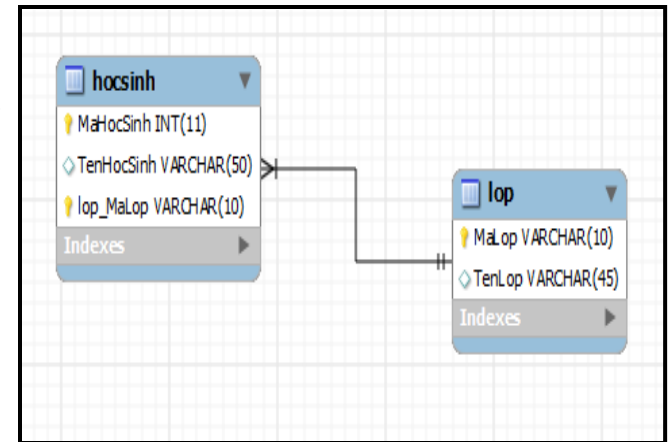
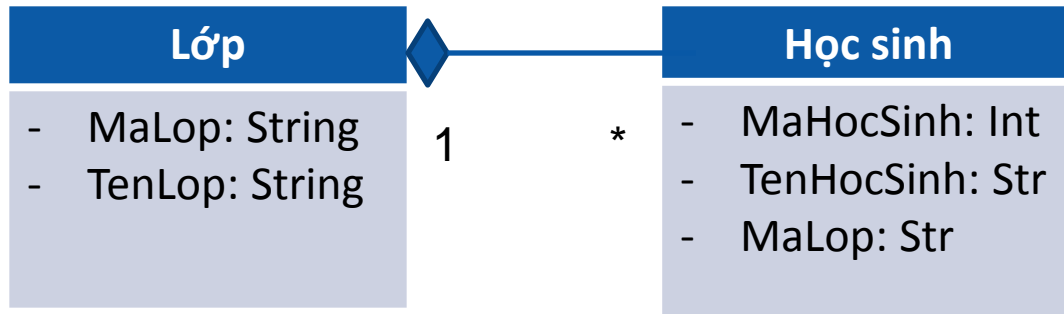


Bài 7: Hibernate Mapping

Nội dung bài học

- Many - to - One
- One to One
- One to Many
- Many to Many

Mapping Many To One



- Một **học sinh** thuộc về 1 **lớp**.
- Một **lớp** có nhiều **học sinh**.

Many to one: LopPOJO

```
1 package pojo;
2
3 public class LopPojo implements java.io.Serializable {
4     private String maLop;
5     private String tenLop;
6 }
//Các phương thức set, get, constructor
```

Many to one: Lop.hbm.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-mapping PUBLIC
3 "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
4 "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
5 <hibernate-mapping>
6     <class name="pojo.LopPojo" table="lop">
7         <id name="maLop" type="string">
8             <column name="MaLop" length="10"/>
9             <generator class="assigned"/>
10        </id>
11        <property name="tenLop" type="string">
12            <column name="TenLop" length="45" />
13        </property>
14    </class>
15 </hibernate-mapping>
```

Many to one: HocSinhPOJO

```
1 package pojo;
2
3 public class HocSinhPojo implements java.io.Serializable {
4     private int maHocSinh;
5     private String tenHocSinh;
6     private LopPojo lop;
7
8     //Các phương thức get, set, constructor.
9
10 }
```

Many to one: HocSinh.hbm.xml

```
1 <hibernate-mapping>
2     <class name="pojo.HocSinhPojo" table="hocsinh">
3         <id name="maHocSinh" column="MaHocSinh" type="integer">
4             <generator class="assigned"/>
5         </id>
6         <property name="tenHocSinh" column="TenHocSinh"
7 type="string"/>
8         <many-to-one name="lop" class="pojo.LopPojo" >
9             <column name="MaLop" />
10        </many-to-one>
11    </class>
12 </hibernate-mapping>
```

<many-to-one

name="lop" → tên thuộc tính cần mapping

class="pojo.LopPojo" > → Tên lớp cần mapping tới

<column name="MaLop" /> → Tên cột trong table HocSinh

</many-to-one>

Lấy thông tin học sinh

```
1 public class Main {
2     public static void main(String[] args) {
3         HocSinhPojo hs = null;
4         SessionFactory ssFac = MyHibernateUtil.getSessionFactory();
5         Session ss = ssFac.openSession();
6         ss.getTransaction().begin();
7         try {
8             hs = (HocSinhPojo)ss.get(HocSinhPojo.class, 1);
9             System.out.println("Tên học sinh: " + hs.getTenHocSinh());
10            System.out.println("Mã lớp: " + hs.getLop().getMaLop());
11            System.out.println("Tên lớp: " + hs.getLop().getTenLop());
12        } catch (HibernateException ex) {
13            System.out.println(ex.getMessage());
14        }
15        finally
16        {
17            ss.close();
18        }
19    }
20 }
```

Lấy thông tin học sinh khi còn mở Session



```
Tên học sinh: HỒ VĂN TẤN
Mã lớp: 10A
Tên lớp: Lớp 10 ban A
BUILD SUCCESSFUL (total time: 2 seconds)
```


Lấy thông tin học sinh

```
1 public class Main {
2     public static void main(String[] args) {
3         HocSinhPojo hs = null;
4         SessionFactory ssFac = MyHibernateUtil.getSessionFactory();
5         Session ss = ssFac.openSession();
6         ss.getTransaction().begin();
7         try {
8             hs = (HocSinhPojo)ss.get(HocSinhPojo.class, 1);
9         } catch (HibernateException ex ) {
10             System.out.println(ex.getMessage());
11         }
12         finally
13         {
14             ss.close();
15         }
16         System.out.println("Tên học sinh: " + hs.getTenHocSinh());
17         System.out.println("Mã lớp: " + hs.getLop().getMaLop());
18         System.out.println("Tên lớp: " + hs.getLop().getTenLop());
19     }
20 }
```

Lấy thông tin học sinh sau khi đóng Session

→ chỉ lấy được tên và mã học sinh, không lấy được tên lớp.

Lỗi

Lấy thông tin học sinh

```
Tên học sinh: Hồ Văn Tấn  
Mã lớp: 10A  
Exception in thread "main" org.hibernate.LazyInitializationException: could not initialize proxy - no Session  
    at org.hibernate.proxy.AbstractLazyInitializer.initialize(AbstractLazyInitializer.java:149)  
    at org.hibernate.proxy.AbstractLazyInitializer.getImplementation(AbstractLazyInitializer.java:195)  
    at org.hibernate.proxy.pojo.javassist.JavassistLazyInitializer.invoke(JavassistLazyInitializer.java:180)  
    at pojo.LopPojo_$$javassist_2.getTenLop(LopPojo_$$javassist_2.java)  
    at qlhs.Main.main(Main.java:31)  
Java Result: 1  
BUILD SUCCESSFUL (total time: 2 seconds)
```

Lấy thông tin học sinh sau khi đóng Session

→ chỉ lấy được tên và mã học sinh, không lấy được tên lớp.

Lỗi

Lấy thông tin học sinh

