Lecture 21: Fetching with Filters and Specific Properties

Theory

HQL supports advanced filtering capabilities and selective property retrieval, allowing for optimised database queries and reduced memory usage.

Filtered Queries with Parameters

Basic Filtering Implementation:

```
Session session = sf.openSession();

// Parameterized query for security and reusability
String brand = "Asus";
Query query = session.createQuery("from Laptop where brand like ?1");
query.setParameter(1, brand);
List<Laptop> laptops = query.getResultList();

System.out.println(laptops);

session.close();
```

Selective Property Retrieval

Fetching Specific Columns:

```
Session session = sf.openSession();

String brand = "Asus";
Query query = session.createQuery("select model from Laptop where brand like ?1");
query.setParameter(1, brand);
List<String> laptops = query.getResultList();

System.out.println(laptops);

session.close();
```

Multiple Properties Selection:

```
String brand = "Asus";
Query query = session.createQuery("select brand, model from Laptop where brand like ?1");
query.setParameter(1, brand);
List<Object[]> laptops = query.getResultList();

// Processing multiple properties
for (Object[] data : laptops) {
    System.out.println((String)data[0] + " " + (String)data[1]);

}
```

TELUSKO

Key Features

Feature	Description	Example
Parameter Binding	Secure parameter substitution	`?1`, `?2` for positional
Selective Fetching	Retrieve only needed columns	`select model from Laptop`
Multiple Properties	Fetch multiple specific fields	`select brand, model from Laptop`
Type Safety	Return appropriate data types	`List<String>` vs `List<Object[]>`

Benefits

- **Performance**: Fetch only required data
- **Security**: Prevents SQL injection attacks
- **Memory Efficiency**: Reduced object overhead
- **Network Optimization**: Less data transfer

TELUSKO