**JDBC:**

1. Use of JDBC API : This helps to talk to a database. To query from a database
2. JDBC Drivers : JDBC-ODBC Driver, Native API, Network Protocol, Pure Java Driver
3. Execute will return Boolean. ExecuteQuery will return resultset. ExceuteUpdate will return an integer
4. **Statement** – Useful when you are using static SQL statements at run time and it cannot accept parameters

**PreparedStatement** – Use when you plan to use the SQL statements many times and it accepts input at run time. Another benefit of a PreparedStatement is that the SQL itself is pre-compiled a single time and then retained in memory by the system, rather than being compiled each and every time the statement is called. This allows for faster execution. Query Plan is created only once irrespective of how many times the query is executed

**CallableStatement** – Use when you want to access the database stored procedures and can accept runtime input parameters.

1. ResultSet - Result that is returned on executing a query. Something that is returned from a executeQuery
2. SetFetchSize() and setMaxRows() in Statement - setFetchSize(int) defines the number of rows that will be read from the database when the ResultSet needs more rows.

setMaxRows(int) sets the limit of the maximum number of rows in a ResultSet object.

1. JDBCBatchProcessing- Execute batch of sql statements
2. Transaction Management –
3. JDBC savepoint
4. Datasource – On app server, we configure datasource. Application will use the JNDI name to connect to the db.
5. CLOB(Character Large Object) & BLOB(Binary Large Object) – Datatypes to store images, files in db. CLOB deals mainly with text. BLOB deals with images
6. JDBC Best Practices – Use connection pooling

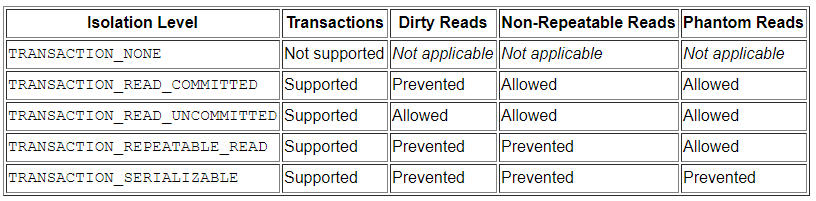
**Transaction Management:**

*Atomicity* – Partial updates are not allowed. Either ALL or NONE

*Consistency*- The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

*Isolation* – In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

*Durability* – The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.



*Problems with Transaction management:*

Dirty Reads : Reading a data that has not been committed. To avoid dirty reads, you can use locking

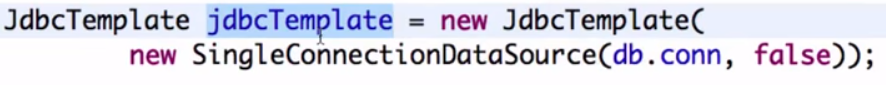
Non-Repeatable Reads : Data read twice inside the same transaction cannot be guaranteed to contain the same value.

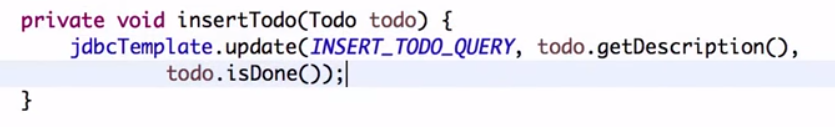
Phantom Reads: During a transaction, new records are added or deleted by another transaction to the records being read

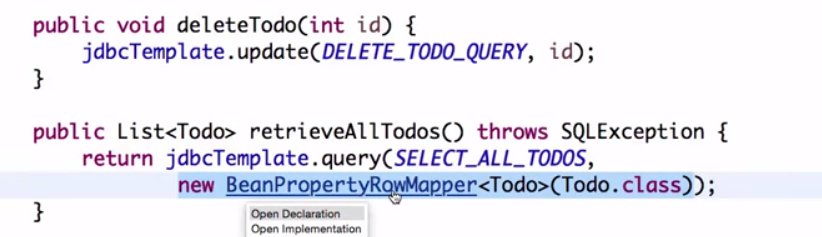
To prevent the dirty reads, non repeatable reads, phantom reads, we set the isolation levels in the program

In REPEATABLE\_READ, read lock is obtained. In Serializable, read lock, write lock and range lock are obtained

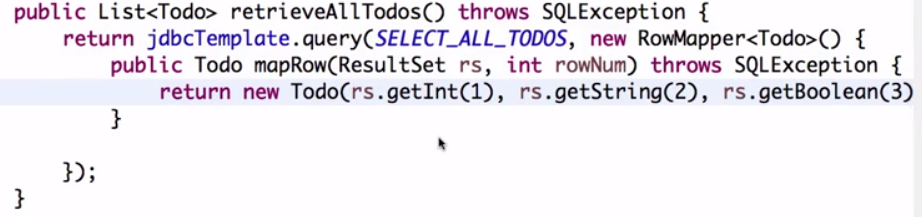
**Spring JDBC:**





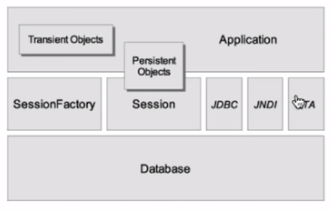


The above method uses In built mapper class to map the result set. Use Custom mapper, if the db fields and the java class fields are not the same



