Bài 7: HIBERNATE

Nội dung

- ORM
- Giới thiệu Hibernate
- Cài đặt Hibernate

JDBC

- Là hình thức kết nối trực tiếp CSDL (Connected Model)
  - Connect: tạo kết nối chương trình với Hệ quản trị CSDL
  - Statement: thực hiện câu lệnh SQL bằng kết nối đã tạo
  - Results: xử lý kết quả
- Ưu điểm:
  - Thực thi nhanh, dễ sử dụng
- Nhược điểm:
  - Tốn nhiều thời gian để viết và kiểm thử cho các câu lệnh SQL của mỗi bảng
  - Nếu ứng dụng kết nối với nhiều hệ quản trị CSDL khác nhau, câu truy vấn có thể khác nhau trong mỗi hệ quản trị CSDL

JDBC (tt)

- Sử dụng dữ liệu theo hướng đối tượng
- Trong JDBC có 2 loại đối tượng dữ liệu:
  - RowSet
  - CachedRowSet

ORM (Object Relational Mapping)

- CSDL thường được thiết kế theo mô hình quan hệ
- Tuy nhiên, hiện nay, phần mềm thường được thiết kế theo hương đối tượng
- Đối với lập trình viên khi xây dựng phần mềm thường muốn làm việc với các đối tượng và không phải nhớ đến các dòng, các cột trong các bảng của cơ sở dữ liệu. ORM cho phép lấy dữ liệu từ CSDL quan hệ vào trong đối tượng
- Ánh xạ dữ liệu giữa các CSDL quan hệ và các đối tượng trong ngôn ngữ lập trình hướng đối tượng

ORM (tt)

- Ưu điểm:
  - Tính độc lập: có thể dùng với nhiều RDBMS khác nhau thông qua cấu hình.
  - Tính đơn giản, dễ sử dụng: ORM cung cấp các API đơn giản và rất dễ sử dụng trong việc truy vấn, thêm, xóa, sửa dữ liệu
  - Tiết kiệm thời gian lập trình, dễ bảo trì, sửa chữa
- Nhược điểm:
  - Tự động sinh ra các câu lệnh SQL nên sẽ khó có thể tác động vào để tối ưu câu lệnh.
  - Việc quản lý session khá phức tạp

ORM với Hibernate

- Được phát triển bởi Gavin King năm 2001
- JBoss chính thức phát triển Hibernate từ năm 2003
- Một ứng dụng có thể chia làm 3 phần: presentation layer, business layer, data layer
- Business layer có thể chia thành 2 layer con:
  - Business logic layer (các tính toán nhằm thỏa mãn yêu cầu người dùng
  - Persistence layer chịu trách nhiệm giao tiếp với data layer (mở kết nối, truy xuất và lưu trữ dữ liệu vào CSDL)
- Hibernate framework là một framework cho persistence layer

