SQL SERVER Interview Questions & Answers - SET 7 (10 Questions)

1. What are Ghost Rows in an Index in SQL Server?

Answer-

At leaf level of an index we have actual data rows. Now when these rows are deleted, they're marked as ghost records.

This means that the row stays on the page but a bit is changed in the row header to indicate that the row is really a ghost.

The page header also reflects the number of ghost records on a page.

When user fires the delete command, SQL returns to the user much faster because it does not have to wait for the records to be deleted physically. Internally they are marked as "ghosted".

Ghost records are present only in the index leaf nodes.

The ghost records can be cleaned up in 3 ways:

- If a record of the same key value as the deleted record is inserted
- If the page needs to be split, the ghost records will be handled

- The Ghost clean-up task (scheduled to run once every 5 seconds). It asynchronously removes them from the leaf level of the index. This same thread carries out the automatic shrinking of databases if you have that option set.

The Ghost cleanup process divides the “ghost pages” into 2 categories:

- Hot Pages (frequently visited by scanning processes)
- Cold Pages

The Ghost cleanup thread is able to retrieve the list of Cold pages from the DBTABLE for that database, or the PFS Page for that interval. The cleanup task cleans up a maximum of 10 ghost pages at a time. Also, while searching for the ghost pages, if it covers 10 PFS Pages, it yields.

As far as hot ghost pages are concerned, the ghost cleanup strives to keep the number of such pages below a specified limit. Also, if the thread cleans up 10 hot ghost pages, it yields. However, if the number of hot ghost pages is above the specified (hard-coded) limit, the task runs non-stop till the count comes down below the threshold value.

If there is no CPU usage on the system, the Ghost cleanup task runs till there are no more ghost pages to clean up.

One of the most common (and quickest) resolutions for a ghost records issue is to restart SQL Server.

2. How temp DB is created or algorithm used to create tempDB?

Answer-

Creation / Birth of tempDB uses below algorithm-

1. First Master DB is open
2. Second Open Model DB
3. TempDB is created using Model DB properties
   - Engine will first lock model DB and tempDB
   - Connections are allowed at this point to the engine, since master DB is already gone.
   - User DBs are started to recover
   - Create the primary DB file for tempDB based on the properties of master
   - Copy extents from model to primary DB file including objects
   - Fix up primary DB file for tempDB
   - Create the primary transaction log file
   - Create and attach other files
   - Notify that tempDB is ready
4. If the above process fails, we will shut down the server

3. What is Spatial Index?

Answer-

SQL Server's spatial data type allows us to store spatial objects and make them available to an application.

SQL Server supports two spatial data types:

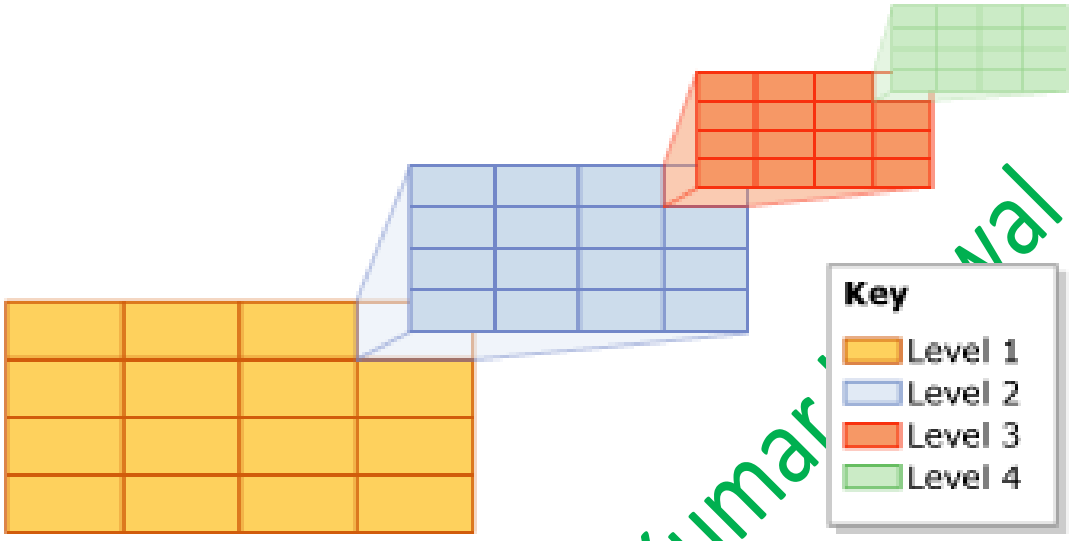
Geometry: Stores the X and Y coordinates that represents lines, points, or polygons.

