IT-Academy

Hibernate
Отображение отношений

Автор: Юлий Слабко

Id as a component type

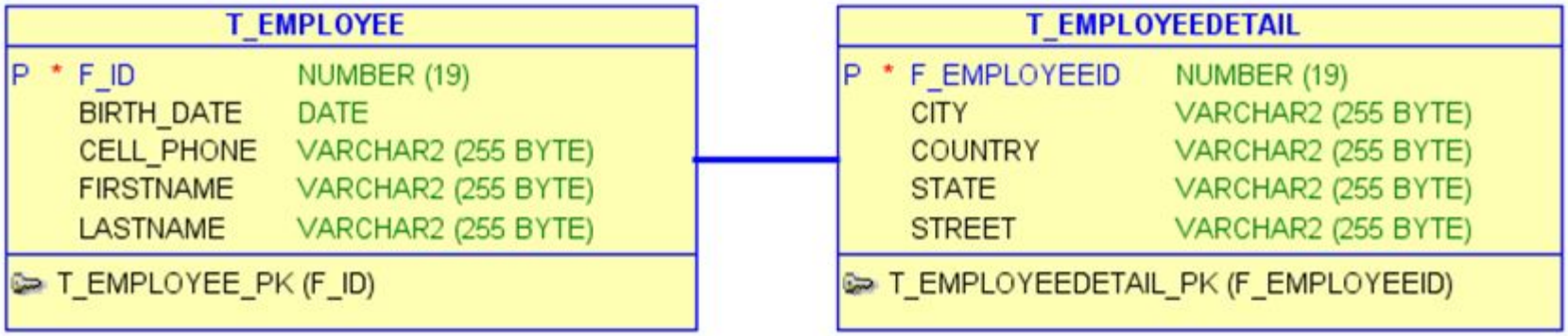
IT-Academy

```
@Entity
class User {
   @EmbeddedId
   @AttributeOverride(name="firstName", column=@Column(name="fld_firstname")
   UserId id;

   Integer age;
}

@Embeddable
class UserId implements Serializable {
   String firstName;
   String lastName;
}
```

One-to-One


T_EMPLOYEE
P * F_ID NUMBER (19)
BIRTH_DATE DATE
CELL_PHONE VARCHAR2 (255 BYTE)
FIRSTNAME VARCHAR2 (255 BYTE)
LASTNAME VARCHAR2 (255 BYTE)
T_EMPLOYEE_PK (F_ID)
T_EMPLOYEEDETAIL
P * F_EMPLOYEEID NUMBER (19)
CITY VARCHAR2 (255 BYTE)
COUNTRY VARCHAR2 (255 BYTE)
STATE VARCHAR2 (255 BYTE)
STREET VARCHAR2 (255 BYTE)
T_EMPLOYEEDETAIL_PK (F_EMPLOYEEID)

One-to-One

IT-Academy

```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="EMPLOYEE")
public class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = IDENTITY)
    @Column(name = "EMPLOYEE_ID", unique = true)
    @Access(AccessType.PROPERTY)
    private Long employeeId;

    @Column(name = "FIRST_NAME")
    private String firstname;

    @Column(name = "LAST_NAME")
    private String lastname;

    @Temporal(TemporalType.TIMESTAMP)
    private Date date;

    @OneToOne(fetch = FetchType.LAZY, mappedBy = "employee", cascade = CascadeType.ALL)
    @Access(AccessType.PROPERTY)
    private EmployeeDetail employeeDetail;
}
```

One-to-One

IT-Academy

```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "EMPLOYEE_DETAILS")
public class EmployeeDetail implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GenericGenerator(name = "one-one",
            strategy = "foreign",
            parameters = @Parameter(name = "property", value = "employee"))
    @GeneratedValue(generator = "one-one")
    @Column(name = "EMPLOYEE_ID")
    @Access(AccessType.PROPERTY)
    private Long id;

    @Column(name = "STREET")
    private String street;

    @Column(name = "CITY")
    private String city;

    @Column(name = "STATE")
    private String state;

    @Column(name = "COUNTRY")
    private String country;

    @OneToOne(fetch = FetchType.LAZY)
    @PrimaryKeyJoinColumn
    @Access(AccessType.PROPERTY)
    private Employee employee;
}
```