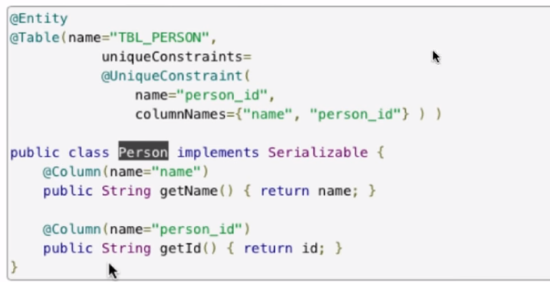
**Hibernate**

1. **JPA –** Java Persistance API. Specification to map Java Object to Db Table. Hibernate is the implementation of JPA. Hibernate is the ORM framework.
2. **Hibernate Architecture**

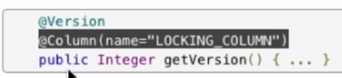


Session Factory -Its just a cache to store all the mapping files

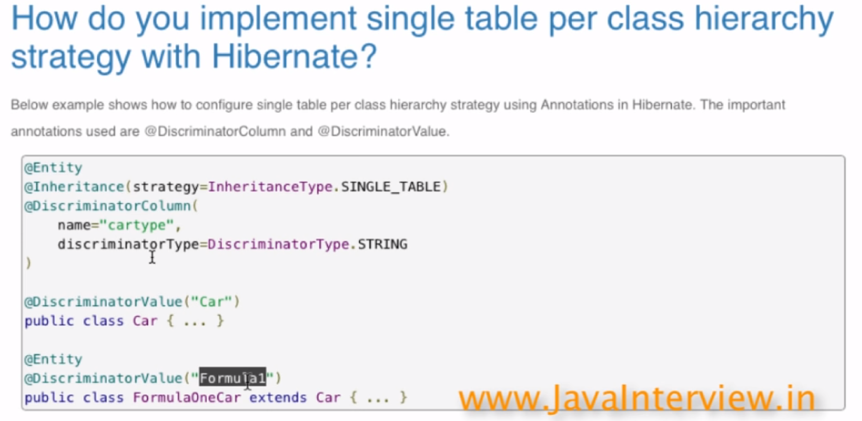
Session- Represents a conversation between Java Application and Persistent store



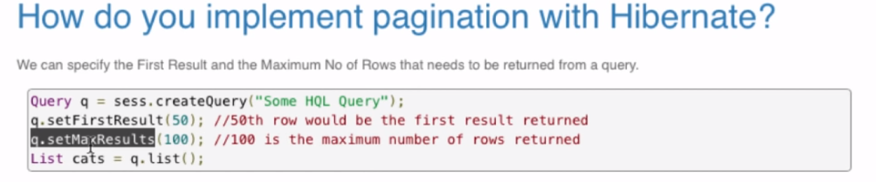
1. @Immutable – Content of the particular object will not be updated to the respective database table
2. Unique Identifier Generator – IDENTITY, SEQUENCE, CUSTOM GENERATOR, AUTO
3. Optimistic Locking – If you lock the table for a long time, the other queries cannot access the table which will result in slow down of the application. To improve the performance of the application, you don’t really hold the lock. You just have the cache with the old data. Hibernate uses @Version or timestamp annotation



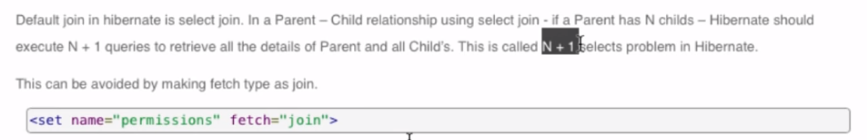
1. Single Table per class hierarchy – You will save the parent class and child class objects in the same table in the database using @Inheritance annotation in the class level with its strategy as Inheritancetype.SINGLE\_TABLE and differentiating the child class with the @DiscriminatorValue annotation. @DiscriminatorColumn annotation should be at the class level



1. Table per class hierarchy - You will save the parent class and child class objects in the different tables in the database using @Inheritance annotation in the class level with its strategy as Inheritancetype.TABLE\_PER\_CLASS
2. One to many Unidirectional Relationship – You basically needs a relationship from parent to child and not from child to parent. This is done using @OnetoMany annotation
3. One to many Bidirectional Relationship – You will have a relationship from parent to child and from child to parent. This is done using @OnetoMany annotation in the parent object and @ManytoOne in the child object
4. Pagination



1. Lazy Association – If you don’t want all the children table to be loaded when the parent table is loaded, you set the association as lazy.
2. N+1 selects problem –



1. Automatic schema generation from Hibernate Mappings – This is done using a tool SchemaExport
2. SchemaValidator – Tool to verify if the mapping configured matches the existing database structure
3. Best Practises –
   1. Use Natural keys instead of using sequences
   2. Place each class mapping in its own file
   3. Use bind variables if use query
   4. Prefer lazy fetch for associations
   5. Use bidirectional relationship
   6. Avoid N+1 selects problem