)

http://msbiskills.com/

Question1. We have a query which is running fine in development but facing performance issues at production. You got the execution plan from production DBA. Now you need to compare with the development execution plan. What is your approach? / Poor query performance in Prod.

Answer –

This is an open ended question; there could be possible reasons for this issue. Some of them are given below-

1. Fragmentation could be one of the issues.
2. Check statistics are updated or not.
3. Check what other processes are running on production server.
4. Check if query is returning multiple query plans; if yes then there could be two reasons – Parameter sniffing or invalid plans due to set options
5. Check which indexes are getting used?
6. Examine the execution plan to find out the Red Flags. You have to understand what is going on bad and then figure out the alternatives.

Question2. What is Lock Escalation?

Answer –

Lock escalation is the process of converting many fine-grain locks into fewer coarse-grain locks, reducing system overhead while increasing the probability of concurrency contention. Every lock takes some memory space – 128 bytes on 64 bit OS and 64 bytes on 32 bit OS. And memory is not the free resource.

So if we have a table with billions of rows and we doing lot of operations on that table, SQL Server starts to use the process that called "Lock Escalation" i.e. instead of keeping locks on every row SQL Server tries to escalate them to the higher (object) level. As soon as you have more than 5.000 locks on one level in your locking hierarchy, SQL Server escalates these many fine-granularity locks into a simple coarse-granularity lock.

By default SQL Server always escalates to the table level. You can control lock escalation using below code.

Note lock goes from top to bottom (DB -> Table -> Page -> Row)

```
ALTER TABLE Area
SET
(
LOCK_ESCALATION = AUTO -- or TABLE or DISABLE
)
GO
```

Notes – You can disable Lock Escalation for time being, but you got to be very careful with this. You can use loop to perform DELETE/UPDATE statements, so that you can prevent Lock Escalations. With this approach huge transactions will be divided into multiple smaller ones, which will also help you with Auto Growth issues that you maybe have with your transaction log.

Question3. What is a Race Condition?

Answer –

A race condition is when two or more programs (or independent parts of a single program) all try to acquire some resource at the same time, resulting in an incorrect answer or conflict.

Question4. Difference between ISNULL and COAELSCE

Answer –

ISNULL	COALESCE
Is Null can accept 2 parameters only.	It can accept multiple parameters. Minimum parameters should be 2 in this case.
Data type here returned by the function is the data type of the first parameter.	Data type returned is the expression with the highest data type precedence. If all expressions are non-nullable, the result is typed as non-nullable.
It is a built in function	Internally Coalesce will be converted to case.
If both the values are NULL it will return NULL (of Data Type INT)	Here one of the values should be NON NULL otherwise It will throw an error. At least one of the arguments to COALESCE must be an expression that is not the NULL constant. We can do something like below-DECLARE @d AS INT = NULL SELECT COALESCE(NULL, @d)
In the below case NULL will be returned. DECLARE @d AS INT = NULL SELECT ISNULL(NULL, @d)	In the below case NULL will be returned. DECLARE @d AS INT = NULL SELECT COALESCE(NULL, @d)

Question5. What is the difference between CROSS APPLY & OUTER APPLY IN T-SQL?

Answer –

The APPLY operator comes in two flavors, CROSS APPLY and OUTER APPLY. It is useful for joining two SQL tables or XML expressions.

CROSS APPLY is equivalent to an INNER JOIN expression and OUTER APPLY is equivalent to a LEFT OUTER JOIN expression. E.g. below-

```
CREATE TABLE EmpApply
(
     EmpId INT
    ,EmpName VARCHAR(100)
    ,DeptID INT
)
GO

INSERT INTO EmpApply VALUES
(1,'Rohit',1),
(2,'Rahul',2),
(3,'Isha',1),
(4,'Disha',NULL)

CREATE TABLE DeptApply
(
     DeptID INT
    ,Name VARCHAR(100)
)
GO

INSERT INTO DeptApply VALUES
(1,'IT'),
(2,'Finance')

SELECT * FROM EmpApply
CROSS APPLY ( SELECT * FROM DeptApply WHERE DeptApply.DeptID = EmpApply.DeptID )ax
```

Output of the above query is

EmpId	EmpName	DeptID	DeptID	Name
1	Rohit	1	1	IT
2	Rahul	2	2	Finance
3	Isha	1	1	IT

```
SELECT * FROM EmpApply
OUTER APPLY ( SELECT * FROM DeptApply WHERE DeptApply.DeptID = EmpApply.DeptID )
ax
```

Output of the above query is –

EmpId	EmpName	DeptID	DeptID	Name
1	Rohit	1	1	IT

2	Rahul	2	2	Finance
3	Isha	1	1	IT
4	Disha	NULL	NULL	NULL

Question6. What is stuff function? Difference between stuff and replace?

Answer-

REPLACE – Replaces all occurrences of a specified string value with another string value.

```
-----------------Syntax -------------------

REPLACE ( string_expression , string_pattern , string_replacement )

-----------------Example--------------------

DECLARE @Text1 AS VARCHAR(100) = 'Pawan - A SQL Dev'
SELECT REPLACE(@Text1,'SQL','MSBI')
```

Output of the above query is

(No column name)
Pawan – A MSBI Dev

STUFF function is used to overwrite existing characters.

```
-----------------Syntax -------------------
STUFF ( character_expression , start , length , replaceWith_expression )

-----------------Example--------------------

DECLARE @Text AS VARCHAR(100) = 'Pawan - A SQL Dev'
SELECT STUFF(@Text,2,5,'NEW')
```

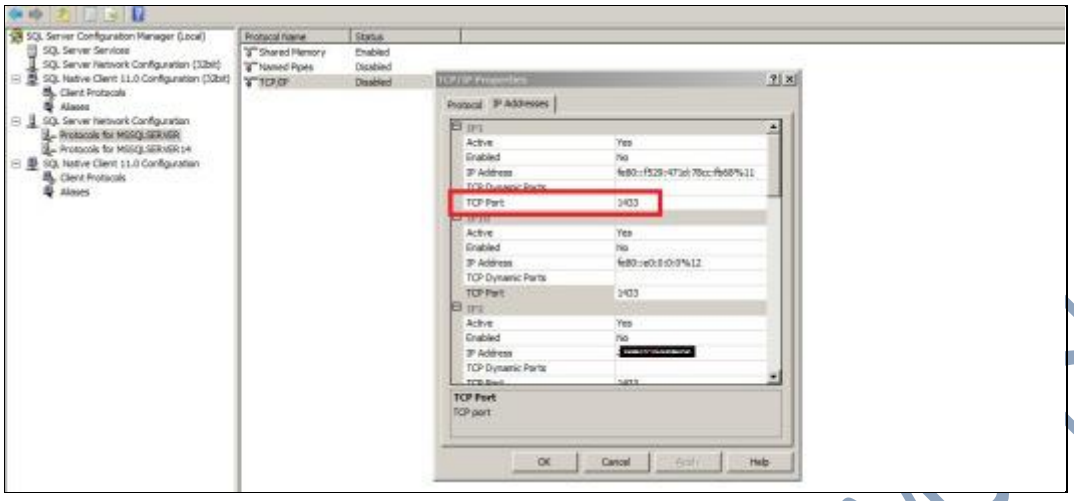
Output of the above query is

(No column name)
PNEW- A SQL Dev

Question7. How to change the port number for SQL Server? Default port no of SQL SERVER

Answer-

Default PORT Number of SQL Server is 1433. You can view the port number under configuration Manager.



URL – https://msdn.microsoft.com/en-us/library/ms177440.aspx

To assign a TCP/IP port number to the SQL Server Database Engine

In SQL Server Configuration Manager, in the console pane, expand **SQL Server Network Configuration**, expand **Protocols for**, and then double-click **TCP/IP**.

