Sql Server questions and terms

# Atomic Blocks

All statements within a transaction will either succeed for fail within the atomic block. If failed, they roll back.

# Collation

Defines a set of rules for sorting and comparing data (i.e. case and accent sensitivity).

# Questions

## Clustered vs. non-clustered index

* The rows are physically sorted in the table in that index order – ***“Clustered” table***
* Faster on reads.
* Actual table data is stored in the leaf nodes of the clustered index.
* There can be only one clustered index per table.
* Because insert/update/deletes are slower, you should use clustered on incremental fields (ID, timestamp)
* Non-clustered index – the table rows are stored in a ***heap***

## CTE (Common Table Expression)

* A temporary result set, which is defined during the execution of a DML statement (select/insert/update/delete/create view)
* Available only in memory for the query duration , not stored in DB
* Uses are: group data, recursion, mult references to a single table
* ***Substitute for a view*** when a view is not required 🡪 you do not have to store the definition in metadata
* Similar to a derived table, but a CTE can be self-referenced and can be referenced mult times in the same qry

## Non-clustered

* Rows are not physically sorted on the index.
* Inserts and updates are quicker.
* Can be used multiple times per table.

## Which TCP/IP port does SQL Server run on? How can it be changed?

SQL Server runs on port 1433. It can be changed from the Network Utility TCP/IP properties.

## What are the different index configurations a table can have?

No indexes, one clustered, one clustered and many non-clustered, one non-clust, many non-clust

## What are different types of Collation Sensitivity?

Case sensitivity

Accent sensitivity – words with accents will match (resume = resume’ )

Char sets UTF-8, UTF-16, for example, specify how characters are mapped to numerical values

## What is OLTP (Online Transaction Processing)?

* High volume of short, on-line transactions
* Emphasis on fast queries, data integrity, multi-access environment
* Measured by the number of transactions per second
* Uses data modelling and rules of data normalization to ensure data integrity.

## What is OLAP (Online Analytical Processing)

* Low volume of transactions
* Complex queries involving data aggregation
* Used for data mining
* Historical data stored in multi-dimensional schemas (star schema)

## What's the difference between a primary key and a unique key?

* They both enforce uniqueness on a column.
* Only one pkey per table, but can be many unique keys per table
* Primary key creates a clustered by default, where unique key creates non-clustered by default.
* Primary key allows does NOT allows Nulls; unique key allows one Null

## Truncate vs. Delete

Delete requires a WHERE clause, and also creates entries in the transact log; also slower operation

Truncate removes all rows from the table; it’s faster and uses less system resources and transact log resources

## When is the use of UPDATE\_STATISTICS command?

Updates the indexes on the table after a large operation (many deletions, bulk copy, etc.)

## What is the difference between a HAVING CLAUSE and a WHERE CLAUSE?

* WHERE clause specifies the criteria that each record must meet
* Having used to filter records from a Group By clause
* Where filters on a result, and is applied before a Group By clause.

## What are the properties and different Types of Sub-Queries?

* Sub qry Properties
  + Must be enclosed by parenths
  + Must be put to the right of a comparison operator
  + Cannot contain an Order by
  + Can contain more than one sub qry
* Sub qry Types
  + Single row
  + Multiple row
  + Multiple column

## Sql Agent ?

Allows you to schedule job if the agent is up and running

## \*\*\* What is Log Shipping ?

Automates both database and transaction log backups on a production SQL server, and then restores them onto a standby server.

## What does it mean to have QUOTED\_IDENTIFIER ON? What are the implications of having it OFF?

SET QUOTED\_IDENTIFIER is ON, identifiers (sql key words) can be delimited by double quotation marks, and literals must be delimited by single quotation marks.

## STUFF vs REPLACE functions

Stuff replaces one set of characters (start/end positions), where Replace will replace every occurrence.

## Primary Key vs. Unique Key ?

* A PK constraint is a unique identifier for each row; every table should have a PK constraint
* A Unique Key constraint enforces uniqueness on a set of columns
* A PK automatically has a UNIQUE constraint defined
* Only one PK per table, but many UNIQUE constraints are allowed.

## What is CHECK Constraint?

A CHECK constraint is used to **limit the values** that can be placed in a column. The check constraints are used to enforce domain integrity.

## \*\*\* What are the advantages of using Stored Procedures?

1. Stored procedure ***can reduce network traffic*** and latency, boosting application performance.
2. Stored procedure ***execution plans*** can be reused, staying cached in SQL Server's memory, reducing server overhead.
3. Stored procedures help promote ***code reuse***.
4. Stored procedures can ***encapsulate logic***. You can change stored procedure code without affecting clients.
5. Stored procedures provide better security to your data

# Sql Statements

## Aggregates

Count, Sum, Avg, Min, Max

## Avg

Select ***avg***(emp\_hourly\_wage) as avg\_wage from employees

Select emp\_dept, ***avg***(emp\_hourly\_wage) as avg\_wage

from employees

**Group By** emp\_dept

* Order by an alias:

Select emp\_dept, ***avg***(emp\_hourly\_wage) as avg\_wage

from employees

**Group By** emp\_dept

Order By avg\_wage

* Get highest paid dept

Select ***top 1*** emp\_dept, ***avg***(emp\_hourly\_wage) as avg\_wage

from employees

**Group By** emp\_dept

Order By avg\_wage

* Get top 50%

Select ***top 50 percent*** emp\_dept, ***avg***(emp\_hourly\_wage) as avg\_wage

from employees

**Group By** emp\_dept

Order By avg\_wage

## Column Aliases

Select SalesOrderId, OrderQry ***AS*** Quantity from Sales.SalesOrderDetail

Select SalesOrderId, Quantity ***=*** OrderQry from Sales.SalesOrderDetail

Select SalesOrderId, ***OrderQry Quantity*** from Sales.SalesOrderDetail

(**“AS” is the ANSI std**; the last one is “***accidental alias***”; the user forgot AS or “=”)

## CTE (Common Table Expression)

* A temporary named result set or view
* Similar to a derived table, but a CTE can be self-referenced and referenced mult times

USE AdventureWorks2008R2;

GO

-- **Define the CTE expression** name and column list.

