SQL SERVER Interview Questions & Answers - SET 9 (10 Questions)

1. What is the first Version of SQL Server?

Answer-

Microsoft SQL Server is a relational database management system developed by Microsoft. The First version of SQL Server was released in 1989.

In 1988 Microsoft joined Ashton-Tate and Sybase to create a variant of Sybase SQL Server for IBM OS/2 (then developed jointly with Microsoft), which was released the following year. This was the first version of Microsoft SQL Server, and served as Microsoft's entry to the enterprise-level database market, competing against Oracle, IBM, and later, Sybase.

SQL Server 4.2 was shipped in 1992, bundled with OS/2 version 1.3, followed by version 4.21 for Windows NT, released alongside Windows NT 3.1. SQL Server 6.0 was the first version designed for NT, and did not include any direction from Sybase.

The Latest entry to the list is SQL Server 2016. Please find the history below.
(Ref - https://en.wikipedia.org/wiki/Microsoft_SQL_Server)

SQL Server release history

Version	Year	Release name	Code name	Internal version
1.0 (OS/2)	1989	SQL Server 1.0 (16 bit)	Ashton-Tate / Microsoft SQL Server	-
1.1 (OS/2)	1991	SQL Server 1.1 (16 bit)	-	-
4.2A (OS/2)	1992	SQL Server 4.2A (16 bit)	-	-
4.2B (OS/2)	1993	SQL Server 4.2B (16 bit)	-	-
4.21a (WinNT)	1993	SQL Server 4.21a	SQLNT	-
6.0	1995	SQL Server 6.0	SQL95	-
6.5	1996	SQL Server 6.5	Hydra	-
7.0	1998	SQL Server 7.0	Sphinx	515
-	1999	SQL Server 7.0 OLAP Tools	Palato mania	-
8.0	2000	SQL Server 2000	Shiloh	539
8.0	2003	SQL Server 2000 64-bit Edition	Liberty	539
9.0	2005	SQL Server 2005	Yukon	611/612
10.0	2008	SQL Server 2008	Katmai	661
10.25	2010	Azure SQL Database	Cloud Database or CloudDB	-
10.50	2010	SQL Server 2008 R2	Kilimanjaro (aka KJ)	665
11.0	2012	SQL Server 2012	Denali	706
12.0	2014	SQL Server 2014	SQL14	782
13.0	2016	SQL Server 2016	-	852

Old version | Older version, still supported | **Latest version** | Latest preview version

2. What is the maximum number of parameters we can declare in a Stored Procedure or Function in SQL Server 2016?

Answer-

The maximum number of parameters we can declare in a stored procedure or function is 2100. The value of each declared parameter must be supplied by the user when the procedure is called unless a default value for the parameter is defined or the value is set to equal another parameter.

3. What is a Latch?

Answer-

Latches are lightweight synchronization primitives that are used by the SQL Server engine to guarantee consistency of in-memory structures including; index, data pages and internal structures such as non-leaf pages in a B-Tree.

Latched are first introduced in SQL 7.0 for Data Pages. Well there are three types of latches in SQL Server.

- IO Latches
- Buffer Latches (BUF)
- Non-Buffer Latches (Non-BUF)

IO Latches & Buffer Latches (BUF)

SQL Server uses buffer latches to protect pages in the buffer pool and I/O latches to protect pages not yet loaded into the buffer pool.

Whenever data is written to or read from a page in the SQL Server buffer pool a worker thread must first acquire a buffer latch for the page.

There are various buffer latch types available for accessing pages in the buffer pool including exclusive latch (PAGELATCH_EX) and shared latch (PAGELATCH_SH).

Non-Buffer latches-

Latches are also used to protect access to internal memory structures other than buffer pool pages; these are known as Non-Buffer latches.

Latches are only controlled by SQL Server Engine.
You can see the waiting times introduced with these types of latches in the DMV **sys.dm_os_wait_stats** in the following picture.

SQLQuery11.sql - P...awankhowal01 (56))* | SUoM_pawan.sqlplan | SQLQuery10.sql - not connected* | SQLQuery

```
SELECT * FROM sys.dm_os_latch_stats
```