1. In the **TCP/IP Properties** dialog box, on the **IP Addresses** tab, several IP addresses appear in the format **IP1**, **IP2**, up to **IPAll**. One of these is for the IP address of the loopback adapter, 127.0.0.1. Additional IP addresses appear for each IP Address on the computer. Right-click each address, and then click **Properties** to identify the IP address that you want to configure.
2. If the **TCP Dynamic Ports** dialog box contains **0**, indicating the Database Engine is listening on dynamic ports, delete the 0.
3. In the **IP*n* Properties** area box, in the **TCP Port** box, type the port number you want this IP address to listen on, and then click **OK**.
4. In the console pane, click **SQL Server Services**.
5. In the details pane, right-click **SQL Server ()** and then click **Restart**, to stop and restart SQL Server.

After you have configured SQL Server to listen on a specific port, there are three ways to connect to a specific port with a client application:

- Run the SQL Server Browser service on the server to connect to the Database Engine instance by name.
- Create an alias on the client, specifying the port number.
- Program the client to connect using a custom connection string.

Question8. What is memory grant in sql server?

Answer-

Query memory grant OR Query Work Buffer is a part of server memory used to store temporary row data while sorting and joining rows.

It is called “grant” because the server requires those queries to “reserve” before actually using memory. This reservation improves query reliability under server load, because a query with reserved memory is less likely to hit out-of-memory while running, and the server prevents one query from dominating entire server memory. For details please visit below

http://blogs.msdn.com/b/sqlqueryprocessing/archive/2010/02/16/understanding-sql-server-memory-grant.aspx

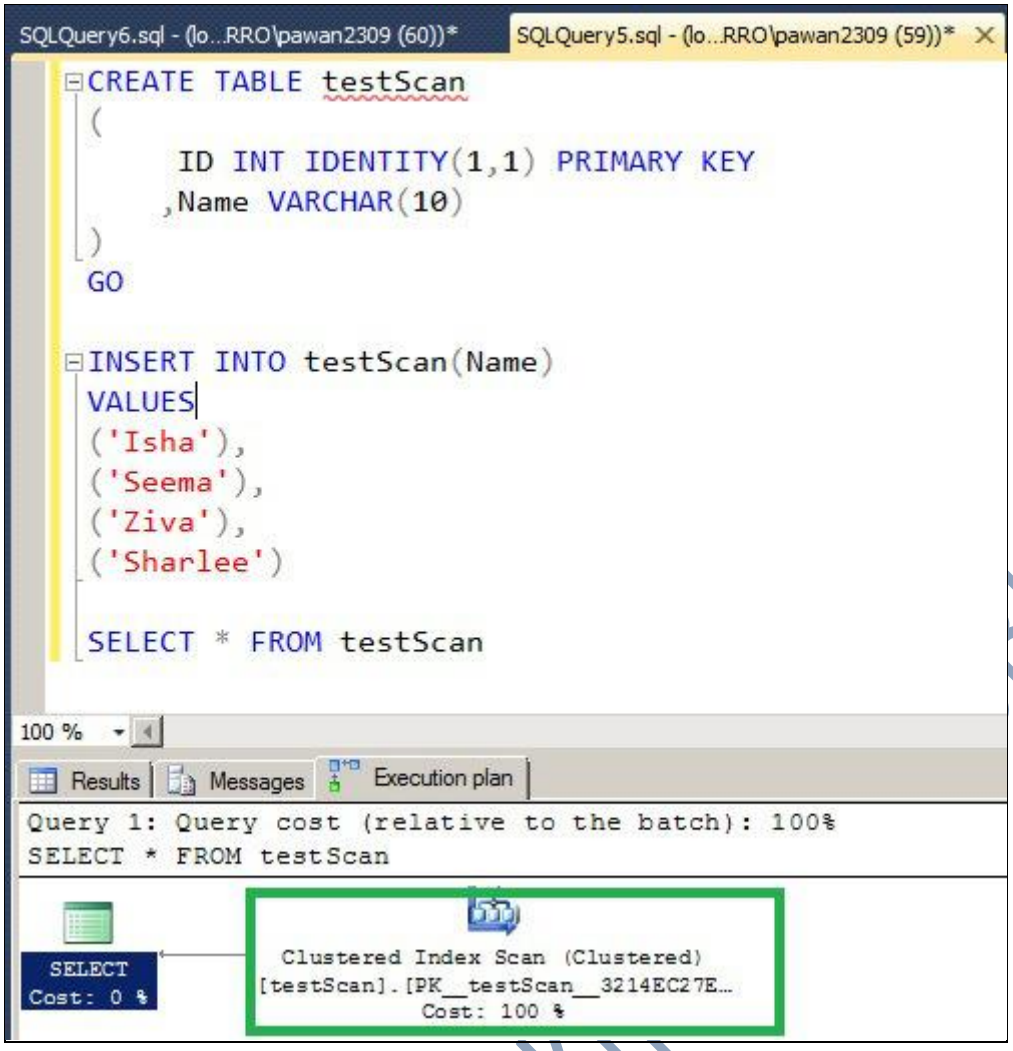
Question9. When index scan happens?

Answer -

An index scan is when SQL Server has to scan all the index pages to find the appropriate records. Please check out the example below

```
CREATE TABLE testScan
(
     ID INT IDENTITY(1,1) PRIMARY KEY
    ,Name VARCHAR(10)
)
GO

INSERT INTO testScan(Name)
VALUES
('Isha'),
('Seema'),
('Ziva'),
('Sharlee')

SELECT * FROM testScan
```

Check out the execution plan of the above query



Question10. How to prevent bad parameter sniffing? What exactly it means?

Answer –

Parameter sniffing is an expected behavior. SQL Server compiles the stored procedures using the parameters send to the first time the procedure is compiled and save it in plan cache for further reuse.

After that every time the procedure executed again, Now the SQL Server retrieves the execution plan from the cache and uses it.

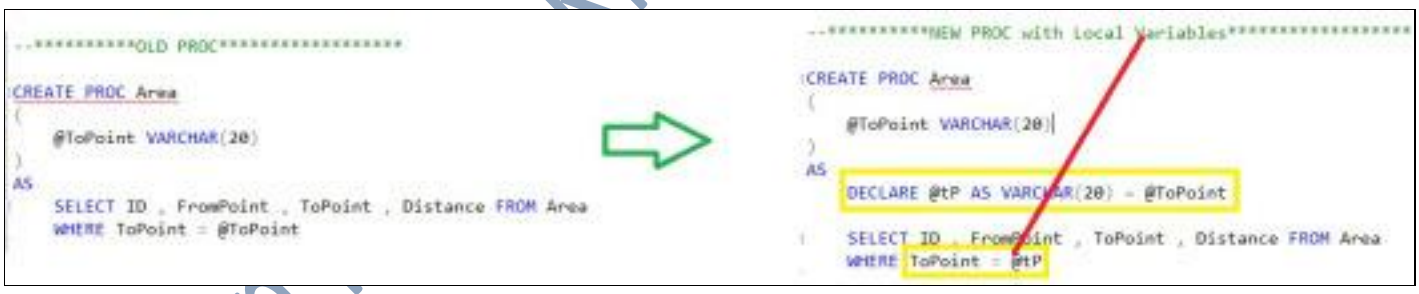
The potential problem arises when the first time the stored procedure is executed, the set of parameters generate an acceptable plan for that set of parameters but very bad for other more common sets of parameters.

Workarounds to overcome this problem are given below

- **OPTION (RECOMPILE)**
- **OPTION (OPTIMIZE FOR (@VARIABLE=VALUE))**
- **OPTION (OPTIMIZE FOR (@VARIABLE UNKNOWN))**
- **Use local variables**

I have explained how we can overcome this using local variable.

```
--**********OLD PROC*****************

CREATE PROC Area
(
    @ToPoint VARCHAR(20)
)
AS
    SELECT ID , FromPoint , ToPoint , Distance FROM Area
    WHERE ToPoint = @ToPoint



--**********NEW PROC with Local Variables*****************


CREATE PROC Area
(
    @ToPoint VARCHAR(20)
)
AS
    DECLARE @tP AS VARCHAR(20) = @ToPoint

    SELECT ID , FromPoint , ToPoint , Distance FROM Area
    WHERE ToPoint = @tP
```



Question11. While creating non clustered indexes on what basis we should choose main columns and include columns?

