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

1. What is a CheckPoint?

Answer-

There are two built in automatic mechanisms SQL Server uses to scan the buffer cache periodically and writes any dirty pages to disk.

- The Lazy Writer
- Checkpoint process

Checkpoint

A checkpoint always writes out all pages that have changed (known as being marked dirty) since the last checkpoint, or since the page was read in from disk. It doesn't matter whether the transaction that changed the page has committed or not. The page is written to disk regardless.

The goal of checkpoint is to reduce the amount of time Sql Server takes to perform **rollforward** and **rollback** operations.

The only exception is for tempDB, where data pages are not written to disk as part of a checkpoint. A checkpoint is only done for tempDB when the tempDB log file reaches 70% full – this is to prevent the tempDB log from growing if at all possible (note that a long-running transaction can still essentially hold the log hostage and prevent it from clearing, just like in a user database).

Checkpoint can be triggered from any of the below -

- From a Manual Checkpoint

```
CheckPoint
GO
```

- From a database or differential backup

```
ALTER DATABASE DBName SET TARGET_RECOVERY_TIME = target_recovery_time {
SECONDS | MINUTES}
GO;
```

- Automatically

```
EXEC [sp_configure] '[recovery interval]', 'seconds'
GO;
```

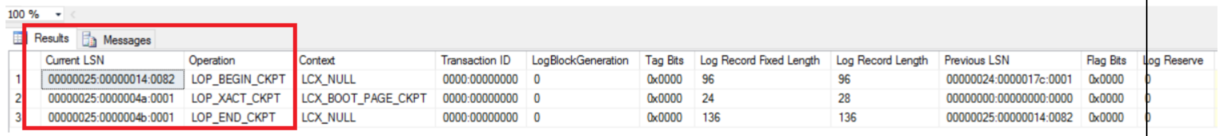
Example

```
CREATE TABLE tab
(
      Id INT
)
GO

INSERT INTO tab VALUES (1), (2), (3)
GO


CHECKPOINT
GO


SELECT * FROM FN_DBLOG (NULL, NULL)
GO
```



	Current LSN	Operation	Context	Transaction ID	LogBlockGeneration	Tag Bits	Log Record Fixed Length	Log Record Length	Previous LSN	Flag Bits	Log Reserve
1	00000025:00000014:0082	LOP_BEGIN_CKPT	LCX_NULL	0000:00000000	0	0x0000	96	96	00000024:0000017c:0001	0x0000	0
2	00000025:0000004a:0001	LOP_XACT_CKPT	LCX_BOOT_PAGE_CKPT	0000:00000000	0	0x0000	24	28	00000000:00000000:0000	0x0000	0
3	00000025:0000004b:0001	LOP_END_CKPT	LCX_NULL	0000:00000000	0	0x0000	136	136	00000025:00000014:0082	0x0000	0

2. What is a Lazy Writer?

Answer-

There are two built in automatic mechanisms SQL Server uses to scan the buffer cache periodically and writes any dirty pages to disk.

- The Lazy Writer
- Checkpoint process

The Lazy Writer

The function of LAZY WRITER is to release the buffer pool memory. When there is a memory pressure and more memory required e.g. for bringing in new pages to the cache. (Buffer Pool / Data Cache / Buffer Cache), Lazy writer releases the “coldest” pages from the buffer pool and makes more memory available for new pages to come in. **Cold Pages means least recently read or written, not accessed in recent time.**

If lazy writer constantly has a lots of work to do, starting to purge pages that are not old at all you have a problem with buffer cache memory. You do not want page’s flow though the buffer cache like a free flowing water. You want them to stay there and be reused, read/written and rewritten again in-memory and not the disk which is slow compared to memory, as long as possible.

3. What is the difference between Lazy Writer and CheckPoint?

Answer-

They both write in-memory pages to the data files on the disk. But which pages, when, and do they release memory or not – there is the difference!

