Hibernate ID Generation Strategies

1. IDENTITY

Uses the database's auto-increment feature. The ID value is generated by the database when a record is inserted. Hibernate retrieves this value after the INSERT operation. It relies on database-native identity columns.

Advantages:

- Simple and efficient for databases supporting auto-increment (MySQL, SQL Server).
- Requires minimal configuration.

Disadvantages:

- No batch inserts, as Hibernate must wait for each ID to be generated.
- Not portable across all databases.

2. SEQUENCE

Uses a database sequence object to generate identifiers. Hibernate calls the sequence to get the next value before performing an INSERT.

Advantages:

- Highly efficient for databases that support sequences (Oracle, PostgreSQL).
- Allows prefetching of IDs and supports batch inserts.

Disadvantages:

- Not supported by databases lacking sequence objects (e.g., MySQL).

3. TABLE

Creates a separate table to simulate a sequence. The table stores and updates the last used ID value. This strategy works in any database.

Advantages:

- Works with all databases regardless of native ID support.

Disadvantages:

- Slower because it requires extra SELECT and UPDATE operations on the table.
- Possible contention under high concurrency.

4. AUTO

Lets Hibernate choose the most appropriate generation strategy based on the database dialect.

Advantages:

- Simplifies configuration and adapts automatically to the target database.

Disadvantages:

- Behavior may differ between databases, causing inconsistencies when switching.

Comparison Table:

Strategy	How It Works	Best For	Main Drawback
IDENTITY	Uses database auto-increment column	MySQL / SQL Server	No batch inserts
SEQUENCE	Uses database sequence object	Oracle / PostgreSQL	Not supported in all DBs
TABLE	Uses a table to store next ID	Any database	Slower due to extra queries
AUTO	Hibernate selects appropriate strategy	All databases	Inconsistent between DBs