```
1 public class Main {
2     public static void main(String[] args) {
3         HocSinhPojo hs = null;
4         SessionFactory ssFac = MyHibernateUtil.getSessionFactory();
5         Session ss = ssFac.openSession();
6         ss.getTransaction().begin();
7         try {
8             hs = (HocSinhPojo)ss.get(HocSinhPojo.class, 1);
9             System.out.println("Tên lớp: " + hs.getLop().getTenLop());
10        } catch (HibernateException ex) {
11            System.out.println(ex.getMessage());
12        }
13        finally
14        {
15            ss.close();
16        }
17        System.out.println("Tên học sinh: " + hs.getTenHocSinh());
18        System.out.println("Mã lớp: " + hs.getLop().getMaLop());
19    }
20 }
```

```
Tên lớp: Lớp 10 ban A
Tên học sinh: HỒ VĂN TẤN
Mã lớp: 10A
BUILD SUCCESSFUL (total time: 2 seconds)
```



Lấy thông tin học sinh

- Nguyên nhân lỗi:
 - Cơ chế **Lazy Initialization** đang được bật (= true)
 - Truy vấn đối tượng **HocSinh** sẽ không kèm theo truy vấn đối tượng **Lop**. (chỉ có thể truy vấn được mã lớp mà không truy vấn được tên lớp).
 - Truy vấn đối tượng cha sẽ không kèm theo truy vấn đối tượng con.

Lazy Initialization & fetch

- Trong Hibernate, Lazy Initialization giúp
 - Tránh các câu truy vấn cơ sở dữ liệu không cần thiết
 - Gia tăng hiệu suất thực thi
 - Lazy mặc định có giá trị là true

Cách 1

- Sau khi có mã lớp, ta dùng làm lấy thông tin lớp theo mã lớp

```
LopDAO.layThongTinLop(int maLop);
```

Cách 2 – Khai báo lazy = false trong Hocsinh.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.HocSinhPojo" table="hocsinh">
3     <id name="maHocSinh" column="MaHocSinh" type="integer">
4       <generator class="assigned"/>
5     </id>
6     <property name="tenHocSinh" column="TenHocSinh" type="string"/>
7     <many-to-one name="lop" class="pojo.LopPojo" lazy="false" >
8       <column name="MaLop" />
9     </many-to-one>
10    </class>
11 </hibernate-mapping>
```

Cơ chế fetch

- Lazy = “false” truy vấn lớp cha kèm theo truy vấn lớp con.
 - Fetch = “select” sử dụng select để truy vấn lớp con. → sử dụng 2 câu truy vấn select để truy vấn cả lớp cha và con, cách này không hiệu quả vì phải truy xuất tới cơ sở dữ liệu 2 lần.
 - Fetch = “join” sử dụng phép kết để gộp truy vấn lớp cha và lớp con trong 1 truy vấn. → hiệu suất cao hơn, sử dụng 1 câu truy vấn.

Cơ chế fetch – sử dụng **select**

Hocsinh.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.HocSinhPojo" table="hocsinh">
3     <id name="maHocSinh" column="MaHocSinh" type="integer">
4       <generator class="assigned"/>
5     </id>
6     <property name="tenHocSinh" column="TenHocSinh" type="string"/>
7     <many-to-one name="lop" class="pojo.LopPojo" lazy="false" fetch="select">
8       <column name="MaLop" />
9     </many-to-one>
10    </class>
11 </hibernate-mapping>
```

Chú ý: mỗi khi sửa lại file cấu hình xml (cấu hình hibernate, cấu hình mapping, ...)
Phải **clean and built** lại project thì thay đổi mới có hiệu lực.

Cơ chế fetch – sử dụng **select**

Hocsinh.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.HocSinhPojo" table="hocsinh">
3     <id name="maHocSinh" column="MaHocSinh" type="integer">
4       <generator class="assigned"/>
5     </id>
6     <property name="tenHocSinh" column="TenHocSinh" type="string"/>
7     <many-to-one name="lop" class="pojo.LopPojo" lazy="false" fetch="select">
8       <column name="MaLop" />
9     </many-to-one>
10   </class>
11 </hibernate-mapping>
```

Chú ý: mỗi khi sửa lại file cấu hình xml (cấu hình hibernate, cấu hình mapping, ...)

Phải **clean and built** lại project thì thay đổi mới có hiệu lực.

Cơ chế fetch – sử dụng **select**

Bản chất, các câu truy vấn HQL đều được chuyển về SQL, như hình dưới có 2 câu select được gọi. ⇔ **truy xuất CSDL 2 lần**

```
Output
Hibernate-Mapping-Many-To-One-QLHS (clean.jar) x  Hibernate-Mapping-Many-To-One-QLHS (run) x
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Oct 16, 2011 1:00:26 PM org.hibernate.engine.jdbc.internal.LobCreatorBuild
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reports
Oct 16, 2011 1:00:26 PM org.hibernate.engine.transaction.internal.Transaction
INFO: HHH000399: Using default transaction strategy (direct JDBC transac
Oct 16, 2011 1:00:26 PM org.hibernate.hql.internal.ast.ASTQueryTranslator
INFO: HHH000397: Using ASTQueryTranslatorFactory
Hibernate:
select
    hocsinhpoj0_.MaHocSinh as MaHocSinh1_0_,
    hocsinhpoj0_.TenHocSinh as TenHocSinh1_0_,
    hocsinhpoj0_.MaLop as MaLop1_0_
from
    hocsinh hocsinhpoj0_
where
    hocsinhpoj0_.MaHocSinh=?
Hibernate:
select
    loppoj0_.MaLop as MaLop0_0_,
    loppoj0_.TenLop as TenLop0_0_
from
    lop loppoj0_
where
    loppoj0_.MaLop=?
Tên học sinh: HỒ VĂN TẤN
Mã danh mục: 10A
Tên danh mục: Lớp 10 ban A
BUILD SUCCESSFUL (total time: 1 second)
```

2 câu truy vấn
select được
gọi.

Cơ chế fetch – sử dụng join

Hocsinh.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.HocSinhPojo" table="hocsinh">
3     <id name="maHocSinh" column="MaHocSinh" type="integer">
4       <generator class="assigned"/>
5     </id>
6     <property name="tenHocSinh" column="TenHocSinh" type="string"/>
7     <many-to-one name="lop" class="pojo.LopPojo" lazy="false" fetch="join">
8       <column name="MaLop" />
9     </many-to-one>
10    </class>
11 </hibernate-mapping>
```

Chú ý: mỗi khi sửa lại file cấu hình xml (cấu hình hibernate, cấu hình mapping, ...)

Phải **clean and built** lại project thì thay đổi mới có hiệu lực.

Cơ chế fetch – sử dụng join

```
1 public class Main {
2     public static void main(String[] args) {
3         HocSinhPojo hs = null;
4         SessionFactory ssFac = MyHibernateUtil.getSessionFactory();
5         Session ss = ssFac.openSession();
6         ss.getTransaction().begin();
7         try {
8             String hql="select hs
9                             from HocSinhPojo hs left join fetch hs.lop
10                             where hs.maHocSinh=:maHocSinh";
11             Query query=ss.createQuery(hql);
12             query.setInteger("maHocSinh", 1);
13             hs = (HocSinhPojo) query.uniqueResult();
14         } catch (HibernateException ex) {
15             System.out.println(ex.getMessage());
16         } finally {
17             ss.close();
18         }
19         System.out.println("Tên họcsinh: " + hs.getTenHocSinh());
20         System.out.println("Mã danh mục: " + hs.getLop().getMaLop());
21         System.out.println("Tên danh mục: " + hs.getLop().getTenLop());
22     }
23 }
```

Cơ chế fetch – sử dụng join

Có 1 câu select được gọi, có sử dụng phép Join ⇔ truy xuất CSDL 2 lần