One-to-One

IT-Academy

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://127.0.0.1:3306/one_one</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">pwd</property>
        <property name="hibernate.connection.pool_size">10</property>
        <property name="hibernate.connection.isolation">2</property>
        <property name="show_sql">true</property>
        <property name="hibernate.hbm2ddl.auto">create</property>
        <!--Mapping-->
        <mapping class="by.academy.it.pojos.Employee"/>
        <mapping class="by.academy.it.pojos.EmployeeDetail"/>
    </session-factory>
</hibernate-configuration>
```

One-to-One

IT-Academy

```
public static void main(String... args) throws Exception {
    EmployeeDao employeeDao = new EmployeeDao();
    Session session = employeeDao.getSession();
    session.beginTransaction();
    Employee e = new Employee(null, "Yuli", "Slabko", new Date(), null);
    EmployeeDetail employeeDetail = new EmployeeDetail(null, "Mira", "Minsk", "", "Blr", null);
    e.setEmployeeDetail(employeeDetail);
    employeeDetail.setEmployee(e);
    session.save(e);
    session.getTransaction().commit();
}
```

```
Hibernate: insert into EMPLOYEE (date, FIRST_NAME, LAST_NAME) values (?, ?, ?)
Hibernate: insert into EMPLOYEE_DETAILS (CITY, COUNTRY, STATE, STREET, EMPLOYEE_ID) values (?, ?, ?, ?, ?)
```

One-to-One. Cascade

IT-Academy

```
public static void main(String... args) throws Exception {
    EmployeeDao employeeDao = new EmployeeDao();
    Session session = employeeDao.getSession();
    session.beginTransaction();
    Employee e = new Employee(null, "Yuli", "Slabko", new Date(), null);
    EmployeeDetail employeeDetail = new EmployeeDetail(null, "Mira", "Minsk", "", "Blr", null);
    e.setEmployeeDetail(employeeDetail);
    employeeDetail.setEmployee(e);
    session.save(e);
    session.getTransaction().commit();
}
```

```
@OneToOne(fetch = FetchType.LAZY, mappedBy = "employee", cascade = CascadeType.ALL)
@Access(AccessType.PROPERTY)
private EmployeeDetail employeeDetail;
```

```
Hibernate: insert into EMPLOYEE (date, FIRST_NAME, LAST_NAME) values (?, ?, ?)
Hibernate: insert into EMPLOYEE_DETAILS (CITY, COUNTRY, STATE, STREET, EMPLOYEE_ID) values (?, ?, ?, ?, ?)
```

```
@OneToOne(fetch = FetchType.LAZY, mappedBy = "employee")
@Access(AccessType.PROPERTY)
private EmployeeDetail employeeDetail;
```

```
Hibernate: insert into EMPLOYEE (date, FIRST_NAME, LAST_NAME) values (?, ?, ?)
```

One-to-One. Cascade

IT-Academy

```
@OneToOne(fetch = FetchType.LAZY, mappedBy = "employee", cascade = CascadeType.REMOVE)
@Access(AccessType.PROPERTY)
private EmployeeDetail employeeDetail;
```

```
session.beginTransaction();
Employee e = (Employee) session.get(Employee.class, id);
session.delete(e);
session.getTransaction().commit();
```

```
Hibernate: select employee0_.EMPLOYEE_ID as EMPLOYEE1_0_0_, employee
Hibernate: select employeede0_.EMPLOYEE_ID as EMPLOYEE1_1_0_, employ
 employeede0_.EMPLOYEE_ID=?
Hibernate: delete from EMPLOYEE_DETAILS where EMPLOYEE_ID=?
Hibernate: delete from EMPLOYEE where EMPLOYEE_ID=?
```