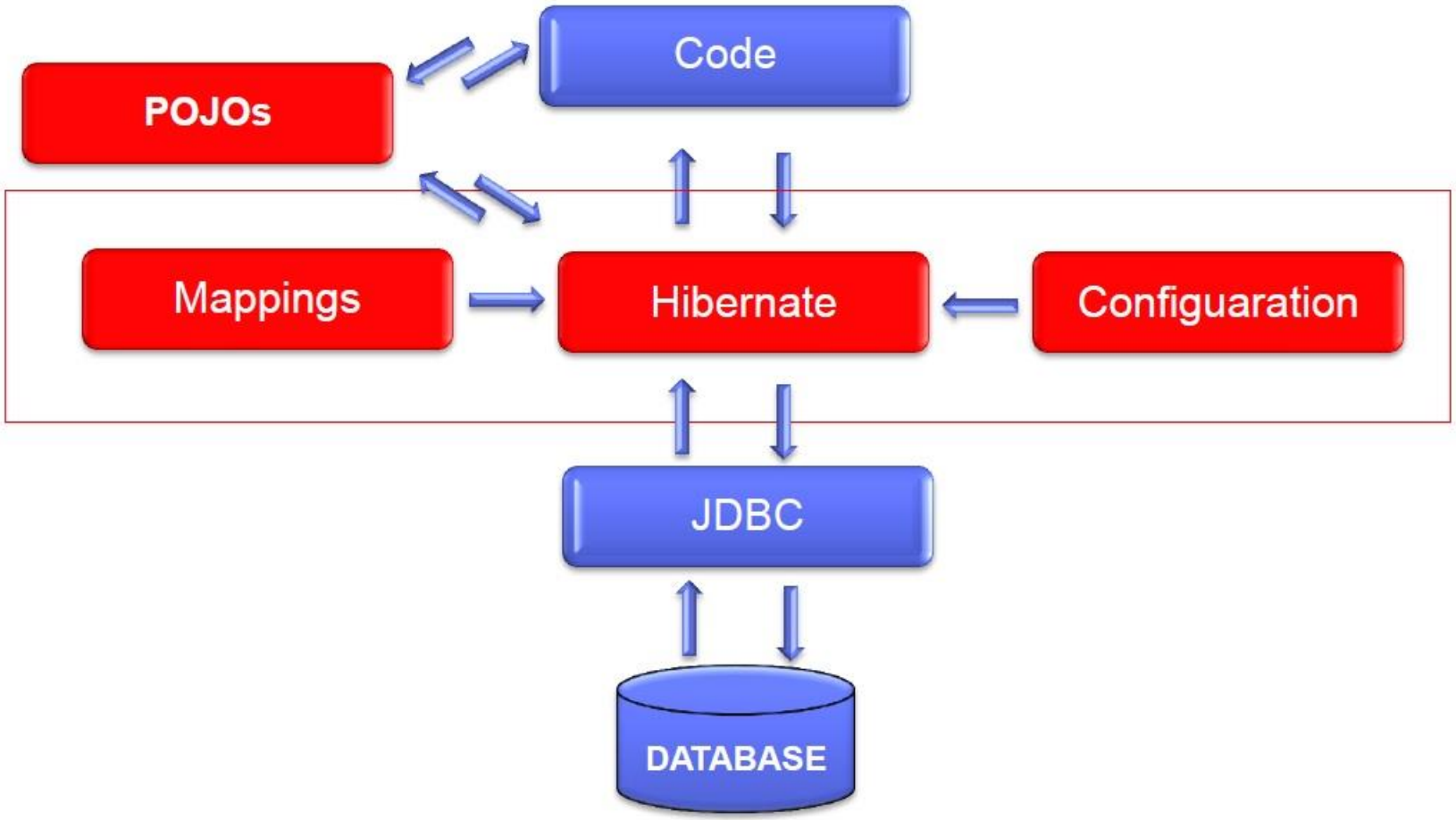
Hibernate

- Ánh xạ giữa các lớp Java đến các bảng trong CSDL dùng các file XML
- Cung cấp các API đơn giản để lưu trữ, truy xuất trực tiếp các đối tượng Java và CSDL.
- Nếu có bất kỳ thay đổi nào trong CSDL thì chỉ cần thay đổi file XML.
- Cung cấp đầy đủ các tiện ích, tính năng truy vấn dữ liệu đơn giản, hiệu quả.
- Thao tác, xử lý được các quan hệ phức tạp của các đối tượng trong CSDL.
- Giảm thiểu sự truy cập đến CSDL đến mức thấp nhất nhờ có chiến lược tìm, nạp thông minh.

Hibernate

- Hibernate O/R Mapping
  - 1-1, 1-n, n-1, n-n
  - Component, Inheritance
- Hibernate Transaction & Concurrency
- Hibernate Query
  - HQL, Criteria Query, Native SQL
- Hibernate Cache, Filter, Interceptor, Event
- Hibernate Monitor
- Hibernate Toolset

Kiến trúc Hibernate



Các bước sử dụng Hibernate

- Bước 1: Tạo cơ sở dữ liệu
- Bước 2: Tạo file cấu hình hibernate.cfg.xml
- Bước 3: Tạo các POJO (Lớp persistent)
- Bước 4: Tạo các file mapping <POJO>.hbm.xml
- Bước 5: Khai báo các file mapping vào hibernate.cfg.xml
- Bước 6: Xây dựng lớp HibernateUtil
- Bước 7: Sử dụng

Tạo cơ sở dữ liệu

MASV	HOTEN	GIOITINH	NOISINH
K1101	Nguyen Van A	Nam	TpHCM
K1102	Tran Ngoc Han	Nu	Kien Giang
K1103	Ha Duy Lap	Nam	Nghe An
K1104	Tran Ngoc Linh	Nu	Tay Ninh
K1105	Tran Minh Long	Nam	TpHCM
K1106	Le Nhat Minh	Nam	TpHCM
K1107	Nguyen Nhu Nhut	Nam	Ha Noi
K1108	Nguyen Manh Tam	Nam	Kien Giang
K1109	Phan Thi Thanh Tam	Nu	Vinh Long

Cấu hình hibernate.cfg.xml

- Được dùng đề khai báo cấu hình các thông tin liên quan đến CSDL: url, driver_class, username, password, pool_size, autocommit.
- Khai báo nơi mà chứa file ánh xạ giữa các lớp thực thể tới các bảng trong CSDL

Cấu hình hibernate.cfg.xml (tt)

- Các thuộc tính cấu hình cơ bản:
  - **hibernate.dialect**: Cho phép hibernate tối ưu hóa SQL cho thích hợp với CSDL quan hệ đặc biệt.
  - **hibernate.connection.driver_class:** lớp driver class.
  - **hibernate.connection.url:** url để kết nối tới CSDL thông qua driver.
  - **hibernate.connection.username:** uername để truy nhập vào CSDL.
  - **hibernate.connection.password:** password để truy nhập vào CSDL.
  - **hibernate.connection.pool_size:** số lượng tối đa các kết nối tới CSDL trong 1 thời điểm.

Cấu hình hibernate.cfg.xml (tt)

- Loại **hibernate.dialect**: của một số CSDL phổ biến:

DB2: org.hibernate.dialect.DB2Dialect
Microsoft SQL Server 2005: org.hibernate.dialect.SQLServer2005Dialect
Microsoft SQL Server 2008: org.hibernate.dialect.SQLServer2008Dialect
MySQL: org.hibernate.dialect.MySQLDialect
Oracle (any version): org.hibernate.dialect.OracleDialect
Oracle 11g: org.hibernate.dialect.Oracle10gDialect
Oracle 10g: org.hibernate.dialect.Oracle10gDialect
Oracle 9i: org.hibernate.dialect.Oracle9iDialect
PostgreSQL: org.hibernate.dialect.PostgreSQLDialect
SAP DB: org.hibernate.dialect.SAPDBDialect
Sybase: org.hibernate.dialect.SybaseDialect
Sybase Anywhere: org.hibernate.dialect.SybaseAnywhereDialect

Cấu hình hibernate.cfg.xml (tt)


```
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <!-- Kết nối Database -->
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/qlsv?autoReco
nnect&amp;useUnicode=true&amp;characterEncoding=utf8</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">sang</property>

        <!-- List of XML mapping files -->
        <mapping resource="SinhVien.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Tạo POJO (Lớp persistent)

- Phải có 1 **phương thức khởi tạo mặc định** (phương thức khởi tạo không có tham số)
- Tất cả các thuộc tính trong lớp cần **đặt "private" hoặc protected, hoặc default** (Hibernate không xử lý các thuộc tính để public)
- Chứa các phương thức **set* , get*, is*** tương tự như 1 lớp **JavaBean**
- **Class không nên để final**
- Các phương thức **không nên để *public final*, chỉ nên để *public***
- Các lớp này **không được kế thừa (extend) các lớp đặc biệt khác** mà **chỉ có thể kế thừa các lớp persistent khác.**

VD: Tạo POJO SinhVien