```
Output - Hibernate-Mapping-Many-To-One-QLHS (run)
INFO: HHH000397: Using ASTQueryTranslatorFactory
Hibernate:
  select
    hocsinhpoj0_.MaHocSinh as MaHocSinh1_0_,
    loppoj01_.MaLop as MaLop0_1_,
    hocsinhpoj0_.TenHocSinh as TenHocSinh1_0_,
    hocsinhpoj0_.MaLop as MaLop1_0_,
    loppoj01_.TenLop as TenLop0_1_
  from
    hocsinh hocsinhpoj0_
  left outer join
    lop loppoj01_
    on hocsinhpoj0_.MaLop=loppoj01_.MaLop
  where
    hocsinhpoj0_.MaHocSinh=?
Tên học sinh: Hồ Văn Tấn
Mã danh mục: 10A
Tên danh mục: Lớp 10 ban A
BUILD SUCCESSFUL (total time: 2 seconds)
```

1 câu truy vấn
select được
gọi.

Cascade

- Save – update
- Delete

Cascade – None cascade

```
1 <hibernate-mapping>
2   <class name="pojo.HocSinhPojo" table="hocsinh">
3     <id name="maHocSinh" column="MaHocSinh" type="integer">
4       <generator class="assigned"/>
5     </id>
6     <property name="tenHocSinh" column="TenHocSinh" type="string"/>
7     <many-to-one name="lop" class="pojo.LopPojo"
8       lazy="false" fetch="join" cascade="none">
9       <column name="MaLop" />
10    </many-to-one>
11  </class>
</hibernate-mapping>
```

Mặc định nếu không khai báo thì **cascade=none**

Cascade – không dùng update-save

```
1 public static void main(String[] args) {
2     HocSinhPojo hs = new HocSinhPojo(15, "Trần Văn Đạt", null);
3     LopPojo lop = new LopPojo("12E", "Lớp 12 chuyên Hóa");
4
5     hs.setLop(lop);
6     if(HocSinhDAO.themHocSinh(hs))
7     {
8         System.out.println("Thêm thành công!");
9     }
10    else
11        System.out.println("Thêm thất bại!");
12 }
```

Output - Hibernate-Mapping-Many-To-One-QLHS (run)

```
Oct 17, 2011 12:17:48 AM org.hibernate.internal.CoreMessageLogger_$logger warn
WARN: SQL Error: 1452, SQLState: 23000
Cannot add or update a child row: a foreign key constraint fails (`quanlyhocsinh`
Oct 17, 2011 12:17:48 AM org.hibernate.internal.CoreMessageLogger_$logger error
Thêm thất bại!
ERROR: Cannot add or update a child row: a foreign key constraint fails (`quanlyh
BUILD SUCCESSFUL (total time: 2 seconds)
```



Lớp 12E không tồn tại trong cơ sở dữ liệu.

Cascade – sử dụng update-save

HocSinh.hbm.xml

```
1 <hibernate-mapping>
2     <class name="pojo.HocSinhPojo" table="hocsinh">
3         <id name="maHocSinh" column="MaHocSinh" type="integer">
4             <generator class="assigned"/>
5         </id>
6         <property name="tenHocSinh" column="TenHocSinh" type="string"/>
7         <many-to-one name="lop" class="pojo.LopPojo"
8             lazy="false" fetch="join" cascade="save-update">
9             <column name="MaLop" />
10        </many-to-one>
11    </class>
</hibernate-mapping>
```

Cascade – sử dụng update-save

```
1 public static void main(String[] args) {
2     HocSinhPojo hs = new HocSinhPojo(15, "Trần Văn Đạt", null);
3     LopPojo lop = new LopPojo("12E", "Lớp 12 chuyên Hóa");
4
5     hs.setLop(lop);
6     if(HocSinhDAO.themHocSinh(hs))
7     {
8         System.out.println("Thêm thành công!");
9     }
10    else
11        System.out.println("Thêm thất bại!");
12 }
```

	MaLop	TenLop
	10A	Lớp 10 ban A
	10B	Lớp 10 ban B
	10C	Lớp 10 ban C
	10D	Lớp 10 ban D
	11A	Lớp 11 ban A
	11B	Lớp 11 ban B
	12A	Lớp 12 ban A
	12B	Lớp 12 ban B
	12D	Lớp 12 ban D
	12E	Lớp 12 chuyên Hóa
*	NULL	NULL



	MaHocSinh	TenHocSinh	MaLop
	1	Hồ Văn Tấn	10A
	2	Sử Bá Thuần	10B
	3	Văn Kinh Luân	10A
	4	Lương Quang Hà	10C
	5	Lý Thị Loan	10D
	15	Trần Văn Đạt	12E
*	NULL	NULL	NULL

Cascade – không dùng delete

```
1 public class Main {  
2     public static void main(String[] args) {  
3         if(LopDAO.xoaLop("12E"))  
4             System.out.println("Xóa thành công!");  
5         else  
6             System.out.println("Xóa thất bại!");  
7     }  
8 }
```

Output - Hibernate-Mapping-Many-To-One-QLHS (run)

```
ERROR: Cannot delete or update a parent row: a foreign key constraint fails (`quanlyhoc`  
Oct 17, 2011 12:54:22 AM org.hibernate.engine.jdbc.batch.internal.AbstractBatchImpl  
org.hibernate.exception.ConstraintViolationException: Cannot delete or update a parent  
INFO: HHH000010: On release of batch it still contained JDBC statements  
BUILD SUCCESSFUL (total time: 2 seconds)
```

LỖI

Không thể xóa lớp 12E → lỗi tham chiếu khóa ngoại

Cascade – sử dụng delete

CHÚ Ý

The screenshot shows the 'hocsinh' database management tool interface. The 'Foreign Keys' tab is selected, displaying the configuration for the 'FK_Lop' foreign key. The 'Referenced Table' is set to 'quanlyhocsinh' and the 'Referenced Column' is 'MaLop'. The 'Column' list includes 'MaHocSinh', 'TenHocSinh', and 'MaLop', with 'MaLop' selected. The 'Foreign Key Options' section shows 'On Update' and 'On Delete' both set to 'CASCADE'. The 'Skip in SQL generation' checkbox is unchecked. The 'Foreign Key Comment' field is empty. The bottom status bar indicates 'DBMS feedback messages will go here upon applying changes.' and includes 'Apply', 'Revert', and 'Close' buttons.

Foreign Key Name	Referenced Table
FK_Lop	'quanlyhocsinh', 'lop'

Column	Referenced Column
<input type="checkbox"/> MaHocSinh	
<input type="checkbox"/> TenHocSinh	
<input checked="" type="checkbox"/> MaLop	MaLop

Foreign Key Options

On Update: CASCADE

On Delete: CASCADE

☐ Skip in SQL generation

- Foreign Key Comment

Table Columns Indexes Foreign Keys Triggers Partitioning Options

DBMS feedback messages will go here upon applying changes.

Apply Revert Close

Để dùng Cascade trong hibernate phải cho phép sử dụng cascade trong CSDL

Cascade – sử dụng delete