```
@OneToOne(fetch = FetchType.LAZY, mappedBy = "employee")
@Access(AccessType.PROPERTY)
private EmployeeDetail employeeDetail;
```

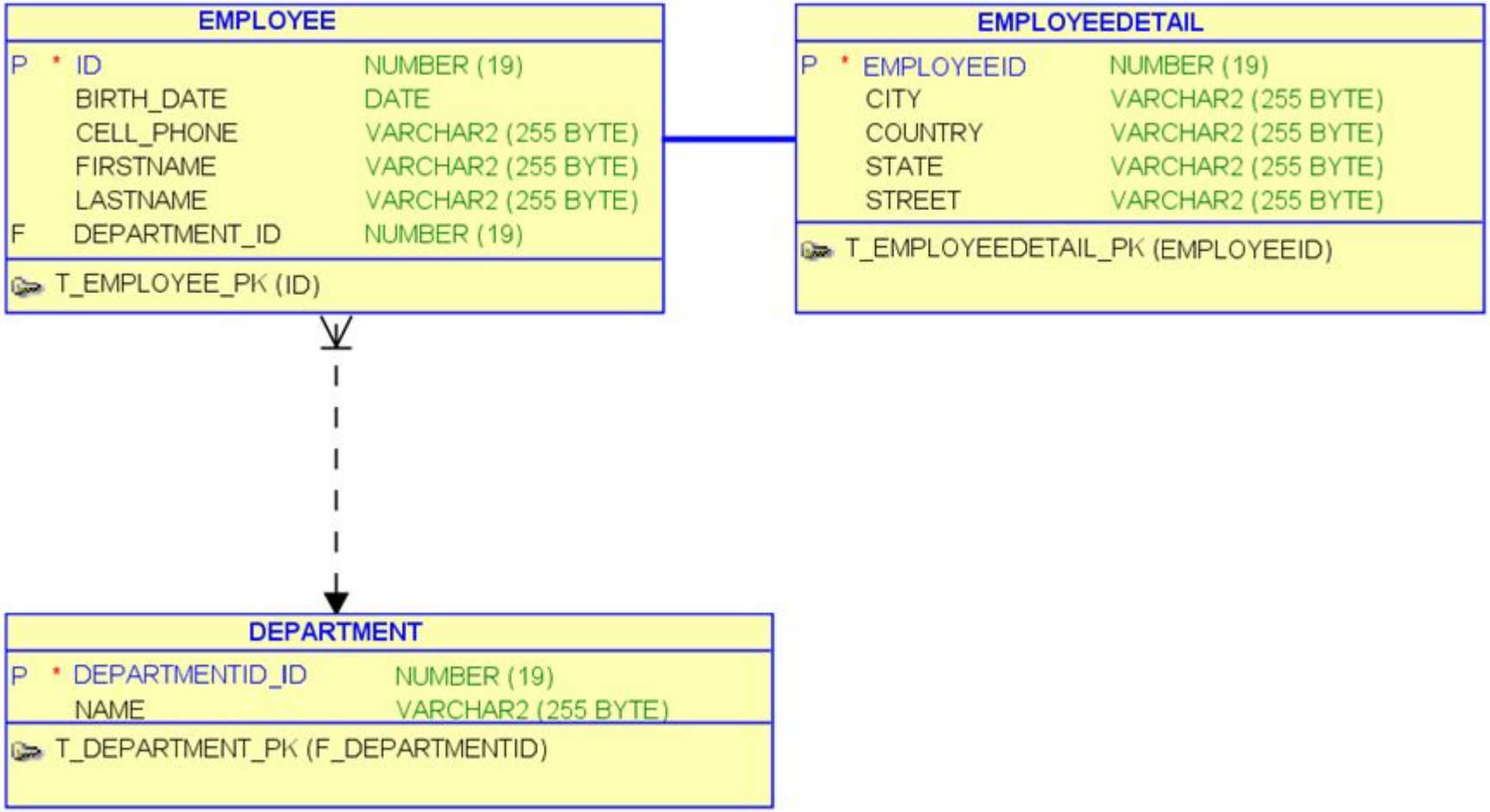
```
session.beginTransaction();
Employee e = (Employee) session.get(Employee.class, id);
session.delete(e);
session.getTransaction().commit();
```