Answer –

A NonClustered index can be extended by including nonkey columns in addition to the index key columns. The nonkey columns are stored at the leaf level of the index b-tree.

The Syntax of a Non Clustered Index with Included column is given below

```
CREATE INDEX <Index_Name> ON <table> (KeyColumns) INCLUDE (NonKeyColumns)
```

- **KeyColumns** – These columns are used for row restriction and processing E.g they were used in WHERE, JOIN, ORDER BY, GROUP BY etc.
- **NonKeyColumns** – These columns are used in SELECT and Aggregation. For e.g. AVG(col) after selection/restriction.

So always choose KeyColumns and NonKeyColumns based on the query requirements only.

Question12. What is the difference between pessimistic locking and optimistic locking?

Answer –

Source – http://dba.stackexchange.com/questions/35812/why-is-optimistic-locking-faster-than-pessimistic-locking

Let's start with the analogy with banks.

Pessimistic locking is like having a guard at the bank door who checks your account number when you try to enter; if someone else accessing your account, then you cannot enter until that other person finishes his/her transaction and leaves.

Pessimistic Locking is when you lock the record for your exclusive use until you have finished with it. It has much better integrity than optimistic locking but requires you to be careful with your application design to avoid Deadlocks.

Optimistic locking, on the other hand, allows you to walk into the bank at any time and try to do your business, but at the risk that as you are walking out the door the bank guard will tell you that your transaction conflicted with someone else's and you will have to go back and do the transaction again.

Optimistic Locking is a strategy where you read a record, take note of a version number and check that the version hasn't changed before you write the record back. When you write the record back you filter the update on the version to make sure it's atomic. (i.e. hasn't been updated between when you check the version and write the record to the disk) and update the version in one hit.

If the record is dirty (i.e. different version to yours), Optimistic locking possibly causes a transaction to fail, but it does so without any "lock" ever having been taken. And if a transaction fails because of optimistic locking, the user is required to start all over again. The word "optimistic" derives from exactly the expectation that the condition that causes transactions to fail for this very reason, will occur only very exceptionally. "Optimistic" locking is the approach that says "I will not be taking actual locks because I hope they won't be needed anyway. If it turns out I was wrong about that, I will accept the inevitable failure.".

This strategy is most applicable to high-volume systems and three-tier architectures where you do not necessarily maintain a connection to the database for your session. In this situation the client cannot actually maintain database locks as the connections are taken from a pool and you may not be using the same connection from one access to the next.

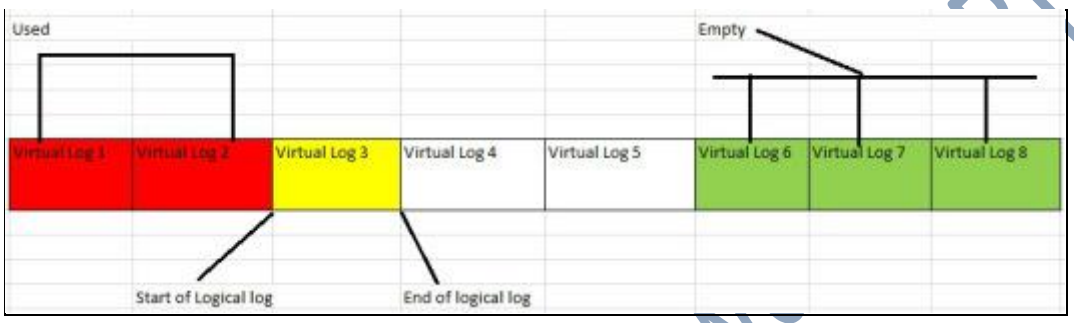
Question13. How VLF's created for tempDB?

Answer – VLFs are Virtual Log Files.

A transaction log stores every transaction made to a SQL Server database, except some which are minimally logged like BULK IMPORT or SELECT INTO.

Internally transaction log is split into the smaller parts called Virtual Log Files (VLFs). When one VLF becomes full, logging continue to write into the next available in the transaction log. The transaction log file can be represented as a circular file. When the logging reaches the end of the file it starts again from the beginning, but only if all the requirements has been met and the inactive parts has been truncated.

The truncation process is necessary to mark all inactive parts so they can be used again and overwritten.



Every time space is allocated for the transaction log file (It may be an Initial creation or log growth) new VLFs are created behind the scenes. The number of new VLFs is determined by the amount of space allocated.

- **If space added is between 0 to 64MB then 4 new VLFs**
- **If space Added is between 64MB to 1GB then 8 new VLFs**
- **If space Added is greater than 1GB then 16 new VLFs**

Use below query to find out the growth and transaction log details

```
SELECT
    name FileName,
    CAST(size*8/1024 AS VARCHAR(10))+'MB' Size,
    CASE is_percent_growth
     WHEN 1 THEN CAST(growth AS VARCHAR(10))+'%'
     ELSE CAST(growth*8/1024 AS VARCHAR(10))+'MB'
     END AutoGrowth
FROM sys.database_files WHERE type_desc = 'LOG'

DBCC LOGINFO;
GO
```

Question14. Can you give some examples for One to One, One to Many and Many to Many relationships?

Answer – There are following types of database relationships.

- One to One Relationships
- One to Many and Many to One Relationships
- Many to Many Relationships

One to One Relationships

Let's say we have two table Customers and Address. We have a relationship between the Customers table and the Addresses table. If each address can belong to only one customer, this relationship is "One to One". Check out the example below.

One to Many and Many to One Relationship

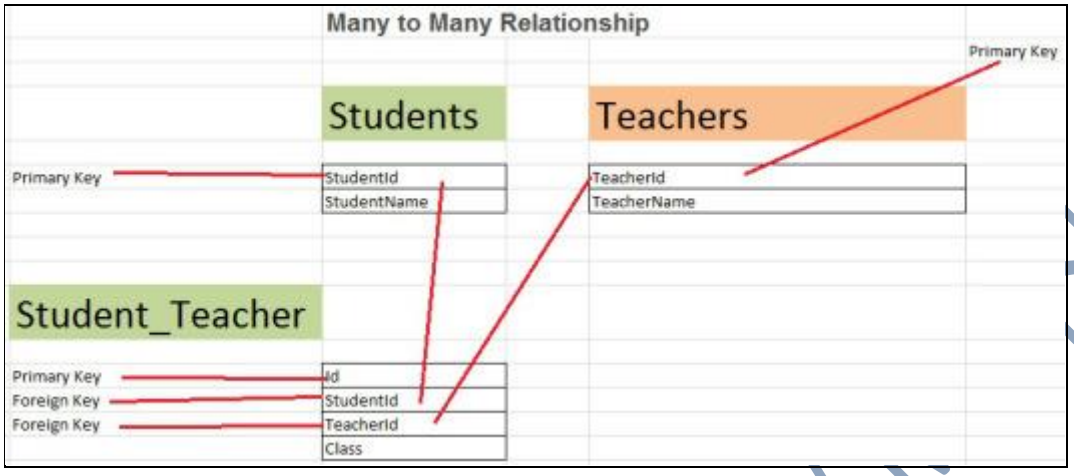
Let's say we have two table Customers and Orders. We have a relationship between the Customers table and the Addresses table. Customers can make multiple orders but we cannot have multiple customer in one order. This kind of relationship called "One to many relationship". Check out the example below.

Many and Many Relationships

Let's say we have two table Students and Teachers. We have a relationship between the Students table and the Teachers table. One teacher can teach multiple students and one student can be taught be multiple teacher. This kind of relationship called "One to many relationship".

Note – Please note that for this kind of relationship we require 3rd table to handle relationship.

Check out the example below.



Question15. How to find all dependent objects of a table?