```
1 <hibernate-mapping>
2   <class name="pojo.HocSinhPojo" table="hocsinh">
3     <id name="maHocSinh" column="MaHocSinh" type="integer">
4       <generator class="assigned"/>
5     </id>
6     <property name="tenHocSinh" column="TenHocSinh" type="string"/>
7     <many-to-one name="lop" class="pojo.LopPojo"
8       lazy="false" fetch="join" cascade="save-update, delete">
9       <column name="MaLop" />
10    </many-to-one>
11  </class>
</hibernate-mapping>
```

Cascade – sử dụng delete

```
1 public class Main {  
2     public static void main(String[] args) {  
3         if(LopDAO.xoaLop("12E"))  
4             System.out.println("Xóa thành công!");  
5     else  
6         System.out.println("Xóa thất bại!");  
7     }  
8 }
```

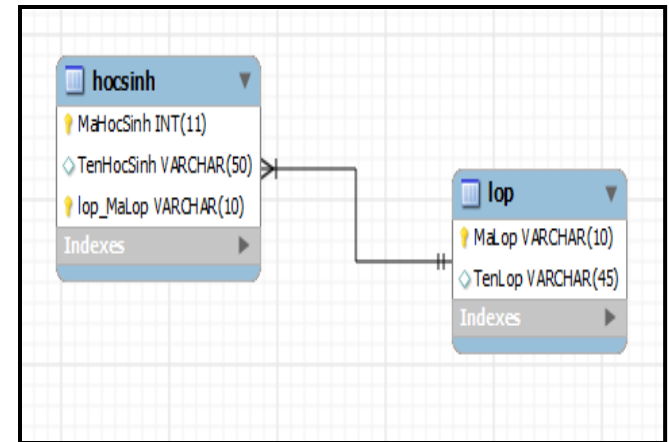
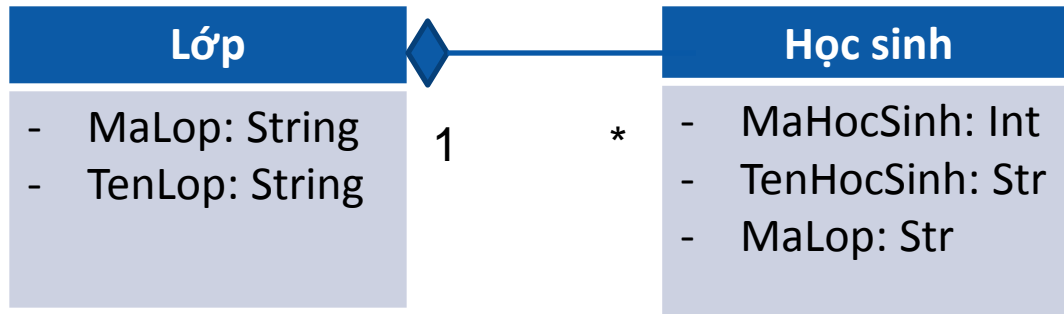
	MaLop	TenLop
▶	10A	Lớp 10 ban A
	10B	Lớp 10 ban B
	10C	Lớp 10 ban C
	10D	Lớp 10 ban D
	11A	Lớp 11 ban A
	11B	Lớp 11 ban B
	12A	Lớp 12 ban A
	12B	Lớp 12 ban B
	12D	Lớp 12 ban D
*	NULL	NULL



Xóa thành công!
BUILD SUCCESSFUL (total time: 1 second)

	MaLop	TenLop
▶	10A	Lớp 10 ban A
	10B	Lớp 10 ban B
	10C	Lớp 10 ban C
	10D	Lớp 10 ban D
	11A	Lớp 11 ban A
	11B	Lớp 11 ban B
	12A	Lớp 12 ban A
	12B	Lớp 12 ban B
	12D	Lớp 12 ban D
*	NULL	NULL

Mapping One To Many



- Một **học sinh** thuộc về 1 **lớp**.
- Một **lớp** có nhiều **học sinh**.

LopPOJO

```
1 package pojo;
2
3 import java.util.HashSet;
4 import java.util.Set;
5
6 public class LopPojo implements java.io.Serializable {
7     private String maLop;
8     private String tenLop;
9     private Set<HocSinhPojo> danhSachHocSinh = new HashSet<HocSinhPojo>(0);
10
11     // Các phương thức get, set, construction
12
13 }
```

Lop.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.LopPojo" table="lop">
3     <id name="maLop" type="string">
4       <column name="MaLop" length="10"/>
5       <generator class="assigned"/>
6     </id>
7     <property name="tenLop" type="string">
8       <column name="TenLop" length="45" />
9     </property>
10    <set name="danhSachHocSinh" lazy="false" fetch="select">
11      <key>
12        <column name="MaLop"/>
13      </key>
14      <one-to-many class="pojo.HocSinhPojo" />
15    </set>
  </class>
</hibernate-mapping>
```

HocSinhPOJO

```
1 package pojo;
2
3 public class HocSinhPojo implements java.io.Serializable {
4     private int maHocSinh;
5     private String tenHocSinh;
6     private LopPojo lop;
7
8     //Các phương thức get, set, constructor.
9
10 }
```

HocSinh.hbm.xml

```
1 <hibernate-mapping>
2     <class name="pojo.HocSinhPojo" table="hocsinh">
3         <id name="maHocSinh" column="MaHocSinh" type="integer">
4             <generator class="assigned"/>
5         </id>
6         <property name="tenHocSinh" column="TenHocSinh"
7 type="string"/>
8         <many-to-one name="lop" class="pojo.LopPojo" >
9             <column name="MaLop" />
10        </many-to-one>
11    </class>
12 </hibernate-mapping>
```

DAO: Lấy thông tin lớp học

```
1 public static LopPojo layThongTinLop(String maLop) {  
2     LopPojo lop = null;  
3     SessionFactory ssFac = MyHibernateUtil.getSessionFactory();  
4     Session ss = ssFac.getCurrentSession();  
5     Transaction trans = ss.getTransaction();  
6     trans.begin();  
7     try {  
8         lop = (LopPojo)ss.get(LopPojo.class, maLop);  
9         trans.commit();  
10    } catch (HibernateException ex ) {  
11        System.out.println(ex.getMessage());  
12    }  
13    return lop;  
14 }
```

Lấy thông tin lớp học

```
1 public class Main {
2     public static void main(String[] args) {
3         LopPojo lop = LopDAO.layThongTinLop("10A");
4         System.out.println("Mã lớp: " + lop.getMaLop());
5         System.out.println("Tên lớp: " + lop.getTenLop());
6         System.out.println("-----");
7         Iterator<HocSinhPojo> dsHocSinh =
8             lop.getDanhsachHocSinh().iterator();
9
10        while(dsHocSinh.hasNext())
11        {
12            HocSinhPojo hs = dsHocSinh.next();
13            System.out.println("Mã học sinh: " + hs.getMaHocSinh());
14            System.out.println("Tên học sinh: " +
15                               hs.getTenHocSinh());
16            System.out.println("_____");
17        }
18    }
19 }
20
```

Lấy thông tin lớp học

Kết quả

```
Output - Hibernate-Mapping-Many-To-One-QLHS (run)
>> Mã lớp: 10A
>> Tên lớp: Lớp 10 ban A
-----
Mã học sinh: 1
Tên học sinh: HỒ VĂN TẤN
-----
Mã học sinh: 4
Tên học sinh: LƯƠNG QUANG HÀ
-----
Mã học sinh: 3
Tên học sinh: VĂN KINH LUÂN
-----
BUILD SUCCESSFUL (total time: 2 seconds)
```

INVERSER

Khi **tạo mới** hay **thay đổi** thông tin “một lớp học” thì có cần phải **cập nhật** lại “mã lớp” của học sinh hay không?

