📘 **Revision Summary – Persistence Context, @Transactional, and Relationships**

✅ **Persistence Context**

* It’s like Hibernate’s **first-level cache** where entities are tracked during a transaction.
* When you fetch an entity, it’s stored in the persistence context.
* Any changes made to that entity are tracked (Dirty Checking).
* At transaction commit → Hibernate compares the entity with its snapshot and automatically generates SQL (INSERT, UPDATE, DELETE).

✅ **@Transactional Annotation**

* Declares a method/class to run inside a **transaction**.
* Ensures **all DB operations** inside it happen as a single unit (commit or rollback).
* Triggers **dirty checking + flush** at the end.
* Without @Transactional, updates may not persist unless you explicitly save.

✅ **Relationships in JPA/Hibernate**

* **One-to-One (1:1):** One entity has exactly one related entity.
* **One-to-Many (1:N):** A parent has many children (e.g., Patient → Appointments).
* **Many-to-One (N:1):** Many children belong to one parent (inverse of 1:N).
* **Many-to-Many (M:N):** Entities on both sides can have multiple relationships (needs a join table).

🔑 **Important Points:**

* **Owning side:** Holds the foreign key (controls updates).
* **Inverse side:** Just a mirror, cannot update FK.
* **Parent–Child lifecycle:** Parent can cascade operations (e.g., delete patient → delete appointments).