CURSO DE HIBERNATE 5

OpenWebinars

HIBERNATE

(1) INTRODUCCION

Persistencia, desfase objecto-relacional, ORM. Productos y estándares

(2) HIBERNATE

Más que un ORM. Comparativa con otros productos. JPA. Maven. Módulos

(3) PRIMER PROYECTO

Hibernate.cfg.xml, EntityManager y persistence.xml

(4) ENTIDADES

Definición del modelo del dominio. Entidades y ciclo de vida. XML y anotaciones. Tipos de datos.

(5) ASOCIACIONES

ManyToOne, OneToMany, OneToOne, ManyToMany

HIBERNATE

(6) ELEMENTOS AVANZADOS

Campos calculados, herencia.

(7) COLECCIONES

Mapeo de colecciones. Tipos (list, set, map). Colecciones ordenadas (sorted vs. ordered).

(8) GENERACION DEL ESQUEMA

Customización del proceso de generación del esquema.

(9) CONTEXTO DE PERSISTENCIA

Almacenamiento, recuperación y borrado de entidades.

(10) TRANSACCIONES

Control de concurrencia. Patrones y antipatrones.

HIBERNATE

(12)
ENVERS

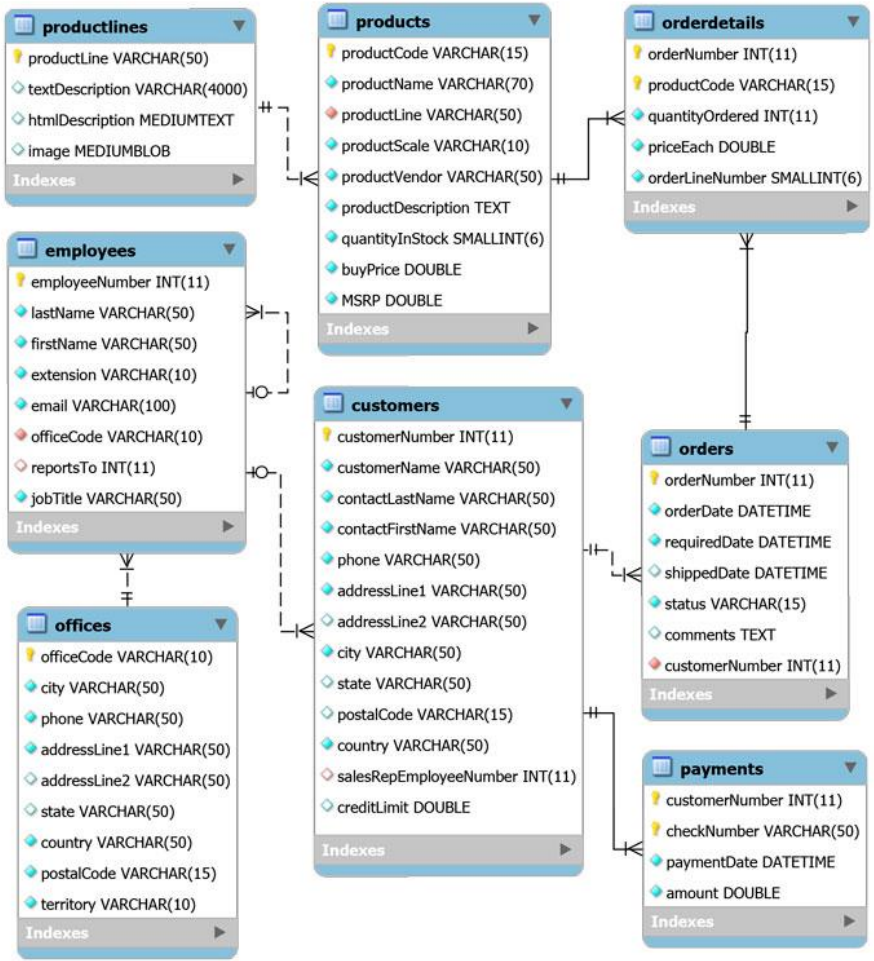
Introducción a la
auditoria de entidades.