Inverser - False

Lưu ý: để xuất câu truy vấn SQL ra console cho tiện việc xem xét như các slide bên dưới ta phải thêm **<property name="hibernate.format_sql">true</property>** vào file cấu hình hibernate.cfg.xml.

```
1 <hibernate-configuration>
2   <session-factory>
3     <property
4       name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
5     <property
6       name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
7     <property
8       name="hibernate.connection.url">jdbc:mysql://localhost:3306/quanlysach?use
9       Unicode=true&characterEncoding=UTF-8</property>
10    <property name="hibernate.connection.username">root</property>
11    <property name="hibernate.connection.password">password</property>
12    <property
13      name="hibernate.current_session_context_class">thread</property>
14    <property name="hibernate.format_sql">true</property>
15    <mapping resource="entity/Khachhang.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

Inverser - False

Khi **tạo mới hay thay đổi** thông tin “một lớp học” thì có cần phải **cập nhật** lại “mã lớp” của học sinh hay không?

- **Nếu `inverser = false` (Mặc định)**

```
1 <set name="danhSachHocSinh" lazy="false" fetch="select"
2 inverse="false">
3     <key>
4         <column name="MaLop" />
5     </key>
6     <one-to-many class="pojo.HocSinhPojo" />
7 </set>
```

Inverser = false đồng nghĩa với **LopHoc** đóng vai trò chủ đạo trong mối quan hệ **LopHoc-HocSinh**, vì vậy khi tạo mới hay cập nhật thông tin một lớp học **sẽ phải cập nhật lại** mã lớp của học sinh.

Inverser - FALSE

```
1 public class Main {
2     public static void main(String[] args) {
3
4         Session session=MyHibernateUtil.getSessionFactory().getCurrentSession();
5         session.beginTransaction();
6
7         LopPojo lop = new LopPojo("12F", "Lớp 12 chuyên văn");
8
9         HocSinhPojo hs = new HocSinhPojo(6, "Mạc Thị Bưởi", lop);
10
11         lop.getDanhSachHocSinh().add(hs);
12
13         session.save(lop);
14         session.save(hs);
15         session.getTransaction().commit();
16     }
17 }
```

Output - Hibernate-Mapping-Many-To-One-QLHS (run)

Hibernate:

insert

into

lop

(TenLop, MaLop)

values

(?, ?)

Hibernate:

insert

into

hocsinh

(TenHocSinh, MaLop, MaHocSinh)

values

(?, ?, ?)

Hibernate:

update

hocsinh

set

MaLop=?

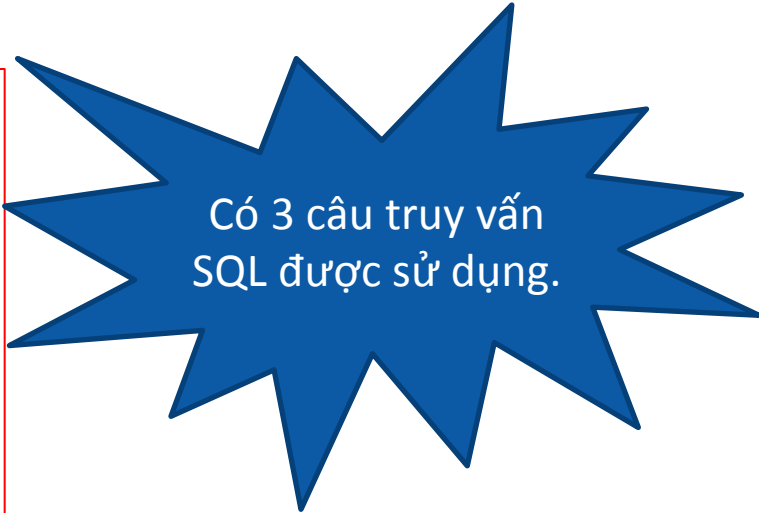
where

MaHocSinh=?

BUILD SUCCESSFUL (total time: 1 second)

Inverser - TRUE

```
Hibernate:
  insert
  into
    lop
    (TenLop, MaLop)
  values
    (?, ?)
Hibernate:
  insert
  into
    hocsinh
    (TenHocSinh, MaLop, MaHocSinh)
  values
    (?, ?, ?)
Hibernate:
  update
    hocsinh
  set
    MaLop=?
  where
    MaHocSinh=?
BUILD SUCCESSFUL (total time: 1 second)
```



Có 3 câu truy vấn SQL được sử dụng.

- Có 2 câu insert dữ liệu
- Có 1 câu update dùng để Update lại mã lớp của học sinh.
- Trong trường hợp này câu update không cần thiết.

Inverser

Khi **tạo mới hay thay đổi** thông tin “một lớp học” thì có cần phải **cập nhật** lại “mã lớp” của học sinh hay không?

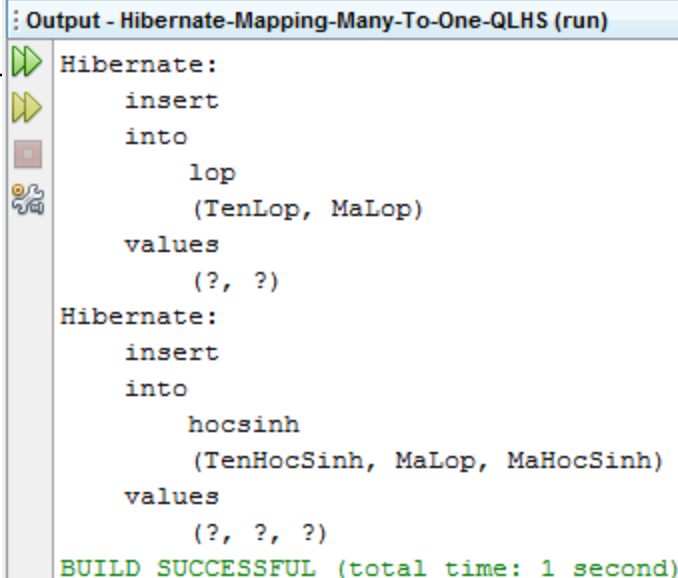
- **Nếu `inverser = true`**

```
1 <set name="danhSachHocSinh" lazy="false" fetch="select"
2 inverse="true">
3     <key>
4         <column name="MaLop" />
5     </key>
6     <one-to-many class="pojo.HocSinhPojo" />
7 </set>
```

Inverser = true đồng nghĩa với HocSinh đóng vai trò chủ đạo trong mối quan hệ này, vì vậy khi tạo mới hay cập nhật thông tin một lớp học sẽ không cần phải cập nhật lại mã lớp của học sinh.

Inverser - TRUE

