**What’s the difference between full, differential and transactional back up**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Description** | **Pros** | **When to Use** |
| **Full Backup** | Takes a complete copy of the entire database. | Full restore point. | Used as the base for other backups. |
| **Differential** | Backs up only the changes made since the last full backup. | Faster and smaller than full backup. | To reduce backup time between fulls. |
| **Transaction Log** | Captures all changes since the last transaction log backup. | Allows point-in-time recovery. | For high-availability and precise restore. |

**------------------------------------------------------------------------------------------------------------------------------------**

**What is permission and What’s the difference between grant and deny and**

**Permissions**

They control what users can do in SQL Server (e.g., SELECT, INSERT, DELETE).

|  |  |  |  |
| --- | --- | --- | --- |
| **Command** | **Meaning** | | **When to Use** |
| **GRANT** | Allows a user to perform a specific action. | | When you want to give access rights. |
| **DENY** | Explicitly prevents a user from doing an action, even if inherited. | | To restrict specific access. |
|  | |

**GRANT:** Explicitly allows a user or role to perform a specific action (e.g., GRANT SELECT ON Table1 TO User1 allows User1 to read Table1).

**DENY:** Explicitly prevents a user or role from performing an action, overriding any inherited permissions (e.g., DENY SELECT ON Table1 TO User1 blocks User1 from reading Table1).

**Levels Applied**:

* Server-level (e.g., GRANT CREATE ANY DATABASE TO User1).
* Database-level (e.g., GRANT CONNECT TO User1).
* Schema/Object-level (e.g., GRANT SELECT ON Schema1.Table1 TO User1).

**----------------------------------------------------------------------------------------------------------------------------**

**What’s SQL profiler and when using it**

**A monitoring tool that captures all activities in the SQL Server engine, like:**

* Executed queries
* Login events
* Errors
* Performance bottlenecks

**When to Use**:

* **Performance Tuning**: Identify slow queries or bottlenecks by analyzing execution times and resource usage.
* **Debugging**: Trace errors, deadlocks, or unexpected behavior in applications.
* **Auditing**: Monitor user activity or security-related events (e.g., login attempts).
* **Query Analysis**: Capture queries to optimize indexing or rewrite inefficient code.

**What is trigger and why use it and on what level and what makes it different from normal Stord procedure**

**Trigger**

A piece of SQL code that executes automatically in response to certain table events like: INSERT , UPDATE , EDELETE

**Level**

* Defined at the table level.
* Types:
  + **DML Triggers**: Fire on data manipulation (INSERT, UPDATE, DELETE). Can be AFTER (post-event) or INSTEAD OF (replaces the event).
  + **DDL Triggers**: Fire on schema changes (e.g., CREATE, ALTER, DROP).
  + **Logon Triggers**: Fire on server logon events.

**Why Use**: Automate actions without explicit user calls, ensure consistency (e.g., audit trails, cascading updates).

|  |  |
| --- | --- |
| **Trigger** | **Stored Procedure** |
| Runs **automatically** on events. | Runs **manually** when called. |
| Tied to a specific table or view. | Independent of tables. |
| Usually no parameters. | Accepts input/output parameters. |
| Great for data integrity & logging. | Great for complex logic or batch tasks. |

**Self Study**

**1- The System Schema** (also called System Catalog or Metadata Catalog) is a set of built-in schemas, views, or tables that contain internal information about everything in the database — this is known as metadata.

**In SQL Server:**

* It’s exposed via the **sys schema.**
* **Examples:**
  + **sys.tables →** Lists all user-defined tables.
  + **sys.columns →** Shows all columns in tables.
  + **sys.indexes →** Contains index metadata.
  + **sys.procedures →** Stored procedures info.

**2- Cold Cache:**

* Cold Cache means the cache is empty — the query is being executed for the first time.
* So the engine must perform all four steps from scratch:

|  |  |  |
| --- | --- | --- |
| **Step** | **Happens?** | **Notes** |
| ✅ Parsing | Yes | SQL syntax is checked |
| ✅ Metadata Binding | Yes | Table/column info is fetched from system schema |
| ✅ Query Tree | Yes | Logical plan is built |
| ✅ Execution Plan | Yes | A new plan is created and optimized |

**This is slower**, because everything is processed fresh.

**Procedure Cache (Plan Cache):**

* After the first run, the generated **execution plan is stored** in the **procedure cache**.
* If the **exact same query** runs again:

|  |  |  |
| --- | --- | --- |
| **Step** | **Happens?** | **Notes** |
| ❌ Parsing | No | Skipped — query is already known |
| ❌ Metadata Binding | No | Already resolved previously |
| ❌ Query Tree | No | Already stored in the plan |
| ✅ Execution Plan | Yes | The saved plan is reused from the cache |

**Much faster**, because the engine skips all the planning steps.

**What is the Procedure Cache?**

* It's a part of memory where the database stores execution plans.
* If the same exact query runs again, it reuses the saved plan.
* This saves time and improves performance.

**What is Cold Cache?**

* It means the cache is empty or not ready.
* This happens:
  + First time the query runs.
  + After restarting the database.
  + After clearing the cache.
  + If the query is written differently (even small changes can break the cache).

**3- What is an Indexed View?**

An **Indexed View** is a special type of view that **stores the result set** physically on disk **using a unique clustered index**.

Unlike a standard view (which just stores the SQL definition), an indexed view **materializes the data**, meaning:

* The data is saved and maintained just like a real table.
* It can be used to improve **performance** on complex joins, aggregations, and calculations.

**Benefits of Indexed Views**

1. **Faster performance**
   * Because data is precomputed and stored, SELECTs can be faster.
2. **Useful in aggregation-heavy queries**
   * Example: SUM(), COUNT(), GROUP BY — especially on big data.
3. **Can be used like a table**
   * Once indexed, the view behaves like a real physical object.

create view studentcountperdept

with encryption

as

select dept\_id, count(\*) as numofstudents

from student

group by dept\_id

create unique clustered index ix\_studentcountperdept

on studentcountperdept (dept\_id)

select \* from studentcountperdept