Geography: Stores the latitude and longitude coordinates that represent lines, points, or polygons.

SQL Server uses a B+ tree structure, which is a variation of the B-tree index. B-tree is nothing but a data structure that keeps data sorted to support search operations, sequential access, and data modifications such as inserts and deletes. SQL Server spatial indexes are built on top of the B+ tree structure, which allows the indexes to use that structure and its access methods. The spatial indexes also use the fundamental principles of XML indexing. XML indexing was introduced in SQL Server 2005 and supports two basic types of indexes: primary and secondary. The primary XML index is a B+ tree that essentially contains one row for each node in the XML instance.

So how does SQL Server implement the spatial index? As already mentioned, SQL Server starts with a B+ tree structure, which organizes data into a linear fashion. Because of this, the indexes must have a way to represent the two-dimensional spatial information as linear data. For this, SQL Server uses a process referred to as the hierarchical uniform decomposition of space.

When the index is created, the database engine decomposes, or refactors, the space into a collection of axes aligned along a four-level grid hierarchy.



https://msdn.microsoft.com/en-IN/library/bb895265.aspx

4. What is the default recursion level for a CTE in SQL SERVER?

Answer-

1. The default maximum recursion depth of a CTE is 100 and if your query go beyond that an error occurs.

2. One can control the maximum recursion depth using the MAXRECURSION query hint (a value between 1 and 32767) and if your query go beyond that an error occurs.

3. If one wants to go ahead without limit can use the MAXRECURSION query hint (a value of 0). In this case we may crash our server due to an infinite loop. So your

code should have manually control the execution otherwise it will go into the infinite loop.

4. E.g.

```
WITH Looper AS
(
    SELECT 0 AS Levels

    UNION ALL

    SELECT (x.Levels + 1) AS
RecursionLevel
    FROM Looper x
    WHERE (x.Levels + 1) <= 200
)
SELECT * FROM Looper
OPTION (MAXRECURSION 200)
```

5. Can you explain me Halloween Problem?

Answer –

As per Wikipedia (https://en.wikipedia.org/wiki/Halloween_Problem)

Halloween Problem refers to a phenomenon in databases in which an update operation causes a change in the physical location of a row, potentially allowing the row to be visited more than once during

the operation. This could even cause an infinite loop in some cases where updates continually place the updated record ahead of the scan performing the update operation.

The potential for this database error was first discovered by Don Chamberlin, Pat Selinger, and Morton Astrahan in 1976, on Halloween day while working on a query that was supposed to give a ten percent raise to every employee who earned less than $25,000. This query would run successfully, with no errors, but when finished all the employees in the database earned at least $25,000, because it kept giving them a raise until they reached that level. The expectation was that the query would iterate over each of the employee records with a salary less than $25,000 precisely once. In fact, because even updated records were visible to the query execution engine and so continued to match the query's criteria, salary records were matching multiple times and each time being given a 10% raise until they were all greater than $25,000.

SQL Optimizer uses a Spool Operator to overcome Halloween protection.

```
CREATE TABLE testHalloween
(
     a SMALLINT PRIMARY KEY
    ,b INT
```

```
    ,c INT
)
GO

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

CREATE NONCLUSTERED INDEX Ix_c ON
testHalloween(c)
GO

--Optimizer will add a Table Spool for
Halloween Protection

UPDATE testHalloween
SET c = c * 2
FROM testHalloween WITH ( INDEX ( Ix_c ) )
WHERE c < 3
GO
```


6. Is it true that stored procedures compiled when they are created?

Answer –

No, it is NOT true that stored procedures compiled when they are created.