```
public class SinhVien {
    private String maSV;
    private String hoTen;
    private String gioiTinh;
    private String noiSinh;
    public SinhVien(){    }
    public SinhVien(String maSV){
        this.maSV = maSV;    }
    public SinhVien(String maSV,
String hoTen, String gioiTinh,
String noiSinh){
        this.maSV = maSV;
        this.hoTen = hoTen;
        this.gioiTinh = gioiTinh;
        this.noiSinh = noiSinh;  }
    public void setMaSV(String
maSV){
        this.maSV = maSV;   }
    public String getMaSV(){
        return maSV;    }
```

```
 public void setHoTen(String
hoTen){
        this.hoTen = hoTen;    }
    public String getHoTen(){
        return hoTen;    }
    public void
setGioiTinh(String gioiTinh){
        this.gioiTinh = gioiTinh;
}
    public String getGioiTinh(){
        return gioiTinh;    }
    public void setNoiSinh(String
noiSinh){
        this.noiSinh = noiSinh;
}
    public String getNoiSinh(){
        return noiSinh;
}
}
```

Tạo file mapping ###.hbm.xml

Các thành phần quan trọng trong file mapping:

<hibernate-mapping>: phần tử gốc của tài liệu ánh xạ(mapping) hibernate, nó chứa các phần tử khác và kết thúc với **</hibernate-mapping>.**

<class> Ánh xạ các lớp đối tượng với bảng trong CSDL.

- name : tên của lớp đối tượng dữ liệu
- table : tên bảng trong CSDL tương ứng với lớp của thuộc tính name

<id> dùng để định danh id duy nhất của lớp đối tượng. Thường ánh xạ với khóa chính trong bảng CSDL. Có các thuộc tính sau :

- name : tên thuộc tính được sử dụng trong lớp persistent
- column : cột trong bảng CSDL để lưu khóa chính.
- type : Kiểu dữ liệu của cột

Tạo file mapping ###.hbm.xml (tt)

<generator> tạo ra giá trị ID khi thêm bản ghi mới vào CSDL bằng Hibernate. Một số phương thức tạo giá trị ID:

- increment : sử dụng để tạo ID, chỉ dùng cho kiểu long, int, short .
- sequence : Hibernate sử dụng chuỗi để tạo khóa tự động. Thường được sử dụng với HQTCSDL : DB2, PostgreSQL, Oracle, SAP DB databases.
- assigned : đăng ký 1 phương thức khởi tạo ID , phương thức này được tạo ra bởi người phát triển ứng dụng.
- identity: ID tự động tạo ra bởi CSDL.

<property> sử dụng các thuộc tính giống như thẻ ID nhưng không có thẻ con <generator>. Được dùng để ánh xạ các cột bình thường (không phải khóa chính).

Tạo file mapping ###.hbm.xml (tt)


```
<hibernate-mapping>
    <class name="hibernateexample.SinhVien" table="sinhvien">
        <id name="maSV" type="string">
            <column length="5" name="MaSV"/>
            <generator class="assigned" />
        </id>
        <property name="hoTen" type="string">
            <column length="25" name="HoTen" />
        </property>
        <property name="gioiTinh" type="string">
            <column length="4" name="GioiTinh" />
        </property>
        <property name="noiSinh" type="string">
            <column length="20" name="NoiSinh" />
        </property>
    </class>

</hibernate-mapping>
```

Kiểu dữ liệu trong hibernate

Các kiểu dữ liệu cơ bản

Kiểu Hibernate	Kiểu Java	Kiểu SQL
integer	int hoặc Integer	INTEGER
long	long hoặc Long	BIGINT
short	short hoặc Short	SMALLINT
float	float hoặc Float	FLOAT
double	double hoặc Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	String	CHAR(1)
string	String	VARCHAR
byte	byte hoặc Byte	TINYINT
boolean	boolean hoặc Boolean	BIT
yes/no	boolean hoặc Boolean	CHAR(1) (‘Y’ or ‘N’)
true/false	boolean hoặc Boolean	CHAR(1) (‘T’ or ‘F’)

Kiểu dữ liệu trong hibernate (tt)

Các kiểu dữ liệu thời gian

Kiểu Hibernate	Kiểu Java	Kiểu SQL
date	java.util.Date hoặc java.sql.Date	DATE
time	java.util.Date hoặc java.sql.Time	TIME
timestamp	java.util.Date hoặc java.sql.Timestamp	TIMESTAMP
calendar	Calendar	TIMESTAMP
calendar_date	Calendar	DATE

Khai báo mapping vào ###.cfg.xml

```
<hibernate-configuration>
    <session-factory>
        <property
        …….
        </property>

        <!-- List of XML mapping files -->
        <mapping resource="SinhVien.hbm.xml"/>
        <mapping jar="application.jar"/>
    </session-factory>
</hibernate-configuration>
```

Sử dụng - Tạo kết nối


```
public class HibernateSessionFactory {
    private static SessionFactory sesstionFac;
    static {
        try {
            sesstionFac = new
Configuration().configure().buildSessionFactory();
        }
        catch(Exception e){
            System.out.println("Initial SessionFactory creation
failed." + e.getMessage());
        }
    }

    public static SessionFactory getSessionFactory(){
        return sesstionFac;
    }
}
```

Lấy danh sách sinh viên


```
public static List<SinhVien> getDSSinhVien(){
        List<SinhVien> dsSV = null;
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        try{
            String sql = "select sv from SinhVien sv";
            Query query = session.createQuery(sql);
            dsSV = query.list();
        }
        catch(HibernateException e){
            System.err.println(e);
        }
        finally{
            session.flush();
            session.close();
        }
        return dsSV;
    }
```

Lấy thông tin sinh viên


```
public static SinhVien getSinhVien(String maSV){
        SinhVien sv = null;
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        try{
            sv = (SinhVien) session.get(SinhVien.class, maSV);
        }
        catch(HibernateException e){
            System.err.println(e);
        }
        finally{
            session.flush();
            session.close();
        }
        return sv;
    }
```

Thêm sinh viên