100 %

Results | Messages

	latch_class	waiting_requests_count	wait_time_ms	max_wait_time_ms
1	ACCESS_METHODS_DATASET_PARENT	7	0	0
2	ACCESS_METHODS_HOBT_FACTORY	0	0	0
3	ACCESS_METHODS_HOBT	7	2	1
4	ACCESS_METHODS_HOBT_COUNT	1	1	1
5	ACCESS_METHODS_HOBT_VIRTUAL_ROOT	0	0	0
6	ACCESS_METHODS_CACHE_ONLY_HOBT_ALLOC	0	0	0
7	ACCESS_METHODS_BULK_ALLOC	0	0	0
8	ACCESS_METHODS_SCAN_RANGE_GENERATOR	0	0	0
9	ACCESS_METHODS_KEY_RANGE_GENERATOR	0	0	0
10	ACCESS_METHODS_IOAFF_KEY_RANGE_GENERATOR	0	0	0
11	ACCESS_METHODS_IOAFF_KEY_TARGET_PAGE_CNT	0	0	0
12	ACCESS_METHODS_IOAFF_QUEUE	0	0	0
13	ACCESS_METHODS_IOAFF_READAHEAD_QUEUE	0	0	0
14	ACCESS_METHODS_IOAFF_READAHEAD	0	0	0
15	ACCESS_METHODS_IOAFF_WAITING_WORKER_QUEUE	0	0	0
16	APPEND_ONLY_STORAGE_INSERT_POINT	0	0	0
17	APPEND_ONLY_STORAGE_FIRST_ALLOC	0	0	0
18	APPEND_ONLY_STORAGE_UNIT_MANAGER	0	0	0
19	APPEND_ONLY_STORAGE_MANAGER	0	0	0
20	BACKUP_RESULT_SET	0	0	0
21	BACKUP_TAPE_POOL	0	0	0
22	BACKUP_LOG_REDO	0	0	0
23	BACKUP_INSTANCE_ID	0	0	0
24	BACKUP_MANAGER	0	0	0
25	BACKUP_MANAGER_DIFFERENTIAL	0	0	0
26	BACKUP_OPERATION	0	0	0

4. What is the difference between Latch and Lock?

Answer-

Latches are internal to the SQL engine and are used to provide memory consistency, whereas locks are used by SQL Server to provide logical transactional consistency. The following table compares latches to locks:

Structure	Purpose	Controlled by	Performance cost	Exposed by
Latch	Guarantee consistency of in-memory structures.	SQL Server engine only.	Performance cost is low. To allow for maximum concurrency and provide maximum performance, latches are held only for the duration of the physical operation on the in-memory structure, unlike locks which are held for the duration of the logical transaction.	??sys.dm_os_wait_stats (Transact-SQL) (http://go.microsoft.com/fwlink/p/?LinkId=212508) - Provides information on PAGELATCH, PAGEIOLATCH and LATCH wait types (LATCH_EX, LATCH_SH is used to group all non-buffer latch waits). ??sys.dm_os_latch_stats (Transact-SQL) (http://go.microsoft.com/fwlink/p/?LinkId=212510) – Provides detailed information about non-buffer latch waits. ??sys.dm_os_latch_stats (Transact-SQL) (http://go.microsoft.com/fwlink/p/?LinkId=223167)

				- This DMV provides aggregated waits for each index, which is very useful for troubleshooting latch related performance issues.

Structure	Purpose	Controlled by	Performance cost	Exposed by
Lock	Guarantee consistency of transactions.	Can be controlled by user.	Performance cost is high relative to latches as locks must be held for the duration of the transaction.	sys.dm_tran_locks (Transact-SQL) (http://go.microsoft.com/fwlink/p/?LinkId=179926). sys.dm_exec_sessions (Transact-SQL) (http://go.microsoft.com/fwlink/p/?LinkId=182932). **Note** For more information about querying SQL Server to obtain information about transaction locks see Displaying Locking Information (Database Engine) (http://go.microsoft.com/fwlink/p/?LinkId=212519).

5. What are Spinlocks?

Answer-

Spinlocks are lightweight synchronization primitives which are used to protect access to data structures. Spinlocks are not unique to SQL Server. They are generally used when it is expected that access to a given data structure will need to be held for a very short period of time.

When a thread attempting to acquire a spinlock is unable to obtain access it executes in a loop periodically checking to determine if the resource is available instead of immediately yielding. After some period of time a thread waiting on a spinlock will yield before it is able to acquire the resource in order to allow other threads running on the same CPU to execute. This is known as a backoff.
On any busy high concurrency system, it is normal to see active contention on frequently accessed structures that are protected by spinlocks.

This is only considered problematic when the contention is such that it introduces significant CPU overhead.