CHECKPOINT writes only **dirty** pages to the disk (dirty = changed in memory since the last checkpoint, not yet written/check pointed to disk), making them “clean”. Checkpoint does not release any memory, the pages STAY in memory, they are not removed from the buffer pool!

LAZY WRITER looks for **least recently used** (“cold” = least recently read or written, not accessed in recent time) pages in the buffer pool, and releases the memory taken by them. Lazy writer releases both **dirty and clean** pages. Clean pages can be released without writing to disk, but dirty pages must first be written to the disk (“flushed” to the disk and become “clean”) and then buffer pool memory can be released. So, total number of pages that lazy writer releases can be higher than the number of pages’ lazy writer writes to the data files, because “clean” pages can be released without writing them to disk. The final result of the lazy writer is less buffer pool memory used, therefore more memory available for the fresh pages in the buffer pool.

Checkpoint process is more efficient in writing to the disk because it can group subsequent pages into larger disk IOs, e.g. 128KB IO size. It internally uses WriteFileGather Windows API function.

Lazy writer can only write 8K pages. Therefore, checkpoint disk throughput is much better than lazy writer's.

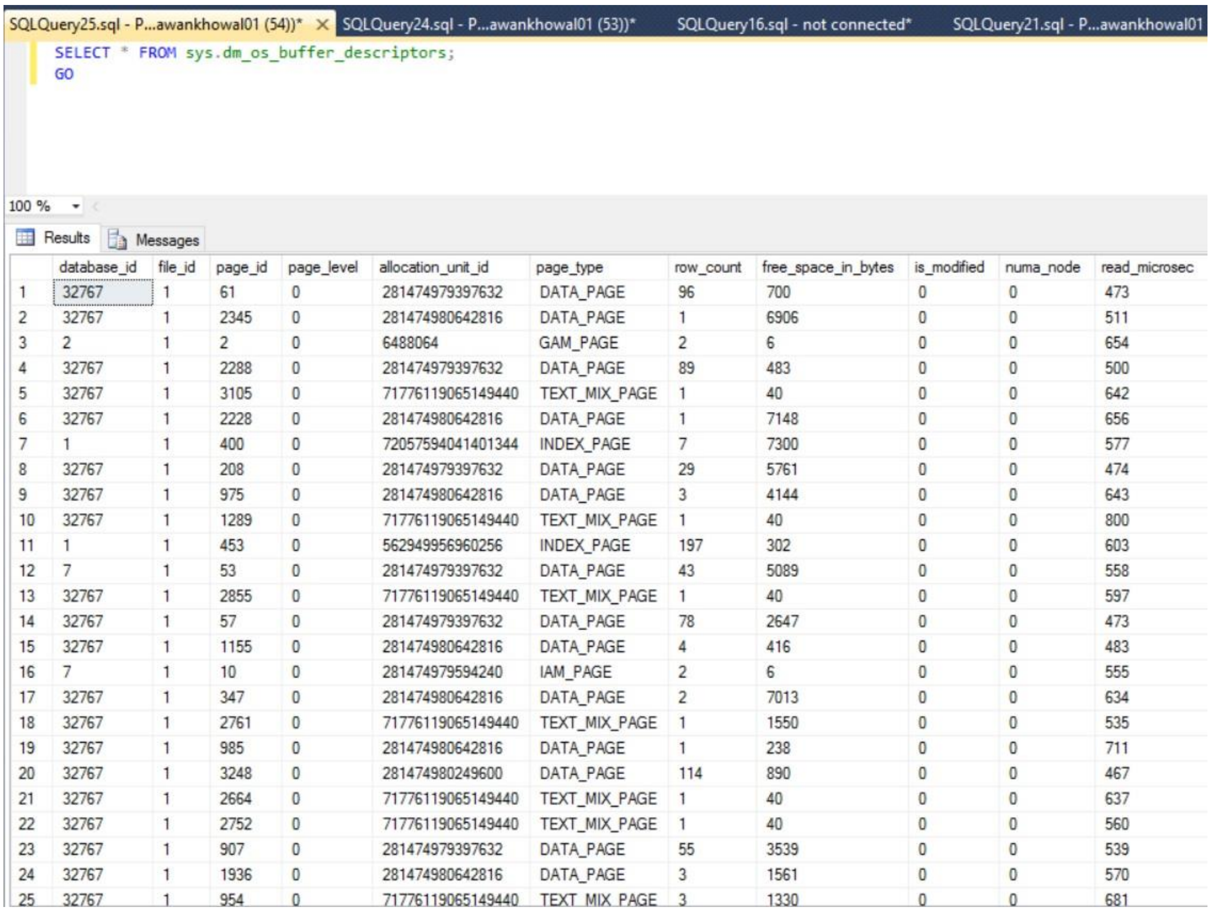
4. What is a Buffer Pool?