```
public static boolean themSV(SinhVien sv){
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        if (SinhVienDAO.getSinhVien(sv.getMaSV())!=null) {
            return false;
        }
        Transaction trans = null;
        try{
            trans = session.beginTransaction();
            session.save(sv);
            trans.commit();
        }catch(HibernateException ex) {
            //Log the exception
            trans.rollback();
            System.err.println(ex);
        }
        finally {
            session.close();
        }
        return true;
    }
```

Xóa sinh viên


```
public static boolean xoaSV(String maSV){
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        SinhVien sv = SinhVienDAO.getSinhVien(maSV);
        if(sv == null){
            return false;
        }
        Transaction trans = null;
        try {
            trans = session.beginTransaction();
            session.delete(sv);
            trans.commit();
        } catch (HibernateException ex) {
            //Log the exception
            trans.rollback();
            System.err.println(ex);
        } finally {
            session.close();
        }
        return true;
    }
```

Sửa sinh viên

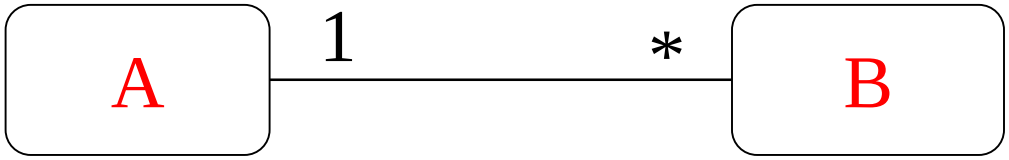

```
public static boolean capNhatSV(SinhVien sv){
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        if (getSinhVien(sv.getMaSV()) == null){
            return false;
        }
        Transaction trans = null;
        try {
            trans = session.beginTransaction();
            session.update(sv);
            trans.commit();
        } catch (HibernateException ex) {
            //Log the exception
            trans.rollback();
            System.err.println(ex);
        } finally {
            session.close();
        }
        return true;
    }
```

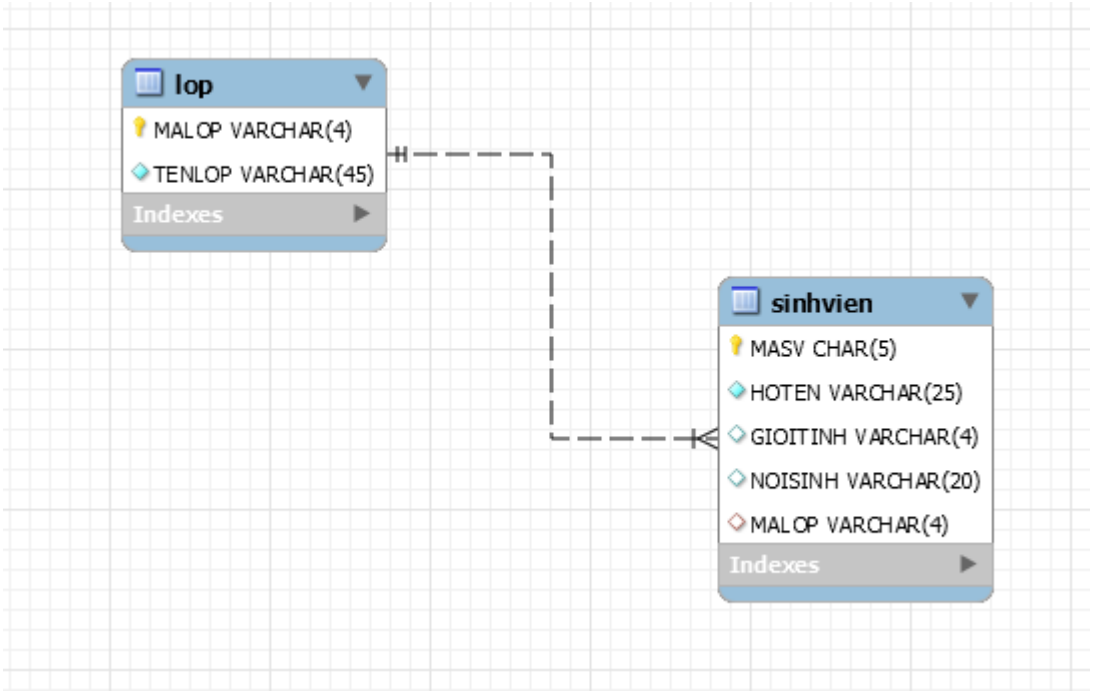
Hibernate mapping

- Mối quan hệ nhiều một – many to one
- Mối quan hệ một nhiều – one to many
- Mối quan hệ một một – one to one
- Mối quan hệ nhiều nhiều – many to many

Many to one

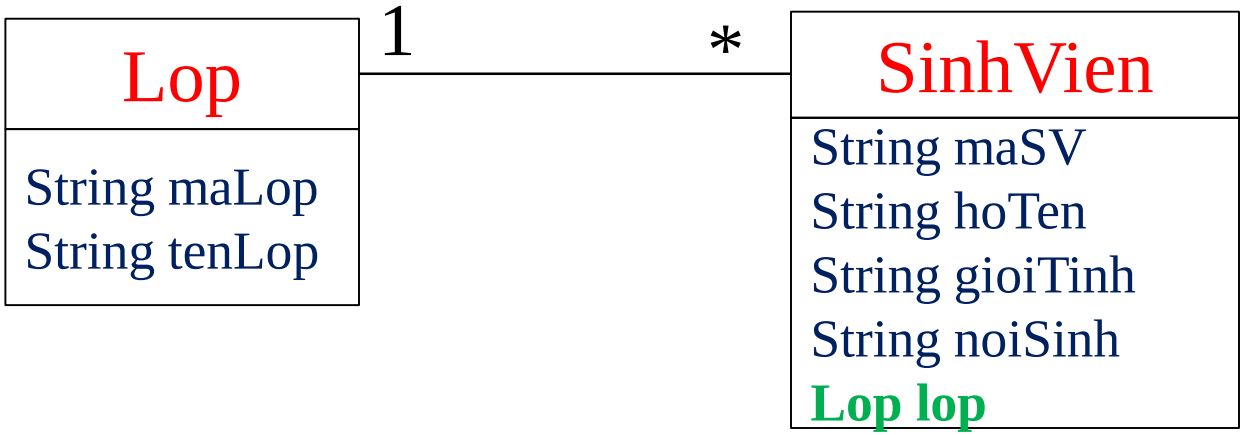
- Mối quan hệ nhiều một (many to one)





Many to one (tt)

- Mối quan hệ nhiều một (many to one)



Tạo POJO Lop