```
1 public class Main {
2     public static void main(String[] args) {
3
4         Session session=MyHibernateUtil.getSessionFactory().getCurrentSession();
5         session.beginTransaction();
6
7         LopPojo lop = new LopPojo("12F", "Lớp 12 chuyên văn");
8
9         HocSinhPojo hs = new HocSinhPojo(6, "Mạc Thị Bưởi", lop);
10
11         lop.getDanhSachHocSinh().add(hs);
12
13         session.save(lop);
14         session.save(hs);
15         session.getTransaction().commit();
16     }
17 }
```

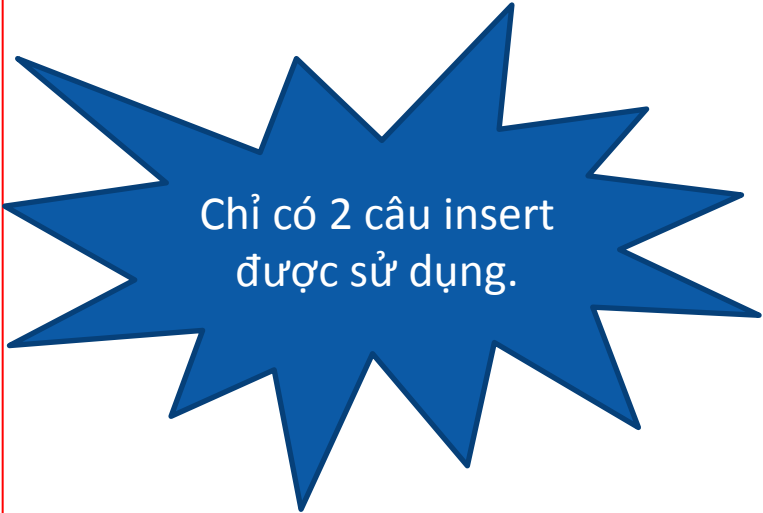


The screenshot shows the 'Output' window of an IDE, titled ': Output - Hibernate-Mapping-Many-To-One-QLHS (run)'. It displays two Hibernate SQL logs. The first log shows an insert into the 'lop' table with parameters '(TenLop, MaLop)' and values '(?, ?)'. The second log shows an insert into the 'hocsinh' table with parameters '(TenHocSinh, MaLop, MaHocSinh)' and values '(?, ?, ?)'. At the bottom, a green status bar indicates 'BUILD SUCCESSFUL (total time: 1 second)'.

```
: Output - Hibernate-Mapping-Many-To-One-QLHS (run)
Hibernate:
insert
into
    lop
    (TenLop, MaLop)
values
    (?, ?)
Hibernate:
insert
into
    hocsinh
    (TenHocSinh, MaLop, MaHocSinh)
values
    (?, ?, ?)
BUILD SUCCESSFUL (total time: 1 second)
```

Inverser - TRUE

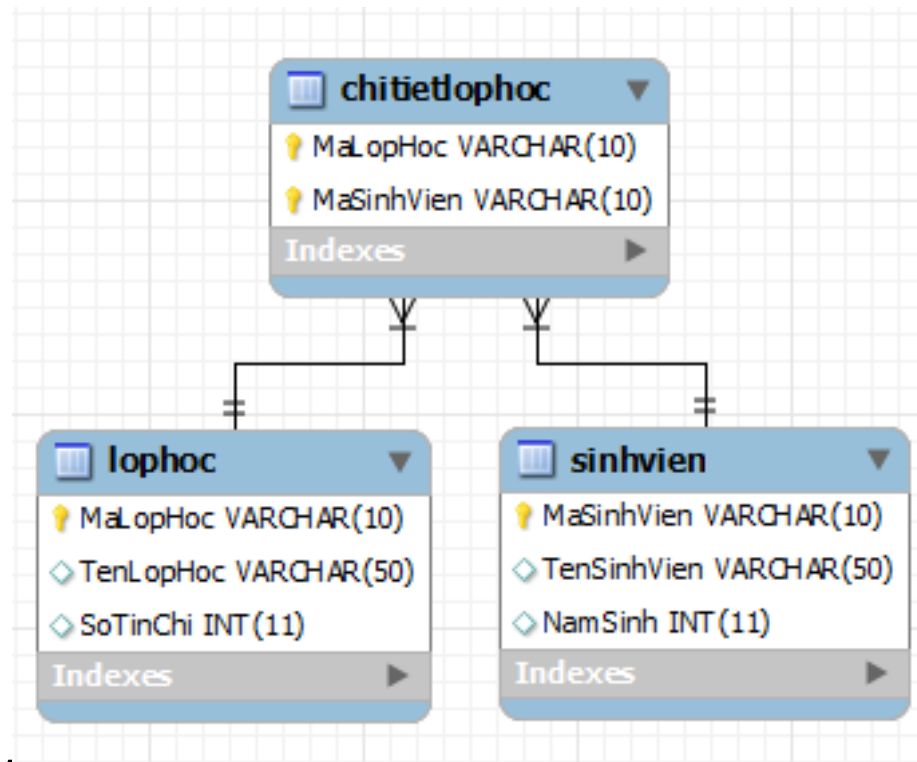
```
Hibernate:
  insert
  into
    lop
    (TenLop, MaLop)
  values
    (?, ?)
Hibernate:
  insert
  into
    hocsinh
    (TenHocSinh, MaLop, MaHocSinh)
  values
    (?, ?, ?)
BUILD SUCCESSFUL (total time: 1 second)
```



Chỉ có 2 câu insert
được sử dụng.

Bớt được một truy vấn đến CSDL
→ tăng hiệu năng đáng kể.

Many to many



- Quan hệ Many – Many:
 - Một **sinhvien** có thể học nhiều **lophoc**.
 - Một **lophoc** có nhiều **sinhvien**.

LopHocPOJO & SinhVienPOJO

```
1 public class LopHocPOJO implements java.io.Serializable {  
2     private String maLopHoc;  
3     private String tenLopHoc;  
4     private int soTinChi;  
5     private Set<SinhVienPOJO> dsSinhVien = new HashSet<SinhVienPOJO>();  
6     //Các phương thức set, get, constructor  
}
```

```
1 public class SinhVienPOJO implements java.io.Serializable {  
2     private String maSinhVien;  
3     private String tenSinhVien;  
4     private int namSinh;  
5     private Set<LopHocPOJO> dsLopHoc = new HashSet<LopHocPOJO>();  
6     //Các phương thức set, get, constructor  
}
```

Mapping: SinhVien.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.SinhVienPOJO" table="sinhvien">
3     <id name="maSinhVien" column="MaSinhVien" type="string" length="10"/>
4     <property name="tenSinhVien" type="string" column="TenSinhVien" length="50" />
5     <property name="namSinh" type="integer" column="NamSinh" />
6     <set name="dsLopHoc" table="chitietlophoc">
7       <key column="MaSinhVien"/>
8       <many-to-many column="MaLopHoc" class="pojo.LopHocPOJO"></many-to-many>
9     </set>
10   </class>
11 </hibernate-mapping>
```