Answer-

A SQL Server buffer pool, also called an SQL Server buffer cache, is a place in system memory that is used for caching table and index data pages as they are modified or read from disk.

Pages are stored in buffers in a Buffer Pool. The buffers are hashed so that they can easily found by DB.

You can see what pages are currently in the buffer pool, and their status using the sys.dm_os_buffer_descriptors


SQLQuery25.sql - P...awankhowal01 (54))*
SQLQuery24.sql - P...awankhowal01 (53))*
SQLQuery16.sql - not connected*
SQLQuery21.sql - P...awankhowal01
100 %
Results
Messages

```
SELECT * FROM sys.dm_os_buffer_descriptors;
GO
```

	database_id	file_id	page_id	page_level	allocation_unit_id	page_type	row_count	free_space_in_bytes	is_modified	numa_node	read_microsec
1	32767	1	61	0	281474979397632	DATA_PAGE	96	700	0	0	473
2	32767	1	2345	0	281474980642816	DATA_PAGE	1	6906	0	0	511
3	2	1	2	0	6488064	GAM_PAGE	2	6	0	0	654
4	32767	1	2288	0	281474979397632	DATA_PAGE	89	483	0	0	500
5	32767	1	3105	0	71776119065149440	TEXT_MIX_PAGE	1	40	0	0	642
6	32767	1	2228	0	281474980642816	DATA_PAGE	1	7148	0	0	656
7	1	1	400	0	72057594041401344	INDEX_PAGE	7	7300	0	0	577
8	32767	1	208	0	281474979397632	DATA_PAGE	29	5761	0	0	474
9	32767	1	975	0	281474980642816	DATA_PAGE	3	4144	0	0	643
10	32767	1	1289	0	71776119065149440	TEXT_MIX_PAGE	1	40	0	0	800
11	1	1	453	0	562949956960256	INDEX_PAGE	197	302	0	0	603
12	7	1	53	0	281474979397632	DATA_PAGE	43	5089	0	0	558
13	32767	1	2855	0	71776119065149440	TEXT_MIX_PAGE	1	40	0	0	597
14	32767	1	57	0	281474979397632	DATA_PAGE	78	2647	0	0	473
15	32767	1	1155	0	281474980642816	DATA_PAGE	4	416	0	0	483
16	7	1	10	0	281474979594240	IAM_PAGE	2	6	0	0	555
17	32767	1	347	0	281474980642816	DATA_PAGE	2	7013	0	0	634
18	32767	1	2761	0	71776119065149440	TEXT_MIX_PAGE	1	1550	0	0	535
19	32767	1	985	0	281474980642816	DATA_PAGE	1	238	0	0	711
20	32767	1	3248	0	281474980249600	DATA_PAGE	114	890	0	0	467
21	32767	1	2664	0	71776119065149440	TEXT_MIX_PAGE	1	40	0	0	637
22	32767	1	2752	0	71776119065149440	TEXT_MIX_PAGE	1	40	0	0	560
23	32767	1	907	0	281474979397632	DATA_PAGE	55	3539	0	0	539
24	32767	1	1936	0	281474980642816	DATA_PAGE	3	1561	0	0	570
25	32767	1	954	0	71776119065149440	TEXT_MIX_PAGE	3	1330	0	0	681

5. What is a Log Flush?

Answer-

Log Flush also writes pages to disk. It writes pages from Log Cache into the Transactional log file (LDF). Once a transaction completes, Log Flush writes those pages (from Log Cache) to LDF file on disk.