Answer –

Some methods are there given below-

```
--Note "testP" is the name of the proc

********************** Method 1 ************************************

sp_depends 'testP'

********************** Method 2 ************************************

SELECT * FROM information_schema.routines
WHERE CHARINDEX('testP', ROUTINE_DEFINITION) > 0
GO

********************** Method 3 ************************************

SELECT referencing_schema_name, referencing_entity_name,
referencing_id, referencing_class_desc, is_caller_dependent
FROM sys.dm_sql_referencing_entities ('testP', 'OBJECT');
GO
```

```
--
```

Sp_depends will be deprecated, and instead, sys.dm_sql_referencing_entities and sys.dm_sql_referenced_entities are recommended.

For details please refer below URL

http://www.mssqltips.com/sqlservertip/2999/different-ways-to-find-sql-server-object-dependencies/

Question16. How to filter nested stored procedure code from profiler?

Answer –

Nested stored procedures are the stored procedures that call another stored procedure(s).

We can use SQL Server Profiler to peek into stored procedure execution details. Let's first create nested stored proc.

```
CREATE PROC ChildProc
AS
    SELECT 'ChildProc'


CREATE PROC MainSP
AS
    EXEC ChildProc
```

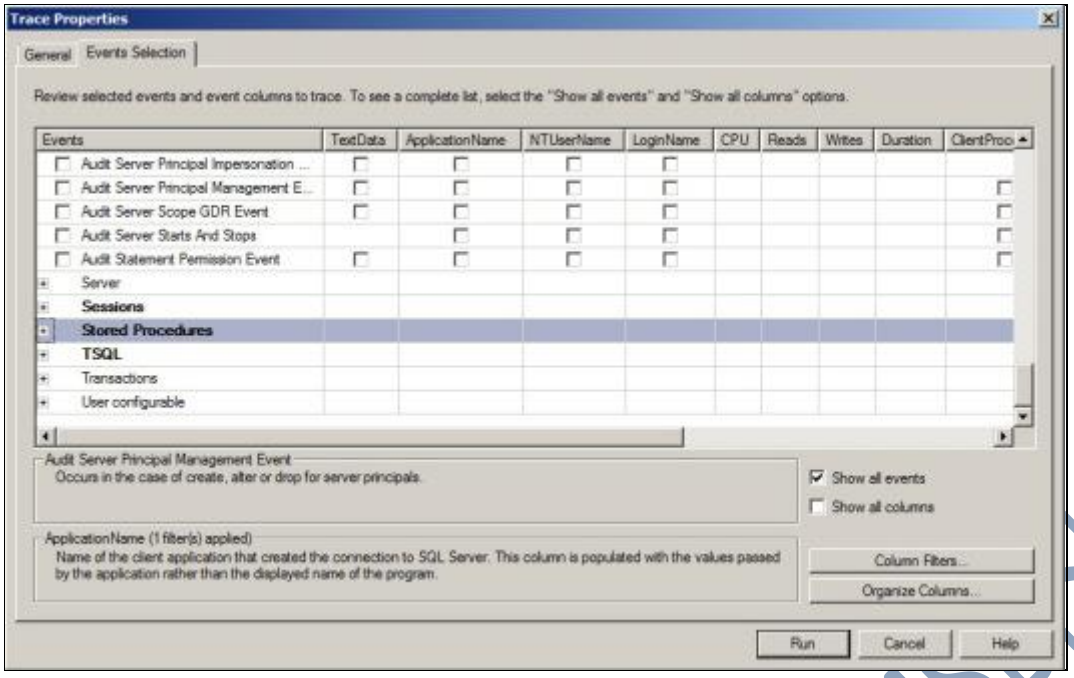
After creating the above stored procedures, execute the parent stored proc using below command.

```
EXEC MainSP
```

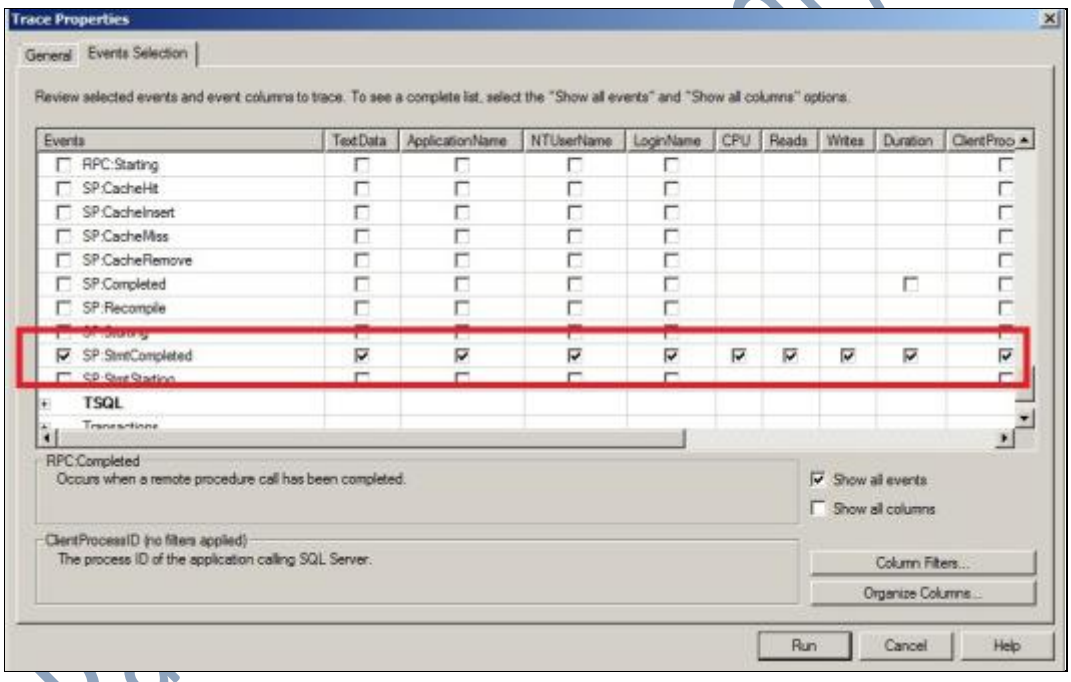
For that please follow the steps given below.

Step 1.Open SQL Server profile by clicking on Start, type SQL SERVER Profiler and click on SQL Server profiler.

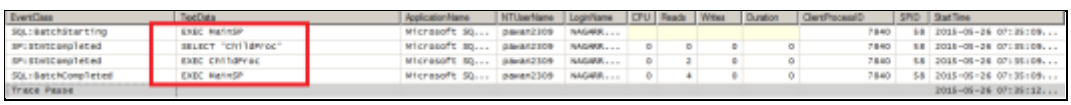
Click on "Show all events", then select stored procedures as shown below.



Select SP: StmtCompleted event class as shown below



Now execute the Main SP and check out the trace below



We can see all the trace i.e. from Parent SP and child SP.

Question17. What are the limitations on “SET ROWCOUNT”?

Answer –

Both TOP and SET ROWCOUNT are both acceptable methods of limiting the result sets from a query; However both are very different commands. The TOP clause of a command limits that single command, while the SET ROWCOUNT command limits all eligible queries within the connection until another SET ROWCOUNT 0 is called.

This could be dangerous sometime if you forget to reset the ROWCOUNT.

Microsoft recommends that we should not use this style as they are planning to stop its affect on DML statements. Microsoft recommends TOP Command.

If both TOP statement and SET ROWCOUNT are used, SET ROWCOUNT over rides TOP when ROWCOUNT value is smaller than TOP value

The below statement will give us only 2 rows as output.

```
SET ROWCOUNT 2
SELECT TOP 4 * FROM Approver
```

SET ROWCOUNT limits all the queries including triggers.

As a part of a SELECT statement, the query optimizer can consider the value of expression in the TOP or FETCH clauses during query optimization. Because SET ROWCOUNT is used outside a statement that executes a query, its value cannot be considered in a query plan.

This setting comes into play during execution and not at parse time.

If you want to reset it, then use the statement below.

```
SET ROWCOUNT 0
```

Note - using SET ROWCOUNT will not affect DELETE, INSERT, and UPDATE statements in a future release of SQL Server.