```
public class Lop {
    private String maLop;
    private String tenLop;
    public Lop(){    }
    public Lop(String maLop, String tenLop){
        this.maLop = maLop;
        this.tenLop = tenLop;}

    //Các hàm Getter và Setter
```

Tạo file mapping Lop.hbm.xml


```
<hibernate-mapping>
    <class name="hibernateexample.Lop" table="lop">
        <id name="maLop" type="string">
            <column name="MALOP" length="4" />
            <generator class="assigned" />
        </id>
        <property name="tenLop" type="string">
            <column name="TENLOP" length="45"  />
        </property>
    </class>
</hibernate-mapping>
```

File mapping SinhVien.hbm.xml


```
<hibernate-mapping>
    <class name="hibernateexample.SinhVien" table="sinhvien">
        <id name="maSV" type="string">
            <column length="5" name="MaSV"/>
            <generator class="assigned" />
        </id>
        <property name="hoTen" type="string">
            <column length="25" name="HoTen" />
        </property>
        <property name="gioiTinh" type="string">
            <column length="4" name="GioiTinh" />
        </property>
        <property name="noiSinh" type="string">
            <column length="20" name="NoiSinh" />
        </property>
        <many-to-one class="hibernateexample.Lop" name="lop"
fetch="select">
            <column name="MALOP" length="4" not-null="true" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

Lấy thông tin sinh viên và tên lớp


```
public static SinhVien getSinhVien(String maSV){
        SinhVien sv = null;
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        try{
            sv = (SinhVien) session.get(SinhVien.class, maSV);
        }
        catch(HibernateException e){
            System.err.println(e);
        }
        finally{
            session.flush();
            session.close();
        }
        return sv;
    }
```

Lấy thông tin sinh viên và tên lớp (tt)

```
public static void main(String[] args) {
     SinhVien sv = SinhVienDAO.getSinhVien("K1213");
     if (sv != null){
          System.out.println("MSSV: "+sv.getMaSV());
          System.out.println("Họ và tên: "+sv.getHoTen());
          System.out.println("Giới tính: " + sv.getGioiTinh());
          System.out.println("Nơi sinh: "+ sv.getNoiSinh());
          Lop lop = sv.getLop();
          System.out.println("Mã lớp: " + lop.getMaLop() + " Tên lớp: "
     + lop.getTenLop());
     }
     else {
          System.out.println("Không có sinh viên trong lớp");
     }
}
```

```
Output - HibernateExample (run) #3
   MSSV: K1213
   Họ và tên: Mai Lớp Không Nghỉ
   Giới tính: Nam
   Nơi sinh: UIT
Exception in thread "main" org.hibernate.LazyInitializationException: could not initialize proxy - no Session
        at org.hibernate.proxy.AbstractLazyInitializer.initialize(AbstractLazyInitializer.java:164)
        at org.hibernate.proxy.AbstractLazyInitializer.getImplementation(AbstractLazyInitializer.java:285)
        at org.hibernate.proxy.pojo.javassist.JavassistLazyInitializer.invoke(JavassistLazyInitializer.java:185)
        at hibernateexample.Lop_$$_jvstf18_1.getTenLop(Lop_$$_jvstf18_1.java)
        at hibernateexample.HibernateExample.main(HibernateExample.java:42)
```

Lý do

- Khi truy xuất các thuộc tính bên trong **Lop** (ngoài mã lớp) từ đối tượng **SinhVien** bên trong **session** thì hoàn toàn được.
- Nhưng khi truy xuất các thuộc tính bên trong **Lop** (ngoài mã lớp) từ đối tượng **SinhVien** bên ngoài **session** thì **exception** sẽ xảy ra.
- Trong Hibernate cơ chế này được gọi là **Lazy Initialization**

Lấy thông tin sinh viên và tên lớp


```
public static SinhVien getSinhVien(String maSV){
        SinhVien sv = null;
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        try{
            sv = (SinhVien) session.get(SinhVien.class, maSV);
            Lop lop = sv.getLop();
            System.out.println("Mã lớp: " + lop.getMaLop() + " Tên
lớp: " + lop.getTenLop());
        }
        catch(HibernateException e){
            System.err.println(e);
        }
        finally{
            session.flush();
            session.close();
        }
        return sv;
    }
```

Lazy Initialization

- Trong Hibernate, Lazy Initialization giúp
  - Tránh các câu truy vấn cơ sở dữ liệu không cần thiết
  - Chưa load dữ liệu tương ứng vào bảng chỉ load dữ liệu cột id lên
  - Gia tăng hiệu suất thực thi
  - Lazy mặc định có giá trị là **true**

Lấy thông tin sinh viên và tên lớp

- Viết phương thức để lấy thông tin sinh viên và tên lớp của sinh viên đó???

