Comparison of GenerationType Strategies in Hibernate

1. GenerationType.IDENTITY

Uses database identity/auto-increment columns.

Database generates the ID after the INSERT.

Supported by: MySQL, SQL Server, DB2.

✅ **Advantages**:

Very simple to use.

Relies on database auto-increment.

❌ **Disadvantages**:

Hibernate does not know the ID before insert.

No efficient batch inserts.

2. GenerationType.SEQUENCE

Uses a database sequence object to generate unique IDs.

Hibernate fetches the ID from the sequence **before** the INSERT.

Supported by: Oracle, PostgreSQL, H2.

✅ **Advantages**:

Hibernate knows ID before insert.

Supports batch inserts.

Very efficient in databases with sequence support.

❌ **Disadvantages**:

Only works if the database supports sequences.

3. GenerationType.TABLE

Uses a separate table to simulate a sequence.

Hibernate updates and reads from this table to get the next ID.

Supported by: Any database.

✅ **Advantages**:

Works with all databases.

Portable solution.

❌ **Disadvantages**:

Slower (extra read/write operations).

Higher contention under heavy load.

Why do we have IDENTITY, SEQUENCE, and TABLE strategies?

We already know they all generate the **ID automatically** instead of writing it manually.
But the reason why we have **three different strategies** is:

1️⃣ Database differences

Not all databases support the same mechanism.

MySQL, SQL Server → support **Auto Increment → IDENTITY**.

Oracle, PostgreSQL → support **Sequences → SEQUENCE**.

Older or limited databases → may support neither Identity nor Sequence → so we use **TABLE** as a fallback.

2️⃣ Performance considerations

SEQUENCE is faster than **IDENTITY**, because Hibernate gets the ID **before the insert** → this allows **batch inserts**.

IDENTITY is slower for batch inserts, since Hibernate must wait for each insert to return the generated ID.

TABLE is the slowest, because it requires an extra read + update in a separate table every time.

3️⃣ Portability

- If you want your application to work on **any database** (without knowing in advance which one), **TABLE** provides a **database-independent solution**.
- But if you know the target DB, you should pick the most efficient option (IDENTITY or SEQUENCE).