Each and every transaction that results in data page changes, also incurs some Log Cache changes. At the end of each transaction (commit), these changes from Log Cache are flushed down to the physical file (LDF). So, in essence, the number of log flushes depend on number of transactions.

Note that SELECT statements do not result in any data page changes, so there are no changes to Log Cache, so no Log Flushes to Transactional Log file.

6. What is the Output of below statement?

```
CREATE TABLE Addy
(
        ID SmallINT IDENTITY(1,1)
      , Name VARCHAR(5) DEFAULT 'Pawan'
      ,
)
GO
```

Answer-

You will not get any error when you execute the above statement. You will get “***Command(s) completed successfully.”.*** This is a well know issue with the SQL Server.

7. Is there a way to Select Identity column without specifying Identity column name in the select list?

Answer-

I am not sure where we will get this kind of requirement. Yes, there are two ways to achieve this.

Solution 1 | IDENTITYCOL

```
CREATE TABLE Addy
(
        ID SmallINT IDENTITY(1,1)
      , Name VARCHAR(5) DEFAULT 'Pawan'
      ,
)
GO

INSERT INTO Addy DEFAULT VALUES
GO 2
```

```
SELECT IDENTITYCOL, Name FROM Addy
```

Solution 2 | $IDENTITY

```
CREATE TABLE Addy1
(
	ID SmallINT IDENTITY(1,1)
      , Name VARCHAR(5) DEFAULT 'Pawan'
      ,
)
GO

INSERT INTO Addy1 DEFAULT VALUES
GO 2

SELECT $IDENTITY, Name FROM Addy1
```

These are reserved words. Microsoft SQL Server uses reserved keywords for defining, manipulating, and accessing databases. Reserved keywords are part of the grammar of the Transact-SQL language that is used by SQL Server to parse and understand Transact-SQL statements and batches.

Although it is syntactically possible to use SQL Server reserved keywords as identifiers and object names in Transact-SQL scripts, you can do this only by using delimited identifiers.

8. What is the purpose of Resource DB in SQL Server?

Answer-

1. The Resource database is a read-only database that contains all the system objects that are included with SQL Server.

2. SQL Server system objects, such as sys.objects, are physically persisted in the Resource database, but they logically appear in the sys schema of every database.

3. The Resource database does not contain user data or user metadata.

4. The Resource database is used in upgrading to a new version of SQL Server an easier and faster procedure.

For details please refer below -

https://msbiskills.com/2015/09/07/sql-server-resource-database-deep-dive/

9. Provide Outputs of SQL Queries?

```
DECLARE @ NVARCHAR(6) = N'पवन'
SELECT @
GO


DECLARE @ NVARCHAR(6) = 'पवन'
SELECT @
GO
```

Answer-

Output of 1st Query –

पवन

Output of 2nd Query – The reason for this is we have not provided N before the initialization part.

???

10. What is a live lock? How it is different from Deadlock?

Answer-

> A deadlock occurs when two processes compete for the same resources, but in an order which causes a stalemate. For example, A locks X, then tries to lock Y, while B has locked Y and tries to lock X. The key is that the two (or more) processes are preventing each other from doing anything.
>
> A livelock occurs when there are overlapping shared locks that prevent another process from acquiring the exclusive lock it needs. The difference is that all of these overlapping processes continue to get their work done, so they are still "live" - only the victim is blocked until they are done. Which may be never on a busy enough, poorly designed system. :-) You may be able to overcome this situation by escalating the deadlock priority for your writers, but I'll be honest, this isn't a scenario I've seen very often, and I've worked with SQL Server since 2000.

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