Avoid using SET ROWCOUNT with DELETE, INSERT, and UPDATE statements in new development work, and plan to modify applications that currently use it. For a similar behavior, use the TOP syntax.

Question18. What is the SQL Query Order of Operations? OR What is the Logical Query Processing Phases and their order in SQL Statement Execution.

Answer –

SQL Server processes SQL statements in a Logical Order. We call it Logical Query Processing Phases. Phases and their order is given below-

FROM
ON
OUTER
WHERE
GROUP BY
CUBE | ROLLUP
HAVING
SELECT
DISTINCT
TOP
ORDER BY

If you want to remember the above sequence use "**Fred-Will-Give-Her-Some-O**"

You can also learn this in detail from below URL-
http://tsql.solidq.com/books/insidetsql2008/Logical%20Query%20Processing%20Poster.pdf

Question19. What is .TUF file? What is the significance of the same? Any implications if the file is deleted?

Answer –

TUF file is the Transaction Undo File.

This file is created when Log Shipping is configured in SQL Server in standby mode. This file is located @ the path where transaction log files were saved.

This File consists of list of uncommitted transactions while backup is going on the primary server in Log Shipping. If .tuf file is got deleted there is no way to repair log shipping except reconfiguring it from scratch.

The transaction undo file contains modifications that were not committed on the source database but were in progress when the transaction log was backed up AND when the log was restored to another database, you left the database in a state that allowed addition transaction log backups to be restored to it (at some point in the future. When another transaction log is restored, SQL Server uses data from the undo file and the transaction log to continue restoring the incomplete transactions (assuming that they are were completed in the next transaction log file). Following the restore, the undo file will be re-written with any transactions that, at that point, are incomplete.

Question20. What is a deadlock and what is a live lock? How will you go about resolving deadlocks?

Answer –

Deadlock is a situation when two processes, each having a lock on one piece of data, attempt to acquire a lock on the other's piece. Each process would wait indefinitely for the other to release the lock, unless one of the user processes is terminated. SQL Server detects deadlocks and terminates one user's process.

A live lock is one, where a request for an exclusive lock is repeatedly denied because a series of overlapping shared locks keeps interfering. SQL Server detects the situation after four denials and refuses further shared locks. A live lock also occurs when read transactions monopolize a table or page, forcing a write transaction to wait indefinitely.

A human example of live lock would be two people who meet face-to-face in a corridor and each move aside to let the other pass, but they end up moving from side to side without making any progress because they always move the same way at the same time and never cross each other. This is good example of live lock.

Question21. Can you tell us different types of Isolation levels? What is the default Isolation level?

Answer –

There are five type of Isolation level in MS SQL Server.

- Read Committed (The Default Isolation Level of MS SQL Server)
- Read Uncommitted
- Repeatable Read
- Serializable
- Snapshot

Question22. How can you move the master database?

Answer – To move master database, you also have to move the Resource database. Microsoft states that the Resource database must reside in the same location as the Master database.

Follow the steps given @

https://msdn.microsoft.com/en-us/library/ms345408.aspx

Question23. When the lazy writer happens and what it'll do?

Answer – The lazy writer is a process that periodically checks the available free space in the buffer cache between two checkpoints and ensures that there is always enough free memory. When the lazy writer determines free pages are needed in the buffer for better performance, it removes the old pages before the regular checkpoint occurs.

Ideally, Lazy Writes should be close to zero. That means that the buffer cache doesn't have to free up dirty pages immediately, it can wait for the automatic check point.

For detailed explanation please click on the below URL-
http://www.sqlshack.com/sql-server-memory-performance-metrics-part-5-understanding-lazy-writes-free-list-stallssec-memory-grants-pending/

Question24. What are the properties of a transaction? What are ACID Properties

Answer – Atomicity Consistency Isolation Durability (ACID) is a concept referring to a database system's four transaction properties: atomicity, consistency, isolation and durability.

Atomicity

Here either all the statements (whether an update, delete or insert) of the transaction will happen or not happen. To guarantee atomicity, SQL Server uses a Write Ahead Transaction Log. The log always gets written to first before the associated data changes. That way, if and when things go wrong, SQL Server will know how to roll back to a state where every transaction happened or didn't happen.

Consistency

A transaction reaching its normal end, thereby committing its results, preserves the consistency of the database. If something bad happens then everything in the transaction will be rolled back. After each transaction DB should be in a consistent state.

Isolation

Events happening within a transaction must be hidden from other transactions running concurrently.

Durability

Once a transaction has been completed and has committed its results to the database, the system must guarantee that these results survive any subsequent malfunctions.

Question25. Why do some system functions require parenthesis and some do not require?

Answer –

In SQL Server most of the system functions require parenthesis at the end. The examples can be GETDATE(), NEWID(), RAND(), ERROR_MESSAGE(), etc.

Some functions do not need the parenthesis. The examples can be CURRENT_TIMESTAMP, CURRENT_USER, etc.

ANSI SQL standard functions do not need parenthesis.

SQL Server functions requires parenthesis

Question26. Why Right Joins does exist?

Answer –

Personally I have never used right join.

According to **Jeremiah Peschka** – Just because we can write our query as a LEFT OUTER JOIN, doesn't mean that you should.

SQL Server provides a RIGHT OUTER JOIN show plan operator (**http://msdn.microsoft.com/en-us/library/ms190390.aspx**).

There are times when it's going to be most efficient to use a right outer join. Leaving that option in the language 1) gives you the same functionality in the language that you have in the optimizer and 2) supports the ANSI SQL specification. There's always a chance, in a sufficiently complex plan on a sufficiently overloaded SQL Server, that SQL Server may time out query compilation.

In theory, if you specify RIGHT OUTER JOIN instead of a LEFT OUTER JOIN, your SQL could provide SQL Server with the hints it needs to create a better plan. If you ever see this situation, though, you should probably blog about it :)

No programming task requires a join, but you can also write all of your queries using syntax like SELECT * FROM a, b, c, d WHERE (a.id = b.a_id OR b.a_id IS NULL) and still have perfectly valid, well-formed, and ANSI compliant SQL.

Question27. Does foreign Key slows down the insertion process in SQL Server?

Answer –

Well the difference would be very negligible. Please read excellent post below for details.

http://www.brentozar.com/archive/2015/05/do-foreign-keys-matter-for-insert-speed/

Question28. Where does SQL Server Agent save jobs?

Answer –

In MSDB, jobs are stored in dbo.sysjobs.

dbo.sysjobsteps – Stores details of the individual steps

dbo.sysjobschedules – Stored schedules of the job

dbo.sysjobhistory – Job history is maintained here.

MSDB also contains other instance level objects such as alerts, operators and SSIS packages.

Question29. How to calculate the likely size of an OLAP cube from the Relational database size?

Answer –

You can use a general rule of Analysis Services data being about 1/4 – 1/3 size of the same data stored in relational database.

Reference – https://social.msdn.microsoft.com/Forums/sqlserver/en-US/6b16d2b2-2913-4714-a21d-07ff91688d11/cube-size-estimation-formula

Question30. SQL Query | Consider the below table below and write some queries to replace 0 by 1 and 1 by 0.

```
CREATE TABLE ZeroOne
(
      Id INT
)
GO

INSERT INTO ZeroOne VALUES (0),(1),(0),(1),(0),(0)
```

Answer –

Some of the options are given below to achieve the expected output.

```
SELECT *, CASE ID WHEN 0 THEN 1 ELSE 0 END Ids FROM ZeroOne

SELECT *, (Id - 1) * -1 Ids FROM ZeroOne

SELECT *, 1 - Id Ids FROM ZeroOne

SELECT *, IIF(ID=0,1,0) Ids FROM ZeroOne

SELECT Id, (Id+1/2 -1) * -1 Ids FROM ZeroOne

DECLARE @t AS INT = 1
SELECT Id, CHOOSE(@t,1,0) Ids FROM ZeroOne
```

Question31. Can you explain sql server transaction log architecture?