Cách 1: Lấy tên lớp từ mã lớp


```
public static void main(String[] args) {
    SinhVien sv = SinhVienDAO.getSinhVien("K1213");
    if (sv != null){
        System.out.println("MSSV: "+sv.getMaSV());
        System.out.println("Họ và tên: "+sv.getHoTen());
        System.out.println("Giới tính: " + sv.getGioiTinh());
        System.out.println("Nơi sinh: "+ sv.getNoiSinh());
        String maLop = sv.getLop().getMaLop();
        Lop lop = LopDAO.getTenLop(maLop);
    }
    else {
        System.out.println("Không có sinh viên trong lớp");
    }
}
```

Cách 2: Sử dụng Hibernate.initialize


```
public static SinhVien getSinhVien(String maSV){
        SinhVien sv = null;
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        try{
            sv = (SinhVien) session.get(SinhVien.class, maSV);
            Hibernate.initialize(sv.getLop());
        }
        catch(HibernateException e){
            System.err.println(e);
        }
        finally{
            session.flush();
            session.close();
        }
        return sv;
    }
```

Cách 2: Sử dụng Hibernate.initialize (tt)

```
public static void main(String[] args) {
    SinhVien sv = SinhVienDAO.getSinhVien("K1213");
    if (sv != null){
        System.out.println("MSSV: "+sv.getMaSV());
        System.out.println("Họ và tên: "+sv.getHoTen());
        System.out.println("Giới tính: " + sv.getGioiTinh());
        System.out.println("Nơi sinh: "+ sv.getNoiSinh());
        Lop lop = sv.getLop();
        System.out.println("Mã lớp: " + lop.getMaLop() + " Tên lớp: "
    + lop.getTenLop());
    }
    else {
        System.out.println("Không có sinh viên trong lớp");
    }
}
```

Cách 3: Điều chỉnh thuộc tính lazy trong SinhVien.hbm.xml


```
<hibernate-mapping>
    <class name="hibernateexample.SinhVien" table="sinhvien">
        <id name="maSV" type="string">
            <column length="5" name="MaSV"/>
            <generator class="assigned" />
        </id>
        <property name="hoTen" type="string">
            <column length="25" name="HoTen" />
        </property>
        <property name="gioiTinh" type="string">
            <column length="4" name="GioiTinh" />
        </property>
        <property name="noiSinh" type="string">
            <column length="20" name="NoiSinh" />
        </property>
        <many-to-one class="hibernateexample.Lop" name="lop"
fetch="select" lazy="false">
            <column name="MALOP" length="4" not-null="true" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

Cách 3: Điều chỉnh thuộc tính lazy trong SinhVien.hbm.xml (tt)

```
public static SinhVien getSinhVien(String maSV){
        SinhVien sv = null;
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        try{
            sv = (SinhVien) session.get(SinhVien.class, maSV);
        }
        catch(HibernateException e){
            System.err.println(e);
        }
        finally{
            session.flush();
            session.close();
        }
        return sv;
    }
```

Cách 3: Điều chỉnh thuộc tính lazy trong SinhVien.hbm.xml (tt)


```
public static void main(String[] args) {
    SinhVien sv = SinhVienDAO.getSinhVien("K1213");
    if (sv != null){
        System.out.println("MSSV: "+sv.getMaSV());
        System.out.println("Họ và tên: "+sv.getHoTen());
        System.out.println("Giới tính: " + sv.getGioiTinh());
        System.out.println("Nơi sinh: "+ sv.getNoiSinh());
        Lop lop = sv.getLop();
        System.out.println("Mã lớp: " + lop.getMaLop() + " Tên lớp: "
    + lop.getTenLop());
    }
    else {
        System.out.println("Không có sinh viên trong lớp");
    }
}
```

Fetch

```
<hibernate-mapping>
    <class name="hibernateexample.SinhVien" table="sinhvien">
        ……….
        <many-to-one class="hibernateexample.Lop" name="lop"
fetch="select" lazy="false">
            <column name="MALOP" length="4" not-null="true" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

- Để lazy initialization bằng **false** cùng với cơ chế nạp **fetch = "select"** , khi đó sẽ có 2 câu lệnh select được phát sinh để truy vấn lấy thông tin đối tượng **SinhVien** và đối tượng **Lop**.
- Điều này có thể không hiệu quả bởi vì cần truy xuất vào CSDL và thực hiện truy vấn hai lần.

Fetch (tt)

```
<hibernate-mapping>
    <class name="hibernateexample.SinhVien" table="sinhvien">
        ……….
        <many-to-one class="hibernateexample.Lop" name="lop"
fetch="join" lazy="false">
            <column name="MALOP" length="4" not-null="true" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

- Thay cơ chế **fetch = "select"** thành **fetch = "join"**, khi đó 1 câu lệnh select được phát sinh duy nhất bằng cách kết bảng để truy vấn
- fetch = "join" sẽ hiệu quả bởi vì chỉ cần truy xuất vào cơ sở dữ liệu và thực hiện truy vấn một lần.

Cascade

- None
- Save-update

Cascade - none


```
<hibernate-mapping>
    <class name="hibernateexample.SinhVien" table="sinhvien">
        ……….
        <many-to-one class="hibernateexample.Lop" name="lop"
fetch="select" lazy="false" cascade="none">
            <column name="MALOP" length="4" not-null="true" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

Cascade - none (tt)

```
public static boolean themSV(SinhVien sv){
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        if (SinhVienDAO.getSinhVien(sv.getMaSV())!=null) {
            return false;
        }
         boolean kq = true;
        Transaction trans = null;
        try{
            trans = session.beginTransaction();
            session.save(sv);
            trans.commit();
        }catch(HibernateException ex) {
            //Log the exception
            trans.rollback();
            System.err.println(ex);
             kq = false;
        }
        finally {
            session.close();          }
        return kq;
    }
```

Cascade - none (tt)

```
public static void main(String[] args){
        SinhVien sv = new SinhVien();
        sv.setMaSV("K1214");
        sv.setHoTen("Mai Lớp Nghỉ");
        sv.setGioiTinh("Nam");
        sv.setNoiSinh("UIT");
        Lop lop = new Lop("K14", "Lop 4 Khoa 1");
        sv.setLop(lop);
        if (SinhVienDAO.themSV(sv)){
            System.out.print("Thêm thành công");
        }
        else{
            System.out.print("Thêm không thành công");
        }
```