```
Hibernate: select employeede0_.EMPLOYEE_ID as EMPLOYEE1_1_0_,
 employeede0_.EMPLOYEE_ID=?
Hibernate: select employee0_.EMPLOYEE_ID as EMPLOYEE1_0_0_, er
Hibernate: delete from EMPLOYEE_DETAILS where EMPLOYEE_ID=?
```

Вопросы

IT-Academy

One To Many


EMPLOYEE
P * ID NUMBER (19)
BIRTH_DATE DATE
CELL_PHONE VARCHAR2 (255 BYTE)
FIRSTNAME VARCHAR2 (255 BYTE)
LASTNAME VARCHAR2 (255 BYTE)
F DEPARTMENT_ID NUMBER (19)
T_EMPLOYEE_PK (ID)
EMPLOYEEDETAIL
P * EMPLOYEEID NUMBER (19)
CITY VARCHAR2 (255 BYTE)
COUNTRY VARCHAR2 (255 BYTE)
STATE VARCHAR2 (255 BYTE)
STREET VARCHAR2 (255 BYTE)
T_EMPLOYEEDETAIL_PK (EMPLOYEEID)
DEPARTMENT
P * DEPARTMENTID_ID NUMBER (19)
NAME VARCHAR2 (255 BYTE)
T_DEPARTMENT_PK (F_DEPARTMENTID)

One To Many.

IT-Academy

```java
@Data
@NoArgsConstructor
@Entity
@Table(name = "DEPARTMENT")
public class Department implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "DEPARTMENT_ID", unique = true)
    private Long departmentId;

    @Column(name = "NAME")
    private String departmentName;

    @OneToMany(mappedBy = "department")
    private Set<Employee> employees = new HashSet<>();

    public Department(String name) {
        this.departmentName = name;
    }
}
```

One To Many

IT-Academy

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="EMPLOYEE")
public class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = IDENTITY)
    @Column(name = "EMPLOYEE_ID", unique = true)
    @Access(AccessType.PROPERTY)
    private Long employeeId;

    @Column(name = "FIRST_NAME")
    private String firstname;

    @Column(name = "LAST_NAME")
    private String lastname;

    @Temporal(TemporalType.TIMESTAMP)
    private Date date;

    @OneToOne(mappedBy = "employee", cascade = CascadeType.ALL)
    @Access(AccessType.PROPERTY)
    private EmployeeDetail employeeDetail;

    @ManyToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "DEPARTMENT_ID")
    private Department department;
}
```

One To Many

IT-Academy

```java
@Entity
@SequenceGenerator(name = "PK", sequenceName = "t_department_seq")
public class Department implements Serializable {
    private static final long serialVersionUID = 6L;
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "PK")
    private Long departmentId;

    @Column
    private String departmentName;

    @OneToMany(mappedBy = "department")
    private Set<Employee> employees;
```

- **@OneToMany** аннотации определяют многозначную связь с коллекцией сущностей.

One To Many. Results. No save-update cascade.

IT-Academy

```
Session session = util.getSession();
session.beginTransaction();
Department department = new Department("Sales");
Employee e = new Employee(null, "Tom", "Hanks", new Date(), null, department);
session.save(department);
session.save(e);
session.getTransaction().commit();
util.releaseSession();
```

```
Hibernate: insert into DEPARTMENT (NAME) values (?)
Hibernate: insert into EMPLOYEE (date, DEPARTMENT_ID, FIRST_NAME, LAST_NAME) values (?, ?, ?, ?)
```

One To Many. Additions. save-update cascade.

IT-Academy

```
@OneToMany(mappedBy = "department", cascade = {CascadeType.ALL})
private Set<Employee> employees = new HashSet<>(0);
```

```
Session session = util.getSession();
session.beginTransaction();
Department department = new Department("QA");
Employee e = new Employee(null, "Tom", "Hanks", new Date(), null, department);
department.getEmployees().add(e);
e.setDepartment(department);
session.save(department);
session.getTransaction().commit();
util.releaseSession();
```

```
Hibernate: insert into DEPARTMENT (NAME) values (?)
Hibernate: insert into EMPLOYEE (date, DEPARTMENT_ID, FIRST_NAME, LAST_NAME) values (?, ?, ?, ?)
```

One To Many. Additions. Delete-orphan cascade.