(11)
CONSULTAS
HQL VS JPQL

Consultas con
parámetros,
Anotaciones. SQL nativo

1.

BASE DE DATOS DE EJEMPLO


productlines
productLine VARCHAR(50)
textDescription VARCHAR(4000)
htmlDescription MEDIUMTEXT
image MEDIUMBLOB
Indexes
products
productCode VARCHAR(15)
productName VARCHAR(70)
productLine VARCHAR(50)
productScale VARCHAR(10)
productVendor VARCHAR(50)
productDescription TEXT
quantityInStock SMALLINT(6)
buyPrice DOUBLE
MSRP DOUBLE
Indexes
orderdetails
orderNumber INT(11)
productCode VARCHAR(15)
quantityOrdered INT(11)
priceEach DOUBLE
orderLineNumber SMALLINT(6)
Indexes
employees
employeeNumber INT(11)
lastName VARCHAR(50)
firstName VARCHAR(50)
extension VARCHAR(10)
email VARCHAR(100)
officeCode VARCHAR(10)
reportsTo INT(11)
jobTitle VARCHAR(50)
Indexes
customers
customerNumber INT(11)
customerName VARCHAR(50)
contactLastName VARCHAR(50)
contactFirstName VARCHAR(50)
phone VARCHAR(50)
addressLine1 VARCHAR(50)
addressLine2 VARCHAR(50)
city VARCHAR(50)
state VARCHAR(50)
postalCode VARCHAR(15)
country VARCHAR(50)
salesRepEmployeeNumber INT(11)
creditLimit DOUBLE
Indexes
orders
orderNumber INT(11)
orderDate DATETIME
requiredDate DATETIME
shippedDate DATETIME
status VARCHAR(15)
comments TEXT
customerNumber INT(11)
Indexes
offices
officeCode VARCHAR(10)
city VARCHAR(50)
phone VARCHAR(50)
addressLine1 VARCHAR(50)
addressLine2 VARCHAR(50)
state VARCHAR(50)
country VARCHAR(50)
postalCode VARCHAR(15)
territory VARCHAR(10)
Indexes
payments
customerNumber INT(11)
checkNumber VARCHAR(50)
paymentDate DATETIME
amount DOUBLE
Indexes

2.

JPQL BÁSICO

CONSULTAS SELECT

```
SELECT c FROM Customer c
```

```
SELECT c.customerName FROM Customer c
```

```
SELECT e.propiedad.anidada.masanidada FROM Entidad e
```

```
SELECT c.customerName, c.city, c.country FROM Customer c
```

CONSULTAS SELECT CON PARÁMETROS

```
SELECT e FROM Employee e WHERE e.jobTitle = :jobTitle
```

```
SELECT o FROM Order o WHERE o.orderDate BETWEEN ?1 AND ?2 AND status = ?3
```

CONSULTAS SELECT CON SALIDA ORDENADA

```
SELECT o FROM Order o ORDER BY o.orderDate DESC
```

3.

QUERY y TYPEDQUERY

QUERY

```
Query query = em.createQuery(
    "select c from Customer c"
);

List<Customer> listCustomer = (List<Customer>) query.getResultList();
```

QUERY

```
Query query = em.createQuery(
  "SELECT e FROM Employee e WHERE e.jobTitle = :jobTitle"
);

query.setParameter("jobTitle", "Sales Rep");
```

```
Query query = em.createQuery(
"SELECT o FROM Order o WHERE o.orderDate BETWEEN ?1 AND ?2 AND status = ?3 ORDER BY o.orderDate DESC"
);

Calendar calendar = Calendar.getInstance();
calendar.set(2003, 0, 1);
query.setParameter(1, calendar.getTime());

calendar.set(2003, 5, 30);
query.setParameter(2, calendar.getTime());

query.setParameter(3, "Shipped");
```

4.

