Hibernate Query Language (HQL):

---

1. Introduction to HQL

Hibernate Query Language (HQL) is an object-oriented query language similar to SQL but tailored for Hibernate ORM. Unlike SQL, which operates directly on database tables, HQL interacts with Java objects (entities). This abstraction allows HQL to be database-agnostic and provides a more flexible and powerful way to query, manipulate, and manage data in Hibernate.

Key Features of HQL:

- **Object-oriented**: Operates on persistent objects, not directly on tables and columns.
- **Database Independent**: Provides a level of abstraction over the SQL dialects of different databases.
- **Supports Named Queries**: Predefined queries that can be reused.
- **Supports Parameter Binding**: Helps prevent SQL injection by using named or positional parameters.

---

2. Basic HQL Syntax

HQL syntax is similar to SQL but uses entity names and attributes instead of table names and columns. Here are some basic operations:

General HQL Syntax:

SELECT [field_list] FROM EntityName [alias] WHERE [conditions]

Examples:

```
FROM Item
SELECT i FROM Item i WHERE i.price > 1000
UPDATE Item SET itemName = 'Refrigerator' WHERE id = 1
DELETE FROM Item WHERE id = 5
```

Item.java

・The Item class is annotated with **@Entity**, indicating it is a **JPA entity**.
・The **@NamedQueries** annotation is used to define reusable named HQL queries:

- **fetchItems**: Retrieves all items.
- **fetchItemsById**: Retrieves an item based on its ID.

・the attributes **id, itemName, and price represent** columns in the database

Advantages of HQL

- **Database Independence**: HQL abstracts SQL specifics, allowing for easier database migration.
- **Object-Oriented**: Works directly with entity objects, aligning with Java's OOP principles.
- **Flexibility and Reusability**: Named queries provide a way to reuse common query logic.

6. Limitations of HQL

- **Learning Curve**: Understanding and mastering HQL syntax may require time, especially when migrating from SQL.
- **Complex Queries**: For complex queries or specific database functions, **native SQL might still be preferred**.

3. Summary of HQL Clauses and Functions

Clause	Description	Example
SELECT	Specifies the fields to retrieve.	SELECT i.itemName FROM Item i
FROM	Specifies the entity to query.	FROM Item
WHERE	Filters the results based on conditions.	WHERE i.price > 20000
ORDER BY	Sorts the results.	ORDER BY i.price DESC
GROUP BY	Groups the results by specified fields.	GROUP BY i.itemName
HAVING	Filters grouped results.	HAVING COUNT(i.id) > 1
JOIN	Joins related entities.	JOIN i.category c
COUNT	Returns the count of records.	SELECT COUNT(i.id) FROM Item i
AVG	Returns the average value.	SELECT AVG(i.price) FROM Item i
SUM	Returns the sum of values.	SELECT SUM(i.price) FROM Item i
LIKE	Searches for a specified pattern.	WHERE i.itemName LIKE 'Fridge%'
IN	Checks if a value is within a set of values.	WHERE i.itemName IN ('Fridge', 'Washing Machine')
BETWEEN	Checks if a value is within a range.	WHERE i.price BETWEEN 10000 AND 50000

Conclusion

HQL provides a high-level abstraction over SQL for querying objects in Hibernate, allowing developers to write more readable and maintainable queries using object-oriented principles..

Using HQL effectively can lead to cleaner, more robust data access code in Hibernate applications.