- Thêm không thành công vì: Chưa có mã lớp trong CSDL

Cascade - save-update

```
<hibernate-mapping>
    <class name="hibernateexample.SinhVien" table="sinhvien">
        ……….
        <many-to-one class="hibernateexample.Lop" name="lop"
fetch="select" lazy="false" cascade="save-update">
            <column name="MALOP" length="4" not-null="true" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

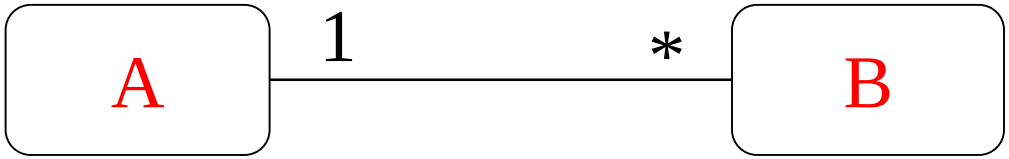
Cascade - save-update (tt)

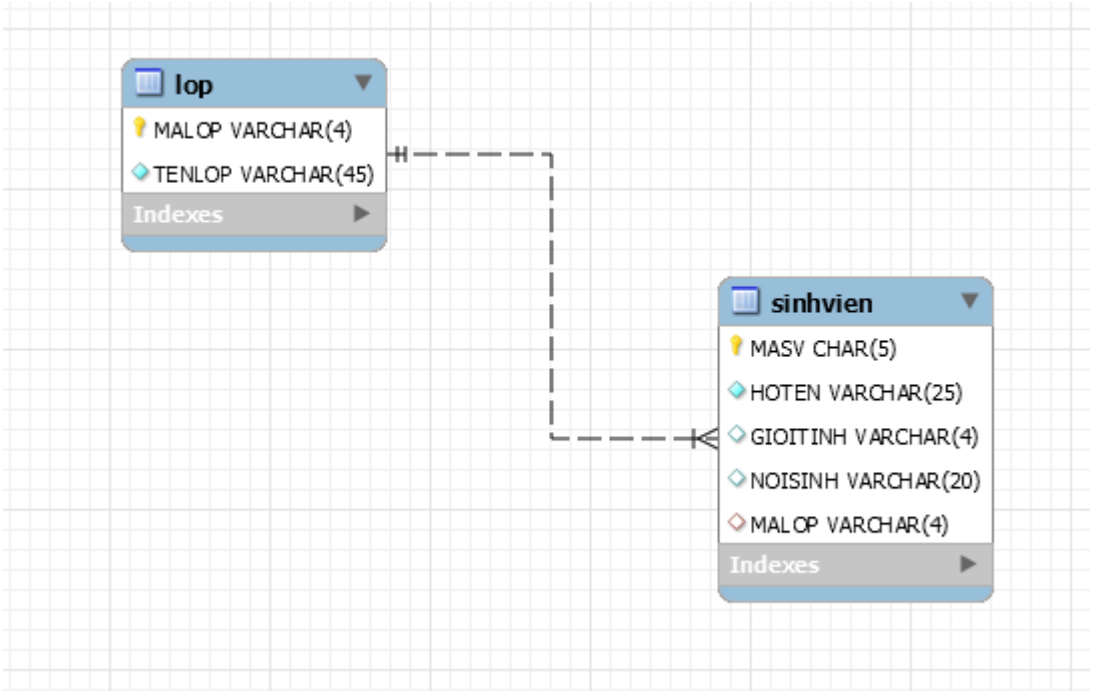
```
public static void main(String[] args){
        SinhVien sv = new SinhVien();
        sv.setMaSV("K1214");
        sv.setHoTen("Mai Lớp Nghỉ");
        sv.setGioiTinh("Nam");
        sv.setNoiSinh("UIT");
        Lop lop = new Lop("K14", "Lop 4 Khoa 1");
        sv.setLop(lop);
        if (SinhVienDAO.themSV(sv)){
            System.out.print("Thêm thành công");
        }
        else{
            System.out.print("Thêm không thành công");
        }
```

- Nếu lớp chưa có trong CSDL, hibernate sẽ tạo lớp vào CSDL

One to many

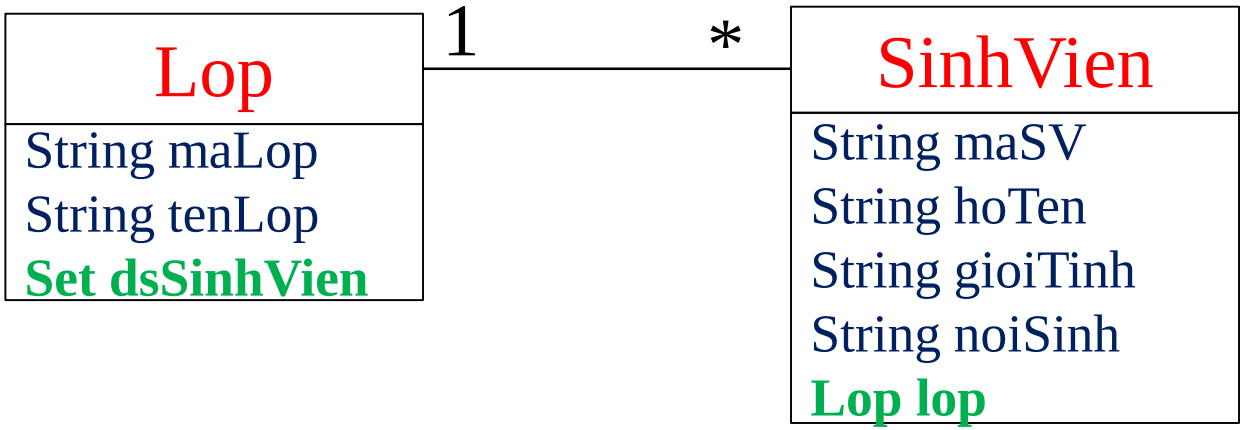
- Mối quan hệ một nhiều (one to many)





One to many (tt)

- Mối quan hệ một nhiều (one to many)



Tạo POJO Lop

```
public class Lop {
    private String maLop;
    private String tenLop;
    private Set<SinhVien> dsSinhVien = new HashSet<SinhVien>(0);

    public Lop(){    }
    public Lop(String maLop, String tenLop){
        this.maLop = maLop;
        this.tenLop = tenLop;}