Answer –

Please click on the URL for detailed answer.

https://technet.microsoft.com/en-us/library/jj835093(v=sql.110).aspx

Question32. What is a boot page?

Every database has a single page that stores critical information about the database itself. It's always page 9 in file 1 (The first file in the PRIMARY filegroup)

Please click here for detailed explanation - http://www.sqlskills.com/blogs/paul/search-engine-qa-20-boot-pages-and-boot-page-corruption/

You can check out the page using below commands

```
DBCC DBINFO ('Pawan');
```

Output

```
DBINFO STRUCTURE:


DBINFO @0x00000000145FDAE0

dbi_version = 706                   dbi_createVersion = 706             dbi_SEVersion = 0
dbi_dvSplitPoint = 0:0:0 (0x00000000:00000000:0000)
dbi_dbbackupLSN = 15034:5896:37 (0x00003aba:00001708:0025)
```

```
dbi_LastLogBackupTime = 2015-01-22 12:17:21.873
dbi_nextseqnum = 1900-01-01 00:00:00.000                                dbi_status =
0x00010000
dbi_crdate = 2013-04-12 12:15:13.890dbi_dbname = Pawan                  dbi_dbid = 7
dbi_cmptlevel = 110                 dbi_masterfixups = 0                dbi_maxDbTimestamp
= 2000
dbi_dbbackupLSN = 15034:5896:37 (0x00003aba:00001708:0025)              dbi_RebuildLogs =
0
dbi_differentialBaseLSN = 15034:5896:37 (0x00003aba:00001708:0025)      dbi_RestoreFlags =
0x0000
dbi_checkptLSN = 15193:4240:119 (0x00003b59:00001090:0077)              dbi_dbccFlags = 2
dbi_COWLastLSN = 0:0:0 (0x00000000:00000000:0000)
dbi_DirtyPageLSN = 15193:4240:119 (0x00003b59:00001090:0077)            dbi_RecoveryFlags
= 0x00000000
dbi_lastxact = 0x1efd1e             dbi_collation = 61448               dbi_relstat =
0x61000000
dbi_familyGUID = ced081af-d46a-41b0-911c-d42adc6b04ce
dbi_maxLogSpaceUsed = 965496832

dbi_recoveryForkNameStack


entry 0

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
m_guid = ced081af-d46a-41b0-911c-d42adc6b04ce

entry 1

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
m_guid = 00000000-0000-0000-0000-000000000000
dbi_differentialBaseGuid = 555d80f3-3003-419b-a1eb-c1b059f76d22
dbi_firstSysIndexes = 0001:00000014
dbi_oldestBackupXactLSN = 0:0:0 (0x00000000:00000000:0000)
dbi_versionChangeLSN = 0:0:0 (0x00000000:00000000:0000)                 dbi_mdUpgStat =
0x0004
dbi_category = 0x0000000000000000   dbi_safetySequence = 0
dbi_dbMirrorId = 00000000-0000-0000-0000-000000000000
dbi_pageUndoLsn = 0:0:0 (0x00000000:00000000:0000)                      dbi_pageUndoState
= 0
dbi_disabledSequence = 0            dbi_dbmRedoLsn = 0:0:0 (0x00000000:00000000:0000)
dbi_dbmOldestXactLsn = 0:0:0 (0x00000000:00000000:0000)                 dbi_CloneCpuCount
= 0
dbi_CloneMemorySize = 0             dbi_updSysCatalog = 1900-01-01 00:00:00.000
dbi_LogBackupChainOrigin = 15034:5896:37 (0x00003aba:00001708:0025)
dbi_dbccLastKnownGood = 1900-01-01 00:00:00.000                         dbi_roleSequence =
0
dbi_dbmHardenedLsn = 0:0:0 (0x00000000:00000000:0000)                   dbi_localState = 0
dbi_safety = 0                      dbi_modDate = 2013-04-12 12:15:13.890
dbi_verRDB = 184552376              dbi_lazyCommitOption = 0
dbi_svcBrokerGUID = b5fe6d7f-a888-4c3f-af66-3e264784ba17
dbi_svcBrokerOptions = 0x00000000
dbi_dbmLogZeroOutstanding = 0       dbi_dbmLastGoodRoleSequence = 0     dbi_dbmRedoQueue =
0
dbi_dbmRedoQueueType = 0            dbi_rmidRegistryValueDeleted = 0
dbi_dbmConnectionTimeout = 0
dbi_fragmentId = 0                  dbi_AuIdNext = 1099511628233
dbi_MinSkipLsn = 0:0:0 (0x00000000:00000000:0000)
dbi_commitTsOfcheckptLSN = 0
dbi_dbEmptyVersionState = 0         dbi_CurrentGeneration = 0
dbi_EncryptionHistory


Scan 0

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
```

```
EncryptionScanInfo:ScanId = 0

Scan 1

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
EncryptionScanInfo:ScanId = 0

Scan 2

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
EncryptionScanInfo:ScanId = 0
dbi_latestVersioningUpgradeLSN = 18:81:67 (0x00000012:00000051:0043)     dbi_splitAGE = 0
dbi_PendingRestoreOutcomesId = 00000000-0000-0000-0000-000000000000
dbi_ContianmentState = 0
DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

```
DBCC TRACEON(3604);
DBCC PAGE(0,1,9,3);

DBCC execution completed. If DBCC printed error messages, contact your system
administrator.

PAGE: (1:9)


BUFFER:


BUF @0x00000002FB1C0000

bpage = 0x00000002FA5DE000          bhash = 0x0000000000000000          bpageno = (1:9)
bdbid = 7                           breferences = 0                     bcputicks = 0
bsampleCount = 0                    bUse1 = 55356                       bstat = 0x10b
blog = 0x15a9a                      bnext = 0x0000000000000000

PAGE HEADER:


Page @0x00000002FA5DE000

m_pageId = (1:9)                    m_headerVersion = 1                 m_type = 13
m_typeFlagBits = 0x0                m_level = 0                         m_flagBits = 0x0
m_objId (AllocUnitId.idObj) = 99    m_indexId (AllocUnitId.idInd) = 0   Metadata:
AllocUnitId = 6488064
Metadata: PartitionId = 0           Metadata: IndexId = 0               Metadata: ObjectId
= 99
m_prevPage = (0:0)                  m_nextPage = (0:0)                  pminlen = 0
m_slotCnt = 1                       m_freeCnt = 6590                    m_freeData = 1600
m_reservedCnt = 0                   m_lsn = (15193:4928:13)             m_xactReserved = 0
m_xdesId = (0:0)                    m_ghostRecCnt = 0                   m_tornBits = -
1057360342
DB Frag ID = 1

Allocation Status

GAM (1:2) = ALLOCATED               SGAM (1:3) = NOT ALLOCATED
PFS (1:1) = 0x64 MIXED_EXT ALLOCATED 100_PCT_FULL                        DIFF (1:6) =
CHANGED
ML (1:7) = NOT MIN_LOGGED
```

```
Slot 0, Offset 0x60, Length 1504, DumpStyle BYTE

Record Type = PRIMARY_RECORD        Record Attributes =                 Record Size = 1504

