Day 10 Task Part2

Comparison

Name: Ahmed Taha

Track: PD

1. **Batch, script, transaction**

* **Batch**:
  + Set of t-SQL commands related and unrelated executed together at the same time and they don’t affect each other if there is an error in anyone of them it doesn’t affect the other statements
  + But careful if you type insert statement and select statement after it might be considered as one statement and give you different behavior so to solve this issue, we use script.
  + to use batch just highlight group of statements and execute them
* **Script**:
  + Same as batch but we separate between each unrelated statement with go keyword
  + You can also use script by highlighting the group of statements and use them together
* **Transaction**:
  + A transaction is a set of commands that are guaranteed to succeed or fail
  + You set conditions for your transaction if it fails, we use rollback to undo all the executed statements and if the condition is true, we use commit key word to go on with the execution

2-**trigger and stored procedure**

-Stored procedure: a user-defined piece of code, which may return a value (making it a function) that is invoked by calling it explicitly.

-Trigger: a stored procedure that runs automatically when various events happen (update, insert, delete), can work also with DDL query

Such as alter-drop ….

3-**stored procedure and functions**

-Stored procedure: a user-defined piece of code, which may return a value (making it a function) that is invoked by calling it explicitly. Can have input and output parameters. Cannot be called by functions. Allow DML queries. Can use Try and Catch. Can use transactions

-Functions: a user-defined piece of code which must return a value. Have only inputs parameters. Can be called by stored procedures. Allow only Select statements. Cannot use Try and Catch. Cannot use transactions.

4-**Drop**:

-Removes a table from the database.

-All tables' rows, indexes, and privileges will also be removed.

-DDL command.

-Cannot be rolled back.

-**Truncate**:

-Is a DDL command.

-Removes all rows from a table.

-Faster in performance because it doesn’t use log in 99% of the cases.

-Resets the identity column.

-**Delete**:

-Is a DML command.

-Can use where clause with DELETE to filter & delete specific records.

-It maintains the log, so it slower than TRUNCATE.

-Does not rest the identity column.

5- **select and select into statement**

-**Select**: to display or show an already existing data from an already existing table.

-**Select into**: to create a new table and insert the data of the selection to that table also can be use with false where condition to just get the table structure.

6- local and global variables

* **Local Variables:** scope of variables is inside the batch, function or stored procedure. Declared by @ before the variable name. Can be assigned.
* **Global Variables:** scope of variables is through the server or the DB. Declared by @@ before the variable name. Cannot be declared and it is assigned automatically.

7- convert and cast statements

* **Convert:** SQL function used to convert one data type to another. Accepts an optional style parameter, which is used for formatting
* **Cast:** SQL function used to convert one data type to another

8- **DDL,DML,DCL,DQL and TCL**

* **DDL:** Data Definition Language. SQL commands that can be used to define the database schema. Deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. *create*, *drop*, *alter*, *truncate*.
* **DML:** Data Manipulation Language. SQL commands that deals with the manipulation of data present in the database. It is the component of the SQL statement that controls access to data and to the database. E.g., *insert*, *update*, *delete*
* **DCL:** Data Control Language. DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system. E.g., *grant*, *revoke*.
* **DQL:** Data Query Language. DQLstatements are used for performing queries on the data within schema objects. It is a component of SQL statement that allows getting data from the database and imposing order upon it. E.g., *Select*
* **TCL:** Transaction Control Language. TCL commands helps the user manage the transactions that take place in a database. E.g., *commit, Rollback, Save Point.*

**9**-**For xml raw and for xml auto**.

* **For XML Raw:** is a clause represents the query output from SQL as XML. Can deal properly the data returned from a single table and represent it correctly as XML. By default, the FOR XML construct will display the columns as attributes. We used the “ELEMENTS” keyword to display the columns as elements
* **For XML Auto:** is a clause represents the query output from SQL as XML. It can handle query of two joined tables properly and display the XML file without repetition.