JOINs

JOIN EXPLÍCITOS (con *FETCH*)

```
Query query = em.createQuery(
    "select c "
  + "from Customer c "
  + "left join fetch c.employee "
);
```

JOIN IMPLÍCITOS

```
TypedQuery<Customer> query = em.createQuery(
    "select c "
  + "from Customer c "
  + "where c.employee = :employee", Customer.class);

query.setParameter("employee", em.find(Employee.class, 1370));
```

5.

CONSULTAS DML

CONSULTAS DE ACTUALIZACIÓN

```
UPDATE Entity e
SET e.prop1 = newValue1, e.prop2 = newValue2, ...
WHERE ...
```

```
DELETE FROM Entity e
WHERE ...
```

```java
em.getTransaction().begin();

//UPDATE

//Incremento del 10% en el límite de crédito a todos los clientes
int numUpdateResults = em.createQuery(
    "update Customer c "
    + "set c.creditLimit = c.creditLimit * 1.1")
    .executeUpdate();
System.out.println("Número de registros afectados: " + numUpdateResults);

//DELETE
```

~~JPQL~~ HQL

```java
int insertedEntities = session.createQuery(
    "insert into Partner (id, name) " +
    "select p.id, p.name " +
    "from Person p ")
.executeUpdate();
```

```java
Date date = null;
try {
date = new SimpleDateFormat("dd/MM/yyyy").parse("06/06/2003");

  int numDeleteResults = em.createQuery(
      "delete from Payment p "
      + "where p.paymentDate = :fecha")
      .setParameter("fecha", date)
      .executeUpdate();

  System.out.println("Número de registros afectados: " + numDeleteResults);


} catch (ParseException e) {
System.err.println("Error en el parseo de la fecha");
}

em.getTransaction().commit();
```

6.

CONSULTAS CON NOMBRE

CONSULTAS CON NOMBRE

```java
@Entity
@Table(name="customers")
@NamedQuery(name="Customer.findAll", query="SELECT c FROM Customer c")
public class Customer implements Serializable {
    //Resto del código
}
```

```java
Query query = em.createNamedQuery("Customer.findAll");
```

7.

CONSULTAS CON SQL NATIVO

CONSULTAS ESCALARES

```
List<Object[]> customers = entityManager.createNativeQuery(
    "SELECT * FROM customers" )
.getResultList();
```

```
List<Object[]> employees = entityManager.createNativeQuery(
    "SELECT employeeNumber, firstName, lastName FROM employees" )
.getResultList();

for(Object[] employee : employees) {
    Number employeeNumber = (Number) employee[0];
    String firstName = (String) employee[1];
    String lastName = (String) employee[2];
}
```

CONSULTAS DE ENTIDADES

```
List<Customer> customers = entityManager.createNativeQuery(
    "SELECT * FROM customers", Customer.class )
.getResultList();
```

CONSULTAS DE ASOCIACIONES

```
List<Employee> employeesList = em.createNativeQuery(
                "SELECT employeeNumber, lastName, firstName, extension, email, officeCode, reportsTo, jobTitle FROM employees",
                Employee.class).getResultList();
```

CONSULTAS CON NOMBRE

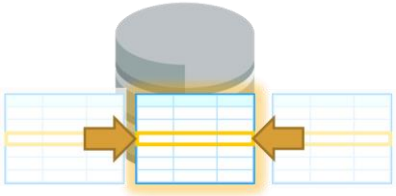
```java
@Entity
@Table(name="employees")
@NamedQuery(name="Employee.findAll", query="SELECT e FROM Employee e")
@NamedNativeQuery(name="Employee.nativeFindAll", query="SELECT * FROM employees", resultClass=Employee.class)
public class Employee implements Serializable {
  //Resto de código
}
```

```java
List<Employee> employeesList = em.createNamedQuery("Employee.nativeFindAll").getResultList();
```

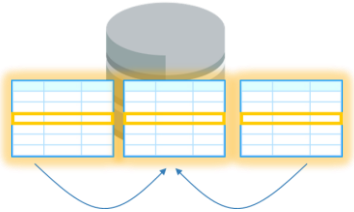
8.

CONSULTAS CON HERENCIA

HERENCIA



SINGLE TABLE



JOINED

TABLE PER CLASS