IT-Academy

```
@OneToMany(mappedBy = "department", orphanRemoval=true)
private Set<Employee> employees = new HashSet<>(0);
```

```
private static void deleteOrphan(Long id) {
    Session session = util.getSession();
    session.beginTransaction();
    Department department = (Department) session.get(Department.class, id);
    Employee employee = null;
    for (Employee e : department.getEmployees()) {
        employee = e;
        break;
    }
    department.getEmployees().remove(employee);
    session.saveOrUpdate(department);
    session.getTransaction().commit();
}
```

```
Hibernate: select department0_.DEPARTMENT_ID as DEPARTME1_0_0_, department0_.NAME as NAME2_
Hibernate: select employees0_.DEPARTMENT_ID as DEPARTME5_0_0_, employees0_.EMPLOYEE_ID as E
 FIRST_NA3_1_1_, employees0_.LAST_NAME as LAST_NAM4_1_1_, employeede1_.EMPLOYEE_ID as EMPL
 EMPLOYEE employees0_ left outer join EMPLOYEE_DETAILS employeede1_ on employees0_.EMPLOYEE
Hibernate: delete from EMPLOYEE where EMPLOYEE_ID=?
```

One To Many. Additions. Read assotiation.

IT-Academy

```
@OneToMany(mappedBy = "department")
@Fetch(FetchMode.SELECT)
@BatchSize(size = 2)
private Set<Employee> employees = new HashSet<>(0);
```

```
Session session = util.getSession();
session.beginTransaction();
Department department = (Department) session.get(Department.class, departmentId);
Iterator<Employee> it = department.getEmployees().iterator();
for (;it.hasNext();) {
    System.out.println(it.next());
}
session.getTransaction().commit();
util.releaseSession();
```

```
Hibernate: select department0_.DEPARTMENT_ID as DEPARTME1_0_0_, department0_.NAME as NAME2_0_0_ from D
Hibernate: select employees0_.DEPARTMENT_ID as DEPARTME5_0_2_, employees0_.EMPLOYEE_ID as EMPLOYEE1_1_
employees0_.FIRST_NAME as FIRST_NA3_1_1_, employees0_.LAST_NAME as LAST_NAM4_1_1_, employeede1_.EMPLOY
STATE4_2_0_, employeede1_.STREET as STREET5_2_0_ from EMPLOYEE employees0_ left outer join EMPLOYEE_DE
where employees0_.DEPARTMENT_ID=?
Employee{employeeId=7, firstname='Roma', lastname='Won'}
Employee{employeeId=6, firstname='Kim', lastname='Talk'}
Employee{employeeId=9, firstname='Roma', lastname='Won'}
Employee{employeeId=4, firstname='Tom', lastname='Hanks'}
Employee{employeeId=8, firstname='Roma', lastname='Won'}
```

One To Many. Additions. Read assotiation.

IT-Academy

```java
@OneToMany(mappedBy = "department")
@Fetch(FetchMode.JOIN)
@BatchSize(size = 2)
private Set<Employee> employees = new HashSet<>(0);
```