Memory Dump @0x00000000145FA060

0000000000000000:   0000e005 c202c202 00000000 00000000 00000000  ..à.Â.Â.............
0000000000000014:   f285ca00 28a40000 00000000 00000000 00000100  ò
Ê.(¤..............
0000000000000028:   755c907b f7efc900 9ea10000 50006100 77006100  u\.{÷ïÉ.¡..P.a.w.a.
000000000000003C:   6e002020 20202020 20202020 20202020 20202020  n.
0000000000000050:   20202020 20202020 20202020 20202020 20202020
0000000000000064:   20202020 20202020 20202020 20202020 20202020
0000000000000078:   20202020 20202020 20202020 20202020 20202020
000000000000008C:   20202020 20202020 20202020 20202020 20202020
00000000000000A0:   20202020 20202020 20202020 20202020 20202020
00000000000000B4:   20202020 20202020 20202020 20202020 20202020
00000000000000C8:   20202020 20202020 20202020 20202020 20202020
00000000000000DC:   20202020 20202020 20202020 20202020 20202020
00000000000000F0:   20202020 20202020 20202020 20202020 20202020
0000000000000104:   20202020 20202020 20202020 20202020 20202020
0000000000000118:   20202020 20202020 20202020 20202020 20202020
000000000000012C:   20202020 20202020 0a000000 07006e00 d0070000      ......n.Ð...
0000000000000140:   00000000 ba3a0000 08170000 25000000 ba3a0000  ....º:......%...º:..
0000000000000154:   08170000 25000000 593b0000 90100000 77000200  ....%...Y;......w...
0000000000000168:   00000000 00000000 00000000 593b0000 90100000  ............Y;......
000000000000017C:   77000000 1efd1e00 00000000 08f00000 00000000  w....ý.......ð......
0000000000000190:   00000061 00000000 af81d0ce 6ad4b041 911cd42a  ...a....¯.ÐÎjÔ°A.Ô*
00000000000001A4:   dc6b04ce 00508c39 00000000 00000000 00000000  Ük.Î.P.9............
00000000000001B8:   00000000 00000000 00000000 00000000 af81d0ce  ................¯.ÐÎ
00000000000001CC:   6ad4b041 911cd42a dc6b04ce 00000000 00000000  jÔ°A.Ô*Ük.Î........
00000000000001E0:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000001F4:   f3805d55 03309b41 a1ebc1b0 59f76d22 14000000  ó.]U.0.A¡ëÁ°Y÷m"....
0000000000000208:   01006302 00000000 00000000 00000000 00000000  ..c.................
000000000000021C:   00000000 00000400 00000000 00000000 00000000  ....................
0000000000000230:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000244:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000258:   00000000 00000000 00000000 00000000 00000000  ....................
000000000000026C:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000280:   ba3a0000 08170000 25000000 00000000 00000000  º:......%...........
0000000000000294:   00000000 00000000 00000000 00000000 f7efc900  ................÷ïÉ.
00000000000002A8:   9ea10000 b80b000b 00000000 7f6dfeb5 88a83f4c  .¡..¸.¸.....mþµ.¨?L
00000000000002BC:   af663e26 4784ba17 00000000 00000000 00000000  ¯f>&G.º.............
00000000000002D0:   00000000 00000000 00000000 00000000 c9010000  ................É...
00000000000002E4:   00010000 00000000 00000000 00000000 00000000  ....................
00000000000002F8:   00000000 00000000 00000000 00000000 00000000  ....................
000000000000030C:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000320:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000334:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000348:   00000000 00000000 00000000 00000000 00000000  ....................
000000000000035C:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000370:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000384:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000398:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000003AC:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000003C0:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000003D4:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000003E8:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000003FC:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000410:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000424:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000438:   00000000 00000000 00000000 00000000 00000000  ....................
000000000000044C:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000460:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000474:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000488:   00000000 00000000 00000000 00000000 00000000  ....................
```

```
000000000000049C:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000004B0:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000004C4:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000004D8:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000004EC:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000500:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000514:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000528:   00000000 00000000 00000000 00000000 00000000  ....................
000000000000053C:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000550:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000564:   00000000 00000000 00000000 00000000 00000000  ....................
0000000000000578:   00000000 00000000 00000000 00000000 00000000  ....................
000000000000058C:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000005A0:   00000000 00000000 00000000 12000000 51000000  ................Q...
00000000000005B4:   43000000 00000000 00000000 00000000 00000000  C...................
00000000000005C8:   00000000 00000000 00000000 00000000 00000000  ....................
00000000000005DC:   00000000                                      ....

DBINFO @0x00000000145FA060

dbi_version = 706                   dbi_createVersion = 706             dbi_SEVersion = 0
dbi_dvSplitPoint = 0:0:0 (0x00000000:00000000:0000)
dbi_dbbackupLSN = 15034:5896:37 (0x00003aba:00001708:0025)
dbi_LastLogBackupTime = 2015-01-22 12:17:21.873
dbi_nextseqnum = 1900-01-01 00:00:00.000                                dbi_status =
0x00010000
dbi_crdate = 2013-04-12 12:15:13.890dbi_dbname = Pawan                  dbi_dbid = 7
dbi_cmptlevel = 110                 dbi_masterfixups = 0                dbi_maxDbTimestamp
= 2000
dbi_dbbackupLSN = 15034:5896:37 (0x00003aba:00001708:0025)               dbi_RebuildLogs =
0
dbi_differentialBaseLSN = 15034:5896:37 (0x00003aba:00001708:0025)       dbi_RestoreFlags =
0x0000
dbi_checkptLSN = 15193:4240:119 (0x00003b59:00001090:0077)               dbi_dbccFlags = 2
dbi_COWLastLSN = 0:0:0 (0x00000000:00000000:0000)
dbi_DirtyPageLSN = 15193:4240:119 (0x00003b59:00001090:0077)             dbi_RecoveryFlags
= 0x00000000
dbi_lastxact = 0x1efd1e             dbi_collation = 61448               dbi_relstat =
0x61000000
dbi_familyGUID = ced081af-d46a-41b0-911c-d42adc6b04ce
dbi_maxLogSpaceUsed = 965496832

dbi_recoveryForkNameStack


entry 0

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
m_guid = ced081af-d46a-41b0-911c-d42adc6b04ce

entry 1

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
m_guid = 00000000-0000-0000-0000-000000000000
dbi_differentialBaseGuid = 555d80f3-3003-419b-a1eb-c1b059f76d22
dbi_firstSysIndexes = 0001:00000014
dbi_oldestBackupXactLSN = 0:0:0 (0x00000000:00000000:0000)
dbi_versionChangeLSN = 0:0:0 (0x00000000:00000000:0000)                  dbi_mdUpgStat =
0x0004
dbi_category = 0x0000000000000000   dbi_safetySequence = 0
dbi_dbMirrorId = 00000000-0000-0000-0000-000000000000
dbi_pageUndoLsn = 0:0:0 (0x00000000:00000000:0000)                       dbi_pageUndoState
= 0
dbi_disabledSequence = 0            dbi_dbmRedoLsn = 0:0:0 (0x00000000:00000000:0000)
dbi_dbmOldestXactLsn = 0:0:0 (0x00000000:00000000:0000)                  dbi_CloneCpuCount
= 0
```

```
dbi_CloneMemorySize = 0             dbi_updSysCatalog = 1900-01-01 00:00:00.000
dbi_LogBackupChainOrigin = 15034:5896:37 (0x00003aba:00001708:0025)
dbi_dbccLastKnownGood = 1900-01-01 00:00:00.000                          dbi_roleSequence =
0
dbi_dbmHardenedLsn = 0:0:0 (0x00000000:00000000:0000)                    dbi_localState = 0
dbi_safety = 0                      dbi_modDate = 2013-04-12 12:15:13.890
dbi_verRDB = 184552376              dbi_lazyCommitOption = 0
dbi_svcBrokerGUID = b5fe6d7f-a888-4c3f-af66-3e264784ba17
dbi_svcBrokerOptions = 0x00000000
dbi_dbmLogZeroOutstanding = 0       dbi_dbmLastGoodRoleSequence = 0     dbi_dbmRedoQueue =
0
dbi_dbmRedoQueueType = 0            dbi_rmidRegistryValueDeleted = 0
dbi_dbmConnectionTimeout = 0
dbi_fragmentId = 0                  dbi_AuIdNext = 1099511628233
dbi_MinSkipLsn = 0:0:0 (0x00000000:00000000:0000)
dbi_commitTsOfcheckptLSN = 0
dbi_dbEmptyVersionState = 0         dbi_CurrentGeneration = 0
dbi_EncryptionHistory


Scan 0

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
EncryptionScanInfo:ScanId = 0

Scan 1

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
EncryptionScanInfo:ScanId = 0

Scan 2

hex (dec) = 0x00000000:00000000:0000 (0:0:0)
EncryptionScanInfo:ScanId = 0
dbi_latestVersioningUpgradeLSN = 18:81:67 (0x00000012:00000051:0043)     dbi_splitAGE = 0
dbi_PendingRestoreOutcomesId = 00000000-0000-0000-0000-000000000000
dbi_ContianmentState = 0


DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

Question33. Where do you write business logic – in the application (as Ad-hoc SQL / in line query) or in the database (Stored Procedures)? Why?