Spinlock statistics are exposed by the sys.dm_os_spinlock_stats Dynamic Management View (DMV) within SQL Server. For example, this query yields the following output:

```
SELECT * FROM SYS.DM_OS_SPINLOCK_STATS
ORDER BY SPINS DESC
```

Editor | Results | Messages

	name	collisions	spins	spins_per_collision	sleep_time	backoffs
1	LOCK_HASH	6581562	729325472109	110813.4	270	2386357
2	X_PACKET_LIST	264958176	28159902803	106.2806	440	159631
3	BUF_FREE_LIST	2702460	2771488056	1025.543	22989	201292
4	SOS_SCHEDULER	9926436	1460805747	147.1632	40	12793
5	SOS_OBJECT_STORE	6865257	923329442	134.4931	82	3626
6	XID_ARRAY	5138285	628308508	122.2798	24	19827
7	QUERYEXEC	1225304	586191110	478.4046	5	1465
8	SOS_SUSPEND_QUEUE	2803154	222818771	79.48859	42	2082
9	LOGCACHE_ACCESS	199922	53029933	265.2531	0	52657
10	LSID	161798	44937460	277.738	0	153
11	SOS_WAITABLE_ADDRESS_HASHBUCKET	100370	24777842	246.865	26	783
12	SOS_RW	23364	9399500	402.307	0	4607

6. What is the difference between a Latch and a Spinlock?

Answer-

SQL Server utilizes spinlocks to protect access to some of its internal data structures. These are used within the engine to serialize access to certain data structures in a similar fashion to latches.

The main difference between a latch and a spinlock is the fact that spinlocks will spin (execute a loop) for a period of time checking for availability of a data structure while a thread attempting to acquire access to a structure protected by a latch will immediately yield if the resource is not available.

Yielding requires context switching of a thread off the CPU so that another thread can execute. This is a relatively expensive operation and for resources that are held for a very short duration it is more efficient overall to allow a thread to execute in a loop periodically checking for availability of the resource.

7. What is a SSD?

Answer-

A solid-state drive (SSD, also known as a solid-state disk although it contains neither an actual disk nor a drive motor to spin a disk) is a solid-state storage device that uses integrated circuit assemblies as memory to store data persistently.

SSDs don't have any moving parts so when a read or write occurs I/O latency will be almost zero. The latency in a spinning drive comes from two things:

- Moving the disk head to the right track on the disk surface (known as the seek time)
- Waiting for the disk to spin to the right point on the track (known as the rotational latency)

This means that SSDs provide a big performance boost when there's an I/O bottleneck.

Also note that SSD performance can start to degrade as the drive gets really full.

8. What you mean by CXPACKET?

Answer-

It is a common wait type. CXPACKET means there are queries running in parallel in our servers and we will always see CXPACKET waits for a parallel query. CXPACKET waits does NOT mean that we have a problematic parallelism. You have to dig deeper to determine whether you have an issue or not. The first thing one should look at is whether you expect parallelism for the query that's using it. Shall explain this in detail in one of the upcoming posts.

9. How many kinds of database files present in SQL Server?

Answer-

View can be used for the following purposes:

- To focus, simplify, and customize the perception each user has of the database.
- As a security mechanism by allowing users to access data through the view, without granting the users permissions to directly access the underlying base tables.
- To provide a backward compatible interface to emulate a table whose schema has changed.

Simple views are expanded inline & They DO NOT directly contribute to performance improvements. However, indexed

views can dramatically improve performance. Indexed views have lot of restrictions, so be careful while using.

10. Provide Outputs of SQL Queries?

```
SELECT ROUND( 109.87 , - 1 ) , ROUND( 104.99 , - 1 ) ,
ROUND( 150 , - 2 )

SELECT ROUND(748.58, -1)  , ROUND(748.58, -2),
     ROUND(751.58, -2)

SELECT ROUND(444, -1) ,  ROUND(444, -2) , ROUND(748.58, -
0), ROUND(748.58, -4)
```


Answer-

ROUND always returns a value. If length is negative and larger than the number of digits before the decimal point, ROUND returns 0.

Example	Result
ROUND(748.58, -4)	0

ROUND returns a rounded *numeric_expression*, regardless of data type, when *length* is a negative number.

Examples	Result
ROUND(748.58, -1)	750.00

Examples	Result
ROUND(748.58, -2)	700.00
ROUND(748.58, -3)	Results in an arithmetic overflow, because 748.58 defaults to decimal(5,2), which cannot return 1000.00.
To round up to 4 digits, change the data type of the input. For example: SELECT ROUND(CAST (748.58 AS decimal (6,2)),-3);	1000.00

```
SELECT ROUND( 109.87 , - 1 ) , ROUND( 104.99 , - 1 ) , ROUND( 150 , - 2 )

SELECT ROUND(748.58, -1)  , ROUND(748.58, -2),  ROUND(751.58, -2)

SELECT ROUND(444, -1) ,  ROUND(444, -2) , ROUND(748.58, -0), ROUND(748.58, -4)
```

100 %

Results | Messages

	(No column name)	(No column name)	(No column name)
1	110.00	100.00	200

	(No column name)	(No column name)	(No column name)
1	750.00	700.00	800.00

	(No column name)	(No column name)	(No column name)	(No column name)
1	440	400	749.00	0.00

That's all folks; I hope you've enjoyed the article and I'll see you soon with some more articles.

Thanks!

Pawan Kumar Khowal

MSBISKills.com