Stored procedure gets compiled when they get executed on the first RUN. When the user fires an EXEC or EXECUTE proc command first time, then the stored procedure gets complied.

You can easily check this using following steps-

Create an SP and check if you get anything in SQL Profiler.

In the second step execute the stored procedure and then check the profiler. The events you should be checking are given below-

1. SP:CacheHit
2. SP:CacheInsert
3. SP:CacheMiss
4. SP:CacheRemove

7. Can we create a Trigger on a View?

Answer –

Yes, we can create a trigger on View. View can be simple or updateable.

We can only create "Instead of Trigger" on a view separately for each of the INSERT, UPDATE, DELETE operation.

INSTEAD OF triggers can be defined on either tables or views; however, INSTEAD OF triggers are most useful for extending the types of updates a view can support. For example, INSTEAD OF triggers can provide the logic to

modify multiple base tables through a view or to modify base tables that contain the following columns:

1. timestamp data type
2. Computed columns
3. Identity columns

8. Why Truncate Command is faster than Delete Command if we are removing all data from the table?

Answer-

DELETE and TRUNCATE statements can be used to delete all data from the table.

DELETE is a logged operation on a per row basis. This means that the deletion of each row gets logged and physically deleted.

TRUNCATE is also a logged operation (Fully & Efficiently logged), but in a different way.

A TRUNCATE TABLE operation does a wholesale delete of all data in the table. The individual records are not deleted one-by-one, instead the data pages comprising the table are simply deallocated. The allocations are unhooked from the table and put onto a queue to be deallocated by a background task called the deferred-drop task. The deferred-drop task does the deallocations instead of them being done as part of the regular

transaction so that no locks need to be acquired while deallocating entire extents.

The deallocation of data pages means that your data rows still actually exist in the data pages, but the extents have been marked as empty for reuse. This is what makes TRUNCATE a faster operation to perform over DELETE.

9. What kind of triggers are "DDL triggers"?

Answer –

There are 2 kinds of Triggers in SQL Server

- Instead of Triggers
- After Triggers

DDL triggers are implemented as AFTER triggers, which means the operation occurs and is then caught in the trigger (and optionally rolled-back, if you put a ROLLBACK statement in the trigger body).
This means they're not quite as lightweight as you might think.

DDL triggers are relatively expensive in nature. What would be better in this case is to specifically GRANT or DENY the ALTER permission. DDL triggers allow you to perform auditing of who did what, so I'm not saying they're without use – just be careful.

10. What kind of triggers are “DDL triggers”?

Answer –

If changes need to be made to a column it is necessary to perform some impact assessment in order to determine what objects will be affected, meaning that SQL table column dependencies within a SQL Server database need to be found.

One of the ways is to use a SYSCOMMENTS table. It is a table which contains entries for each view, rule, default, trigger, CHECK constraint, DEFAULT constraint, and stored procedure. The column TEXT in the syscomments table contains the actual code for all these objects, and knowing it you can write a code to check the dependencies:

```
SELECT Name
FROM syscomments c
JOIN sysobjects o on c.id = o.id
WHERE TEXT LIKE '%Person%' AND TEXT
LIKE '%FirstName%'
```

The query above is used to determine which objects use the FirstName column of the Person table.

	name
1	vStoreWithContacts
2	vVendorWithContacts
3	ufnGetContactInformation
4	uspGetEmployeeManagers
5	uspGetManagerEmployees
6	tr_u_AUDIT_Person
7	vAdditionalContactInfo
8	vEmployee
9	vEmployeeDepartment
10	vEmployeeDepartmentHistory

This is just a simplified example that contain some basic information. The downside with this method is that it will return all objects that merely mention words "Person" and "FirstName" without actually making a reference to Person.FirstName column

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