Answer –

Mostly I used to write stored procedures because they are easier for us to test and fine tune. If you want to change the stored procedure in future it is easy. You can just change it and deploy on the server required and test the application. You don't need to open the application, change the query and deploy it and after that you can test it. You can save lot of time in this case.

Stored Procedures don't provide much advantage in security cases, unless restricting access to rows in complex manner. They are better to manage and change in future. Sps are better for complex operations.

So, which one is better to use SPs or ad-hoc SQL? The answer is "it depends."

For details please visit –

https://www.simple-talk.com/sql/t-sql-programming/to-sp-or-not-to-sp-in-sql-server/

Question34. What's the fastest way to insert thousands of records into the database?

Answer-

1. Use BULK INSERT - it is designed for exactly for huge insertions and *significantly* increases the speed of inserts.
2. You can also use Batch inserts. That is, only send 1000 rows at a time, rather than one row at a time, so you hugely reduce round trips/server calls.
3. You can also use SQL BCP utility.

Question35. What's the difference between a primary key and a clustered index?

Sr.No	Clustered Index	Primary Key
1	Clustered index will create only Index on the table. It will not create constraint on the table.	Primary key internally creates 2 objects. They are Index and Primary Key constraint. Both Index and Primary Key constraint can be clustered or non-clustered depending on what you have written in primary key definition. If you don't specify anything then Unique clustered index will be created and a Primary key constraint (clustered) will be created.

2	Here if you create non unique clustered index you can insert multiple null values.
	
	If you create unique clustered index you can insert single null value.
		We cannot insert null values if we have Primary key on the table. You will get below error if you try.
		
		`Msg 515, Level 16, State 2, Line 6`
		`Cannot insert the value NULL into column 'ID', table 'Pawan.dbo.testPrimClus1'; column does not allow nulls. INSERT fails.`
		`The statement has been terminated.`

3.	We can also add Clustered index after table creation using create index command. We don't need to alter table in this case. E.g.
	
	`CREATE TABLE testPrimClus4`
	`(`
	`ID INT`
	`)`
	`GO`
	
	`CREATE UNIQUE CLUSTERED INDEX Ix_Clx ON testPrimClus4(ID)`
		We can add primary key after that creating table using below alter command. Please note that we need to first drop existing constraints. Also primary key column should be non-null. E.g.
		
		`CREATE TABLE testPrimClus113`
		`(`
		`ID INT NOT NULL`
		`)`
		`GO`
		
		`ALTER TABLE testPrimClus113 ADD PRIMARY KEY (ID)`

Question36. What is alzebrizer?

The algebrizer is used resolves all the names of the various objects, tables and columns, referred to within the query string. The algebrizer identifies, at the individual column level, all the data types (varchar (50)) versus datetime and so on) for the objects being accessed. It also determines the location of aggregates (such as GROUP BY, and MAX) within the query, a process called aggregate binding.

This algebrizer process is important because the query may have aliases or synonyms, names that don't exist in the database, that need to be resolved, or the query may refer to objects not in the database. When objects don't exist in the database, SQL Server returns an error from this step, defining the invalid object name.

The algebrizer outputs a binary called the query processor tree, which is then passed on to the query optimizer. The algebrizer's output includes a hash, a coded value representing the query. The optimizer uses the hash to determine whether there is already a plan generated and stored in the plan cache. If there is a plan there, the process stops here and that plan is used. This reduces all the overhead required by the query optimizer to generate a new plan

For details please visit *Grant Fritchey's* book online.

Question37. How to capture the long running queries?

Answer-

```
SELECT TOP 10
      r.session_id
,     r.start_time
,     TotalElapsedTime_ms = r.total_elapsed_time
```

```
,     r.[status]
,     r.command
,     DatabaseName = DB_Name(r.database_id)
,     r.wait_type
,     r.last_wait_type
,     r.wait_resource
,     r.cpu_time
,     r.reads
,     r.writes
,     r.logical_reads
,     t.[text] AS [executing batch]
,     SUBSTRING(
                  t.[text], r.statement_start_offset / 2,
                  (     CASE WHEN r.statement_end_offset = -1 THEN DATALENGTH
(t.[text])
                              ELSE r.statement_end_offset
                        END - r.statement_start_offset ) / 2
               ) AS [executing statement]
,     p.query_plan
FROM
      sys.dm_exec_requests r
CROSS APPLY
      sys.dm_exec_sql_text(r.sql_handle) AS t
CROSS APPLY
      sys.dm_exec_query_plan(r.plan_handle) AS p
ORDER BY
      r.total_elapsed_time DESC;
```

Question38. Do you have any idea about sparse column?

Answer –

Sparse columns are normal columns. They have an optimized storage for null values. Sparse columns reduce the space requirements for null values at the cost of more overhead to retrieve nonnull values. Consider using sparse columns when the space saved is at least 20 percent to 40 percent. E.g.

```
CREATE TABLE Sparses
(
      ID INT
     ,NAME VARCHAR(100) SPARSE
)
GO
```

For details please refer - https://msdn.microsoft.com/en-IN/library/cc280604.aspx

Question39. How B-Tree forms for indexes with included column?

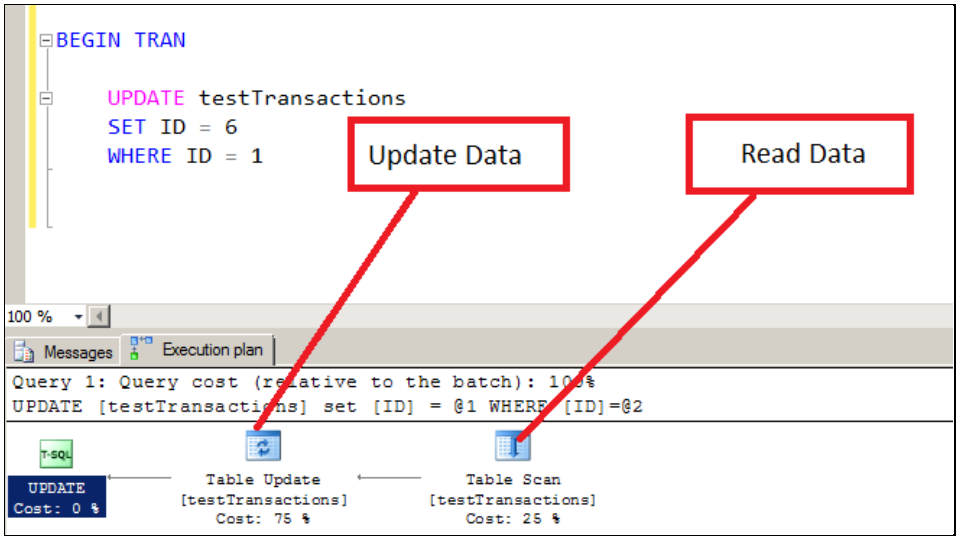
Answer -

It is the same B+ tree as we got for NonClustered Indexes. **The Difference is that at leaf level you will have key column as well as non-key columns.** Please note that you cannot add a column in Key column section and in Non-Key column section together. A column has to be in Key columns or in the non key columns section.

Question40. What happens when a transaction runs on SQL server? Let's say simple update statement "Update Table set col1 = value where col2 = value"

Update Lock is used in SQL Server when performing an UPDATE statement. When you check the execution plan, you can see that such a plan always consists of 3 parts:

- Reading data
- Get New Value
- Write data



When SQL Server initially reads the data to be changed in the first part of the query plan, Update Locks are acquired on the individual records. And finally these Update Locks are converted to Exclusive (X) Locks when the data is changed in the third part of the query plan. UPDATE Locks are required to avoid deadlock situations in UPDATE query plans.