Mapping: LopHoc.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.LopHocPOJO" table="lophoc">
3     <id name="maLopHoc" type="string" column="MaLopHoc" length="10" />
4     <property name="tenLopHoc" type="string" column="TenLopHoc" length="50" />
5     <property name="soTinChi" type="integer" column="SoTinChi" />
6     <set name="dsSinhVien" table="chitietlophoc">
7       <key column="MaLopHoc"/>
8       <many-to-many column="MaSinhVien" class="pojo.SinhVienPOJO"></many-to-many>
9     </set>
10   </class>
11 </hibernate-mapping>
```

SinhVienDAO

```
1 public static List<SinhVienPOJO> layDsSinhVien() {  
2     List<SinhVienPOJO> list = null;  
3     SessionFactory sf = MyHibernateUtil.getSessionFactory();  
4     Session ss = sf.getCurrentSession();  
5     Transaction trans = ss.beginTransaction();  
6     try {  
7         trans.begin();  
8         list = ss.createQuery("from pojo.SinhVienPOJO").list();  
9         trans.commit();  
10    } catch (Exception ex) {  
11        System.out.append(ex.getMessage());  
12    }  
13    return list;  
14 }
```

App

```
1 List<SinhVienPOJO> list = dao.SinhVienDAO.layDsSinhVien();
2     for (int i = 0; i < list.size(); ++i) {
3         System.out.println(list.get(i).getMaSinhVien() + "-"
4             + list.get(i).getTenSinhVien() + "-"
5             + list.get(i).getNamSinh());
6     }
```

	MaSinhVien	TenSinhVien	NamSinh
▶	SV01	Hồ Văn Tấn	1990
	SV02	Sử Bá Thuận	1990
	SV03	Đào Quang Duy	1989
*	NULL	NULL	NULL

```
SV01-Hồ Văn Tấn-1990
SV02-Sử Bá Thuận-1990
SV03-Đào Quang Duy-1989
```

App

```
1 List<SinhVienPOJO> list = dao.SinhVienDAO.layDsSinhVien();
2   for (int i = 0; i < list.size(); ++i) {
3       System.out.println(list.get(i).getMaSinhVien() + "-"
4           + list.get(i).getTenSinhVien() + "-"
5           + list.get(i).getNamSinh());
6       Iterator<LopHocPOJO> dsLop = list.get(i).getDsLopHoc().iterator();
7       while (dsLop.hasNext()) {
8           LopHocPOJO lh = dsLop.next();
9           System.out.println(lh.getMaLopHoc() +
10               "-" + lh.getTenLopHoc() +
11               "-" + lh.getSoTinChi());
12       }
13   }
```



failed to lazily initialize a collection of role:.pojo.SinhVienPOJO.dsLopHoc, no session or session was closed

App

- Cách giải quyết:
 - Điều chỉnh thuộc tính lazy trong file mapping.
 - Mặc định lazy = "true"

```
1 <set name="dsLopHoc" table="chitietlophoc" lazy="false" fetch="join">
2     <key column="MaSinhVien"/>
3     <many-to-many column="MaLopHoc" class="pojo.LopHocPOJO"/>
4 </set>
```

Lazy Initialization & fetch

- Lazy = "true" truy vấn lớp cha **sẽ không kèm theo** truy vấn lớp con
- Lazy = "false" truy vấn lớp cha **kèm theo** truy vấn lớp con.
 - Fetch = "select" sử dụng select để truy vấn lớp con. → sử dụng 2 câu truy vấn, làm chậm tốc độ.
 - Fetch = "join" sử dụng phép kết để gộp truy vấn lớp cha và lớp con trong 1 truy vấn.

Lazy Initialization & fetch

- Trong Hibernate, Lazy Initialization giúp
 - Tránh các câu truy vấn cơ sở dữ liệu không cần thiết
 - Gia tăng hiệu suất thực thi
 - Lazy mặc định có giá trị là true

App

```
1 List<SinhVienPOJO> list = dao.SinhVienDAO.layDsSinhVien();
2     for (int i = 0; i < list.size(); ++i) {
3         System.out.println(list.get(i).getMaSinhVien() + "-"
4             + list.get(i).getTenSinhVien() + "-"
5             + list.get(i).getNamSinh());
6         Iterator<LopHocPOJO> dsLop = list.get(i).getDsLopHoc().iterator();
7         while (dsLop.hasNext()) {
8             LopHocPOJO lh = dsLop.next();
9             System.out.println(lh.getMaLopHoc() +
10                 "-" + lh.getTenLopHoc() +
11                 "-" + lh.getSoTinChi());
12         }
13     }
```

Kết quả sau khi chạy lại đoạn code.



```
SV01-Hồ Văn Tân-1990
CTT02-Tin học cơ sở-4
CTT01-Nhập môn lập trình-4
SV02-Sử Bá Thuần-1990
CTT02-Tin học cơ sở-4
SV03-Đào Quang Duy-1989
```


SinhVienDAO

```
1 public static boolean dangKyLop(SinhVienPOJO info) {
2     boolean kq = true;
3     SessionFactory sf = MyHibernateUtil.getSessionFactory();
4     Session ss = sf.getCurrentSession();
5     Transaction trans = ss.beginTransaction();
6     try {
7         trans.begin();
8         ss.saveOrUpdate(info);
9         trans.commit();
10    } catch (Exception ex) {
11        kq = false;
12        System.out.append(ex.getMessage());
13    }
14    return kq;
15 }
```

App

```
1 SinhVienPOJO sv = dao.SinhVienDAO.layThongTinSinhVien("SV03");  
2   sv.getDsLopHoc().add(new LopHocPOJO("CTT03", "Java phân tán", 4));  
3   dao.SinhVienDAO.dangKyLop(sv);
```

```
SEVERE: Cannot add or update a child row: a foreign key constraint fails (`quanlysinhvien`.`chitietlophoc`, CONSTRAINT `FK_Lop`  
Oct 16, 2011 4:32:40 PM org.hibernate.event.def.AbstractFlushingEventListener performExecutions  
SEVERE: Could not synchronize database state with session  
org.hibernate.exception.ConstraintViolationException: Could not execute JDBC batch update  
Could not execute JDBC batch update      at org.hibernate.exception.SQLStateConverter.convert(SQLStateConverter.java:71)
```



Trong CSDL không tồn
tại sẵn lophoc có mã
là CTT03 như lúc
thêm

Cascade

- Cách giải quyết:
 - Điều chỉnh thuộc tính cascade trong file mapping.
 - Cascade có 2 giá trị: save-update / delete
 - Nếu dùng cả 2 giá trị thì cách nhau bằng dấu ,

