Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

The **Query Store** in SQL Server is a powerful feature introduced in SQL Server 2016 that helps capture, track, and analyze the performance of queries over time. It stores historical query execution data, which makes it easier to identify and troubleshoot performance issues related to queries, including regressions, long-running queries, and unexpected behavior.

**Key Features of Query Store:**

1. **Capturing Query Performance Data**: Query Store tracks performance-related data, such as execution time, CPU usage, and memory consumption for each query.
2. **Historical Data**: It saves historical execution statistics, allowing you to analyze the performance of queries across different periods.
3. **Plan Forcing**: Query Store allows you to force a specific query plan to be used consistently, even if SQL Server attempts to generate a different, potentially less optimal plan.
4. **Automatic Query Tuning**: Query Store works with automatic tuning features to improve performance by automatically forcing the best plan for queries that show regressions.
5. **Comparing Query Performance Over Time**: You can compare query performance across different periods to see how execution behavior has changed and identify any performance degradation.



**How Query Store Works:**

1. **Data Collection**:
    - Query Store captures query execution data, including the query text, execution plan, and performance metrics (such as execution count, CPU time, IO statistics, etc.).
    - The data is stored in a set of internal tables in the sys schema.
2. **Schema**: The Query Store tables are stored in the system database sys and consist of several key tables:
    - sys.query_store_plan: Stores the execution plans for queries.
    - sys.query_store_query: Stores the queries that have been executed.
    - sys.query_store_runtime_stats: Stores runtime statistics for each query execution.
    - sys.query_store_runtime_stats_interval: Stores performance data for each time interval.
    - sys.query_store_wait_stats: Stores wait statistics for queries.
3. **Data Retention**:
    - The Query Store maintains data for a configurable period, and older data can be purged when the retention limit is exceeded.
    - You can configure the retention period based on your system's needs.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

4. **Query Store Configuration**:
    - Query Store is **enabled** by default for new databases in SQL Server 2016+.
    - It can be configured at the **database level** using the ALTER DATABASE command.

Example:

```
ALTER DATABASE [YourDatabase]
SET QUERY_STORE = ON;
```

5. **Query Plan Forcing**:
    - Query Store allows you to force a specific execution plan to be used for a query, preventing the query from regressing.

Example to force a plan:

```
EXEC sp_query_store_force_plan
@query_id = <query_id>,
@plan_id = <plan_id>;
```

6. **Query Store and Automatic Tuning**:
    - Query Store integrates with SQL Server's **Automatic Tuning** feature, which automatically analyzes the query execution history to detect query performance regressions and can automatically force a good plan or eliminate plan regressions.



**Key Benefits of Using Query Store:**

1. **Performance Troubleshooting**: Query Store can help diagnose performance issues by storing historical data. This is particularly useful when you don't know exactly when a performance problem started.
2. **Plan Regressions**: Query Store helps you detect when SQL Server has chosen a different, less optimal query plan, making it easier to revert to a previous, better plan.
3. **Baseline Performance**: You can establish a baseline of query performance, making it easier to identify deviations from normal performance.
4. **Integration with SQL Server Management Studio (SSMS)**: SSMS provides built-in reports and views to analyze Query Store data visually, making it accessible to DBAs and developers.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

**Query Store Views:**

SQL Server provides several views to analyze Query Store data:

1. **sys.query_store_query**:
   - Contains information about the queries being executed.
   - Columns include query_id, query_text_id, query_hash, etc.
2. **sys.query_store_plan**:
   - Stores the execution plans for the queries.
   - Columns include plan_id, query_id, plan_hash, execution_count, etc.
3. **sys.query_store_runtime_stats**:
   - Contains runtime statistics for each query execution.
   - Columns include query_id, plan_id, avg_duration, total_cpu_time, etc.
4. **sys.query_store_wait_stats**:
   - Contains wait statistics for queries.
   - Columns include query_id, plan_id, wait_type, wait_time_ms, etc.

**Query Store Management and Maintenance**

**Cleaning Data**: Query Store will automatically purge old data based on the retention settings, but you can also manually clean data if needed.

```
ALTER DATABASE [YourDatabase] SET QUERY_STORE CLEAR;
```

**Changing Retention Period**: You can configure the retention period for Query Store data based on your system's needs.

```
ALTER DATABASE [YourDatabase]
SET QUERY_STORE (QUERY_STORE_RETENTION = 90);
```
**Example Usage:**

**Check if Query Store is enabled**:

```
SELECT name, is_query_store_on
FROM sys.databases
WHERE name = 'YourDatabase';
```

**View captured queries**:

```
SELECT query_id, query_text_id, COUNT(*) AS execution_count
FROM sys.query_store_query
GROUP BY query_id, query_text_id
ORDER BY execution_count DESC;
```

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

**View execution plan for a query:**

```
SELECT q.query_id, p.plan_id, p.query_plan
FROM sys.query_store_query AS q
INNER JOIN sys.query_store_plan AS p
  ON q.query_id = p.query_id
WHERE q.query_id = <YourQueryId>;
```

**Force a plan**:

```
EXEC sp_query_store_force_plan
   @query_id = <YourQueryId>,
  @plan_id = <YourPlanId>;
```

**Summary:**

The **Query Store** in SQL Server is a powerful tool for monitoring and optimizing query performance over time. By capturing query execution plans, runtime statistics, and other relevant data, it provides an efficient way to troubleshoot performance issues, identify regressions, and maintain a stable query performance baseline.



Query Store also allows for plan forcing and integrates with automatic tuning to automatically address performance problems.