**10- Table valued and multi statement function**

* **Table valued function:** You simply state RETURNS TABLE and the return table’s definition will be based on the function’s SELECT statement. No need to specify the structure of the return table. Do not use the BEGIN/END syntax. Generally, faster than multi statement function.
* **Multi statement function:** Your RETURNS syntax explicitly specifies the structure of the return table. This is done by declaring a TABLE variable that will be used to store and accumulate the rows that are returned as the value of the function. Use the BEGIN/END syntax. Generally slower than Table valued function.

**11-Varchar(50) and varchar(max)**

* **Varchar(50):** Character data type. Used with character data type with maximum of 50 characters. If smaller characters are entered to the column data, will store the only entered characters at the memory as opposed to char data type. Used mainly when the sizes of the column data entries vary considerably.
* **Varchar(max):** Character data type. If smaller characters are entered to the column data, will store the only entered characters at the memory as opposed to char data type. Used mainly when the sizes of the column data entries vary considerably, and the string length might exceed 8,000 bytes.

**12-Datetime(3), datetime2(7) and datetimeoffset(7)**

* **datetime():**
* Takes 8 bytes in storage.
* Character length is 19 positions minimum & 23 maximum.
* Date range is 1753-01-01 through 9999-12-3.
* Time range is 00:00:00 through 23:59:59.997.
* Accuracy is rounded to increments of .000, .003, or .007 seconds.
* Has no Time Zone Offset.
* **datetime2():**
* Takes 6 to 8 bytes, depending on the precision plus 1 byte to store the precision.
* Character length is 19 positions minimum & 27 maximum.
* Date range is 0001-01-01 through 9999-12-31.
* Time range is 00:00:00 through 23:59:59.9999999
* Accuracy is 100 nanoseconds.
* Has no Time Zone Offset.
* **Datetimeoffset():**
* 8 to 10 bytes, depending on the precision Plus 1 byte to store the precision
* Character length is 26 positions minimum & 34 maximum
* Date range is 0001-01-01 through 9999-12-31
* Time range is 00:00:00 through 23:59:59.9999999
* Accuracy is 100 nanoseconds
* Time zone offset range is -14:00 through +14:00

**13-Default instance and named instance**

* **Default** **Instance**:
* A type of instance that is used when installing a single instance of SQL server.
* There is one default instance
* If the user plans to install a single instance of SQL server, it is a default instance.
* **Default** **Instance**:
* A named instance is a type of instance where the user specifies an instance name when installing the instance.
* There are multiple named instances.
* If the user plans to install multiple instances on the same computer, then the instances other than the default instance are named instances.

**14-SQL and windows Authentication**

* **Windows Authentication:** When you are accessing SQL Server from the same computer it is installed on, you should not be prompted to type in a username and password. In addition, you are not, if you are using Windows Authentication. With Windows Authentication, the SQL Server service already knows that someone is logged in into the operating system with the correct credentials, and it uses these credentials to allow the user into its databases.
* **SQL Authentication:** QL Authentication is the typical authentication used for various database systems, composed of a username and a password. Obviously, an instance of SQL Server can have multiple such user accounts (using SQL authentication) with different usernames and passwords. In shared servers where different users should have access to different databases, SQL authentication should be used. In addition, when a client (remote computer) connects to an instance of SQL Server on other computer than the one on which the client is running, SQL Server authentication is needed.

**15-Clustered and non-clustered index**

* **Clustered Index:**
* Clustered index is faster.
* Clustered index requires less memory for operations.
* In clustered index, index is the main data.
* A table can have only one clustered index.
* Clustered index has inherent ability of storing data on the disk.
* Clustered index store pointers to block not data.
* In Clustered index leaf nodes are actual data itself.
* In Clustered index, Clustered key defines order of data within table.
* A Clustered index is a type of index in which table records are physically reordered to match the index.
* **non-clustered index:**
* Non-clustered index is slower.
* Non-clustered index is slower.
* In Non-Clustered index, index is the copy of data.
* A table can have multiple non-clustered index.
* Non-Clustered index store both value and a pointer to actual row that holds data.
* In Non-Clustered index leaf nodes are not the actual data itself rather they only contains included columns.
* In Non-Clustered index, index key defines order of data within index.
* A Non-Clustered index is a special type of index in which logical order of index does not match physical stored order of the rows on disk.