```
1 <set name="dsLopHoc" table="chitietlophoc" lazy="false" fetch="join"  
2 cascade="save-update">  
3     <key column="MaSinhVien"/>  
4     <many-to-many column="MaLopHoc" class="pojo.LopHocPOJO"/>  
    </set>
```

App

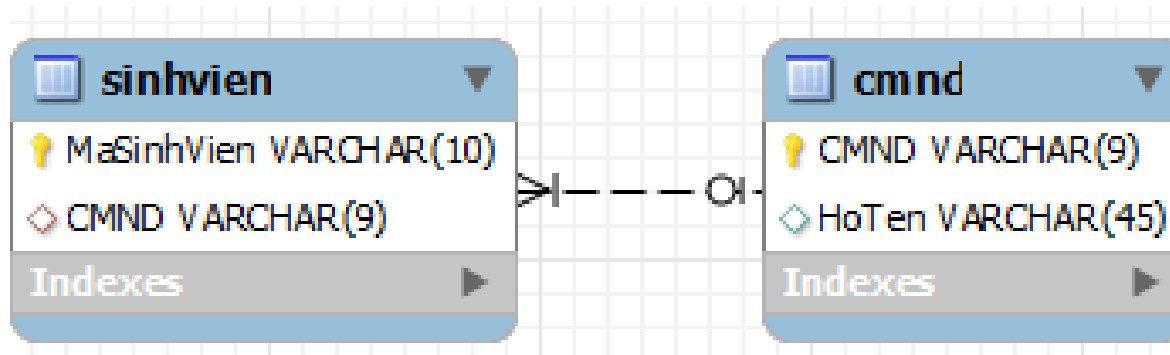
```
1 SSinhVienPOJO sv = dao.SinhVienDAO.layThongTinSinhVien("SV03");
2     sv.getDsLopHoc().add(new LopHocPOJO("CTT03", "Java phân tán", 4));
3     boolean kq = dao.SinhVienDAO.dangKyLop(sv);
4     if (kq)
5         System.out.println("Thêm thành công");
6     else
7         System.out.println("Thêm thất bại");
```

	MaLopHoc	TenLopHoc	SoTinChi
▶	CTT01	Nhập môn lập trình	4
	CTT02	Tin học cơ sở	4
	CTT03	Java phân tán	4
*	NULL	NULL	NULL

	MaLopHoc	MaSinhVien
▶	CTT01	SV01
	CTT02	SV01
	CTT02	SV02
	CTT03	SV03
*	NULL	NULL

```
Thêm thành công
BUILD SUCCESSFUL (total time: 2 seconds)
```

One to One



- Quan hệ One – One:
 - Một **sinhvien** có duy nhất một **cmnd**.
 - Một **cmnd** thuộc về duy nhất một **sinhvien**.

SinhVienPOJO & cmndPOJO

```
1 public class SinhVienPOJO implements java.io.Serializable {  
2     private String maSinhVien;  
3     private cmndPOJO cmnd;  
4     //Các phương thức set, get, constructor  
5 }
```

```
1 public class cmndPOJO implements java.io.Serializable {  
2     private String cmnd;  
3     private String hoTen;  
4     private SinhVienPOJO sinhVien;  
5     //Các phương thức set, get, constructor  
6 }
```


Mapping

- Mapping mỗi quan hệ một-một giống như mapping mỗi quan hệ nhiều-một:
 - Nhưng thêm thuộc tính `unique="true"`
- Có thể khai báo sử dụng thuộc tính:
 - Lazy
 - Fetch
 - Cascade

Mapping: SinhVien.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.SinhVienPOJO" table="sinhvien">
3     <id name="maSinhVien" column="MaSinhVien" type="string" length="10" />
4     <many-to-one class="pojo.cmndPOJO" name="cmnd" fetch="join" lazy="false"
5 cascade="save-update,delete">
6       <column name="CMND" length="9" unique="true" />
7     </many-to-one>
8   </class>
9 </hibernate-mapping>
```

Mapping: cmnd.hbm.xml

```
1 <hibernate-mapping>
2   <class name="pojo.cmndPOJO" table="cmnd">
3     <id name="cmnd" type="string" column="CMND" length="9" />
4     <property name="hoTen" column="HoTen" length="45" type="string" />
5     <one-to-one name="sinhVien" class="antlr.SinhVienPOJO" property-ref="cmnd"
6 cascade="save-update,delete" />
7   </class>
8 </hibernate-mapping>
```

SinhVienDAO

```
1 public static List<SinhVienPOJO> layDanhSachSinhVien() {  
2     List<SinhVienPOJO> list = null;  
3     SessionFactory sf = MyHibernateUtil.getSessionFactory();  
4     Session ss = sf.getCurrentSession();  
5     Transaction trans = ss.getTransaction();  
6     try {  
7         trans.begin();  
8         list = ss.createQuery("from pojo.SinhVienPOJO").list();  
9         trans.commit();  
10    } catch (Exception ex) {  
11        System.out.println(ex.getMessage());  
12    }  
13    return list;  
14 }
```

App

```
1 List<SinhVienPOJO> dsSV = dao.SinhVienDAO.layDanhSachSinhVien();
2   for (int i = 0; i < dsSV.size(); ++i) {
3       SinhVienPOJO sv = dsSV.get(i);
4       cmndPOJO cmnd = sv.getCmnd();
5       System.out.println("Mã SV: " + sv.getMaSinhVien());
6       System.out.println("Số CMND: " + cmnd.getCmnd());
7       System.out.println("Họ tên: " + cmnd.getHoTen());
8       System.out.println("-----");
9   }
```

	MaSinhVien	CMND
▶	0812463	222222222
	0812505	333333333
*	NULL	NULL

	CMND	HoTen
▶	222222222	Hồ Văn Tấn
	333333333	Sử Bá Thuận
*	NULL	NULL

```
Mã SV: 0812463
Số CMND: 222222222
Họ tên: HỒ VĂN TẤN
-----
Mã SV: 0812505
Số CMND: 333333333
Họ tên: SỬ BÁ THUẬN
-----
```

SinhVienDAO

```
1 public static SinhVienPOJO layThongTinSinhVien(String maSinhVien){
2     SinhVienPOJO sv = null;
3     SessionFactory sf = MyHibernateUtil.getSessionFactory();
4     Session ss = sf.getCurrentSession();
5     Transaction trans = ss.getTransaction();
6     try {
7         trans.begin();
8         sv = (SinhVienPOJO)ss.get(SinhVienPOJO.class, maSinhVien);
9         trans.commit();
10    }catch (Exception ex) {
11        System.out.println(ex.getMessage());
12    }
13    return sv;
14 }
```

SinhVienDAO

```
1 public static boolean themSinhVien(SinhVienPOJO info) {
2     boolean kq = true;
3     if (SinhVienDAO.layThongTinSinhVien(info.getMaSinhVien()) != null) {
4         return false;
5     }
6     SessionFactory sf = MyHibernateUtil.getSessionFactory();
7     Session ss = sf.getCurrentSession();
8     Transaction trans = ss.getTransaction();
9     try {
10         trans.begin();
11         ss.save(info);
12         trans.commit();
13     } catch (Exception ex) {
14         trans.rollback();
15         System.out.println(ex.getMessage());
16         kq = false;
17     }
18     return kq;
19 }
```

App

```
1 SinhVienPOJO sv = new SinhVienPOJO();
2   sv.setMaSinhVien("0812462");
3   cmndPOJO cmnd = new cmndPOJO();
4   cmnd.setCmnd("4444444444");
5   cmnd.setHoTen("Thái Huy Tân");
6   sv.setCmnd(cmnd);
7   boolean kq = dao.SinhVienDAO.themSinhVien(sv);
8   if (kq)
9       System.out.println("Thêm thành công");
10  else
11      System.out.println("Thêm thất bại");
```

	MaSinhVien	CMND		CMND	HoTen
▶	0812463	222222222	▶	222222222	Hồ Văn Tấn
	0812505	333333333		333333333	Sử Bá Thuận
	0812462	444444444		444444444	Thái Huy Tân
*	NULL	NULL	*	NULL	NULL

```
Thêm thành công
BUILD SUCCESSFUL (total time: 2 seconds)
```


XIN CẢM ƠN!