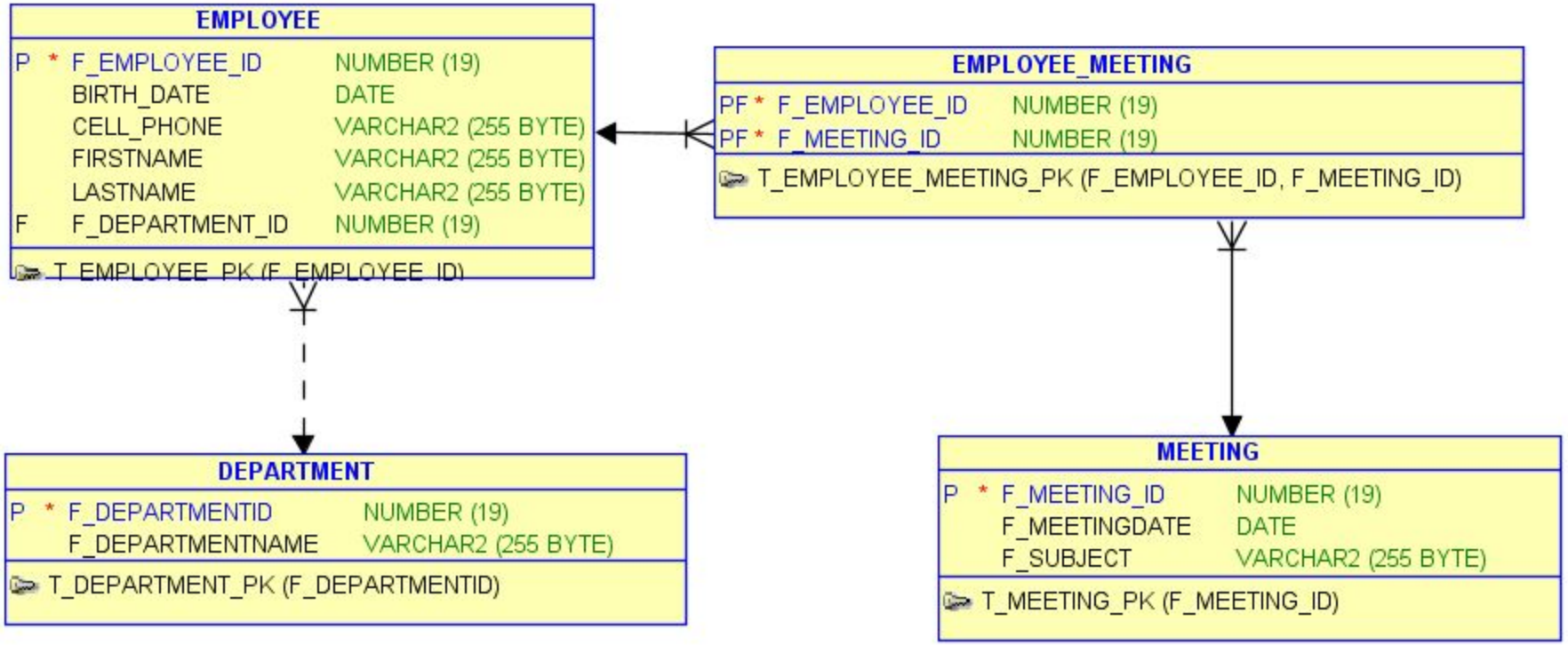
```java
Session session = util.getSession();
session.beginTransaction();
Department department = (Department) session.get(Department.class, departmentId);
Iterator<Employee> it = department.getEmployees().iterator();
for (;it.hasNext();) {
    System.out.println(it.next());
}
session.getTransaction().commit();
util.releaseSession();
```

```
Hibernate: select department0_.DEPARTMENT_ID as DEPARTME1_0_0_, department0_.NAME as NAM
   employees1_.date as date2_1_2_, employees1_.DEPARTMENT_ID as DEPARTME5_1_2_, employee
   CITY2_2_3_, employeede2_.COUNTRY as COUNTRY3_2_3_, employeede2_.STATE as STATE4_2_3_,
left outer join EMPLOYEE employees1_ on department0_.DEPARTMENT_ID=employees1_.DEPARTMEN
where department0_.DEPARTMENT_ID=?
Employee{employeeId=7, firstname='Roma', lastname='Won'}
Employee{employeeId=6, firstname='Kim', lastname='Talk'}
Employee{employeeId=9, firstname='Roma', lastname='Won'}
Employee{employeeId=4, firstname='Tom', lastname='Hanks'}
Employee{employeeId=8, firstname='Roma', lastname='Won'}
```

Вопросы

IT-Academy

Hibernate Annotation (Many To Many)