**16-Group by rollup and group by cube**

* **Group by rollup:** Is used to calculate sub-totals and grand totals for a set of columns passed to the “GROUP BY ROLLUP” clause. Can also be used to calculate sub-totals for each column, based on the groupings within that column. It is a combination of 2 or 3 queries.
* **Group by Cube:** Is used in combination with the GROUP BY clause, however the CUBE operator produces results by generating all combinations of columns specified in the GROUP BY CUBE clause. It is a combination of 4 queries

**17-Sequence object and identity**

* **Identify:**
* Identity is a table level object, in other words it is dependent on the table.
* This property is set or used with the CREATE TABLE and ALTER TABLE statements.
* We cannot restart the Identity counter after the specified interval.
* We cannot cache the identity column.
* We cannot define the maximum value for an Identity column. It is dependent on the data type for the identity column.
* We can reseed an Identity property but we cannot change the step size.
* We cannot generate a range for an identity column.
* If any column is marked as an Identity then we cannot insert data within this column directly. We must first turn off the Identity of the column.
* We cannot get the value of an Identity column before inserting a record.
* We cannot create an Identity property in descending order.
* **Sequence:**
* Sequence is a database-level object so it is independent of tables.
* A Sequence object allows us to synchronize a seed value across the multiple tables
* Using the CYCLE property, we can restart the counter after a specific interval.
* Using the “CACHE” property, we can cache a sequence and improve the performance of SQL Server.
* Using the MAXVALUE property, we can define a maximum value for the sequence.
* We can alter the seed well as the Step size of a Sequence object any time.
* Using the "sp\_sequence\_get\_range" Stored Procedure, we can generate a range of sequence numbers from the sequence object.
* A Sequence does not depend on the table so we can insert any value in the column.
* We can get the value of the next Sequence number for a Sequence Object before inserting a record.
* We can create a Sequence number in descending order using a sequence object.

**18-Inline function and view**

* **View:**
* You can write Update statements *against* them just as though they were an updatable view.
* The engine can optimize views. For example, if you run SELECT \* FROM view1 WHERE x = 10; and you have index on table field that maps to X, then it will be used.
* Does not accept parameters.
* Can have triggers since they can be used to change underlying tables.
* Can use side-effecting operator
* **Inline Function:**
* Can accept parameters
* Must have a return
* Cannot be indexed and performance decreases when number of rows increases.
* Cannot have triggers.
* Cannot use side-effecting operator.

**19-Table variable and temporary table**

* **Table Variable**:
* Acts like a variable and exists for a particular batch of query execution.
* It is dropped once it comes out of batch.
* It is created in the memory database but may be pushed out to tempdb.
* Table variable can be passed as a parameter to functions and stored procedures.
* Table variable is faster than temporary table.
* **Temporary Tables**:
* Are physically created in the tempdb database.
* These tables act as the normal table and can have constraints, index like normal tables.
* Cannot be passed as a parameter to functions and stored procedures.
* Its lifetime could be locally (session-based) or across many users while all logging in the server.

**20-Row\_number() and dense\_Rank() function**

* **Row Number():**
* Generates a unique ranking even with [duplicate records](http://javarevisited.blogspot.com/2012/12/how-to-find-duplicate-records-in-table-mysql-query-example.html)**.**
* If the ORDER BY clause cannot distinguish between two rows, it will still give them different rankings, though which record will come earlier or later is decided randomly.
* For example: If two employees *Shane* and *Rick* have the same salary and has row number 4 and 5, this is random, if you run again, Shane might come 5th.
* **Dense\_Rank():**
* Gives the same ranking to rows that cannot be distinguished by the order by clause.
* Generates a contiguous sequence of ranks like (1,2,3,...).