**WITH Sales\_CTE** (SalesPersonID, SalesOrderID, SalesYear)

AS

-- Define the CTE query.

(

SELECT SalesPersonID, SalesOrderID, YEAR(OrderDate) AS SalesYear

FROM Sales.SalesOrderHeader

WHERE SalesPersonID IS NOT NULL

)

-- **Define the outer query** referencing the CTE name.

SELECT SalesPersonID, COUNT(SalesOrderID) AS TotalSales, SalesYear

FROM **Sales\_CTE**

GROUP BY SalesYear, SalesPersonID

ORDER BY SalesPersonID, SalesYear;

GO

* It’s life is limited to the current query (i.e. local/global Temp tables are limited to either current session or all user sessions, respectively).

## Distinct

* Provides a unique list; i.e. removes duplicates from the result
* It goes across the entire row, not for a specific column
* Typically very expensive, esp as the table grows

Ex/ Display a uniq list of StoreIDs

Select **Distinct StoreID** from sales.customer;

## Group By

Associated with aggregate functions

## Having clause

* Applied after the Where clause
* HAVING works on groups, whereas WHERE clause works on indiv rows

Select country from customers

GROUP BY Country

HAVING Count(\*)

Ex/ Get products where price > 15, then ***group the categories*** having more than 5 items in them

(i.e. each product record has a ProductID and a CategoryID, so CategoryID=1 may have many products under this category; CatoryID=2 may only have just 5 products in it)

Select CategoryID , Count(\*) from Products

Where UnitPrice > 15

GROUP BY CategoryID

HAVING Count(\*) > 5

## JOINS

Select vendor\_id, vendor\_name, vendor\_phone, **count(product\_id) as prod\_count**

From vendors ***JOIN*** products

On vendor\_id = product\_vendor\_id

Group By vendor\_id, vendor\_name, vendor\_phone

\*\*\* without Group By, Sql throws an error on the select cols

### Inner Join

* Focuses on the commonality between two tables.
* There must be at least some matching data between two (or more) tables that are being compared.
* Searches tables for matching or overlapping data, then returns the data into one new table.

### Outer Joins - http://www.diffen.com/difference/Inner\_Join\_vs\_Outer\_Join

* Returns both matching and non-matching rows in the other table
  + Left Outer Join (or Left Join)
    - Sometimes nulls will be produced, as some data is shared and some is not
    - Rows are included from the table on the left which doesn’t have corresponding entries in the right table
  + Right Outer Join (or Right Join)
    - Rows from the left table which do not have a matching entry in the right table
  + Full Outer Join (or Full Join)
    - Returns data from both tables, regardless
    - Matches will be duplicated, non-matches return nulls

## Merge

Used to insert/update/delete some target table based on the results of a source table.

Ex/ Updat one or more cols in a table if a matching row exists, or insert a new row if there is no matching row.

\*\*\* in a stored procedure with two params passed in (<https://msdn.microsoft.com/en-us/library/bb510625.aspx> )

MERGE Production.UnitMeasure AS target

USING (SELECT @UnitMeasureCode, @Name) AS source (UnitMeasureCode, Name)

ON (target.UnitMeasureCode = source.UnitMeasureCode)

WHEN MATCHED THEN

UPDATE SET Name = source.Name

WHEN NOT MATCHED THEN

INSERT (UnitMeasureCode, Name)

VALUES (source.UnitMeasureCode, source.Name)

OUTPUT deleted.\*, $action, inserted.\* INTO #MyTempTable;

## Temp Table

* Local temp table - only available to the SQL Server session or connection (means single user) that created the tables. They are auto deleted after session/conn is broke
* Global temp table – Available to all user sessions/connections; delete when all sessions are closed.

## Transaction Isolation Levels

* Read Committed
  + The default isolation level
  + It will read only committed values on a table
  + It waits for pending update transactions (locks) to complete (insert/update/delete)
  + You can use “with(nolock)” at the end of qry to force a dirty read
  + Non-repeatable reads and phantom data are possible between individual statements within the curr transac
* Read Uncommitted
  + Statements can read rows which are modified in another open transact (not committed yet)
* Repeatable Read
  + Shared locks are held on the data until transact completes, no other transac’s can update the table
  + This prevents other transac’s from modifying rows that are read in the current transac
  + Currency is lower because it uses shared locks
  + ***Allows*** other transactions to ***Insert*** data (which can lead to ***phantom reads*** in the curr transact)
* Serializable
  + Cannot read data that has been modified and not committed by another transact
  + No other transact can modify data in the current transac until the current transact completes
  + No other transact can insert data that would fall in the range of keys in the curr transact
  + Range locks are placed in the range of keys that match the current search condition in curr transac
* Snapshot
  + Only recognizes data that was committed prior to the start of the current transac
  + Any other transactions that modify data are NOT visible to the current transact
  + Snapshot transacts do NOT request locks, except for when a database is being recovered

## View

* A virtual table defined by a query. The query is executed when I select from the view.
* The underlying table can be modified from the view.