Many To Many

IT-Academy

```java
@Data @NoArgsConstructor @AllArgsConstructor
@Entity
@Table(name="EMPLOYEE")
public class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id @GeneratedValue(strategy = IDENTITY)
    @Column(name = "EMPLOYEE_ID", unique = true)
    private Long employeeId;
    @Column(name = "FIRST_NAME")
    private String firstname;
    @Column(name = "LAST_NAME")
    private String lastname;
    @Temporal(TemporalType.TIMESTAMP)
    private Date date;

    @OneToOne(mappedBy = "employee", cascade = CascadeType.PERSIST)
    private EmployeeDetail employeeDetail;

    @ManyToOne
    @JoinColumn(name = "DEPARTMENT_ID")
    private Department department;

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "EMPLOYEE_MEETING", joinColumns = {@JoinColumn(name = "EMPLOYEE_ID")},
            inverseJoinColumns = {@JoinColumn(name = "MEETING_ID")}
    )
    private Set<Meeting> meetings = new HashSet<>(0);
```

Many To Many

IT-Academy

```java
@Data
@NoArgsConstructor
@Entity
@Table(name = "MEETING")
public class Meeting implements Serializable {
    private static final long serialVersionUID = 8L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "MEETING_ID")
    private Long meetingId;

    @Column(name = "SUBJECT")
    private String subject;

    @Column(name = "DATE")
    private Date startDate;

    @ManyToMany(mappedBy = "meetings")
    private Set<Employee> employees = new HashSet<Employee>();

    public Meeting(String subject) {
        this.subject = subject;
        this.startDate = new Date();
    }
```

Many To Many

IT-Academy

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://127.0.0.1:3306/many_many</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">yuli</property>
        <property name="hibernate.connection.pool_size">10</property>
        <property name="hibernate.connection.isolation">2</property>
        <property name="show_sql">true</property>
        <property name="hibernate.hbm2ddl.auto">validate</property>
        <!--Mapping-->
        <mapping class="by.academy.it.pojos.Employee"/>
        <mapping class="by.academy.it.pojos.EmployeeDetail"/>
        <mapping class="by.academy.it.pojos.Department"/>
        <mapping class="by.academy.it.pojos.Meeting"/>
    </session-factory>
</hibernate-configuration>
```

Many To Many

IT-Academy

```
Session session = util.getSession();
session.beginTransaction();
Employee employee = (Employee) session.load(Employee.class, id);
Meeting meeting = new Meeting("Hibernate relation grooming");
meeting.getEmployees().add(employee);
employee.getMeetings().add(meeting);
session.saveOrUpdate(employee);
session.getTransaction().commit();
```

```
Hibernate: select employee0_.EMPLOYEE_ID as EMPLOYEE1_1_0_, employee0_.date as date2_1_0_, emplo
 DEPARTME1_0_1_, department1_.NAME as NAME2_0_1_, employeede2_.EMPLOYEE_ID as EMPLOYEE1_2_2_, em
 employee0_ left outer join DEPARTMENT department1_ on employee0_.DEPARTMENT_ID=department1_.DEP
Hibernate: select employees0_.DEPARTMENT_ID as DEPARTME5_0_0_, employees0_.EMPLOYEE_ID as EMPLOY
 FIRST_NA3_1_1_, employees0_.LAST_NAME as LAST_NAM4_1_1_, employeede1_.EMPLOYEE_ID as EMPLOYEE1_
 EMPLOYEE employees0_ left outer join EMPLOYEE_DETAILS employeede1_ on employees0_.EMPLOYEE_ID=e
Hibernate: select meetings0_.EMPLOYEE_ID as EMPLOYEE1_1_0_, meetings0_.MEETING_ID as MEETING_2_3
 MEETING meeting1_ on meetings0_.MEETING_ID=meeting1_.MEETING_ID where meetings0_.EMPLOYEE_ID=?
Hibernate: insert into MEETING (DATE, SUBJECT) values (?, ?)
Hibernate: insert into EMPLOYEE_MEETING (EMPLOYEE_ID, MEETING_ID) values (?, ?)
```

Вопросы

IT-Academy

Спасибо за внимание