    //Các hàm Getter và Setter
```

Tạo file mapping Lop.hbm.xml


```
<hibernate-mapping>
    <class name="hibernateexample.Lop" table="lop">
        <id name="maLop" type="string">
            <column name="MALOP" length="4" />
            <generator class="assigned" />
        </id>
        <property name="tenLop" type="string">
            <column name="TENLOP" length="45"  />
        </property>
       <set inverse="true" name ="dsSinhVien" fetch="join"
lazy="false" >
            <key>
                <column length="4" name="MALOP" not-null="true" />
            </key>
            <one-to-many class="hibernateexample.SinhVien" />
        </set>
    </class>
</hibernate-mapping>
```

Lấy thông tin lớp


```
public class LopDAO {
    public static Lop getLop(String maLop){
        Lop lop = null;
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        try{
            lop = (Lop) session.get(Lop.class, maLop);
        }catch(HibernateException e){
            System.err.println(e);
        }
        finally{
            session.flush();
            session.close();
        }
        return lop;

    }
}
```

Lấy thông tin sinh viên theo lớp(tt)


```
public static void main(String[] args){
      //Lấy thông tin danh sách sinh viên theo lớp
      Lop lop = LopDAO.getLop("K11");
      System.out.println("Mã lớp: "+lop.getMaLop());
      System.out.println("Tên lớp: "+lop.getTenLop());
      Iterator<SinhVien> dsSV = lop.getDsSinhVien().iterator();
      while (dsSV.hasNext()){
         SinhVien sv = dsSV.next();
         System.out.println("MSSV: "+sv.getMaSV());
         System.out.println("Họ và tên: "+sv.getHoTen());
         System.out.println("Giới tính: " + sv.getGioiTinh());
         System.out.println("Nơi sinh: "+ sv.getNoiSinh());
      }
}
```

Cascade

- None
- delete

Cascade - none

```
<hibernate-mapping>
    <class name="hibernateexample.Lop" table="lop">
        ……….
        <set inverse="true" name ="dsSinhVien" fetch="join"
lazy="false" cascade="none" >
            <key>
                <column length="4" name="MALOP" not-null="true" />
            </key>
            <one-to-many class="hibernateexample.SinhVien" />
        </set>
    </class>
</hibernate-mapping>
```

Cascade - none (tt)


```
public static boolean xoaLop(String maLop){
        Session session =
HibernateSessionFactory.getSessionFactory().openSession();
        Lop lop = LopDAO.getLop(maLop);
        if(lop == null){
            return false;
        }
        boolean kq = true;
        Transaction trans = null;
        try {
            trans = session.beginTransaction();
            session.delete(lop);
            trans.commit();
        } catch (HibernateException ex) {
            //Log the exception
            trans.rollback();
            System.err.println(ex);
            kq = false;
        } finally {
            session.close();
        }
        return kq;     }
```

Cascade - none (tt)

```
public static void main(String[] args){
        //Xóa lớp
        if (LopDAO.xoaLop("K14")) {
            System.out.println("Xóa thành công");
        }
        else{
            System.out.println("Xóa không thành công");
        }
```

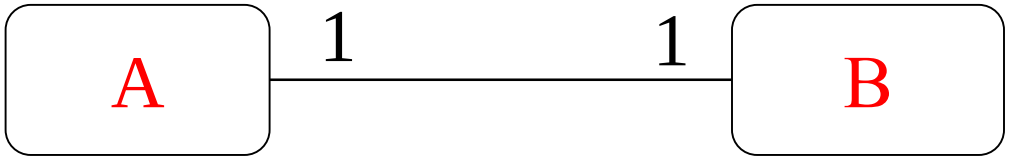
- Xóa không thành công

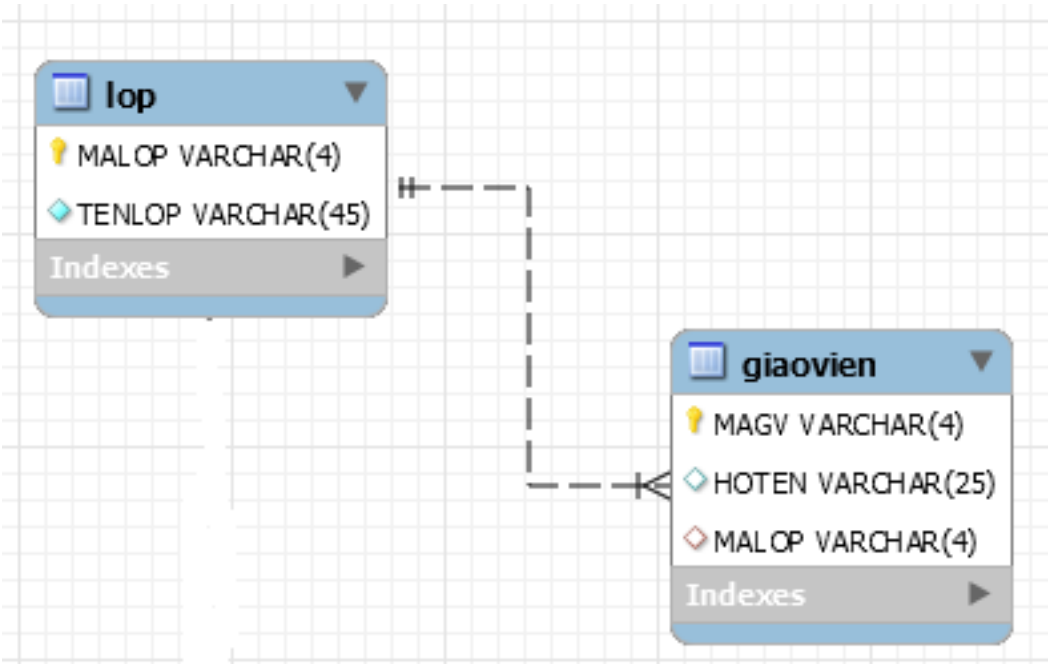
Cascade - delete

```
<hibernate-mapping>
    <class name="hibernateexample.Lop" table="lop">
        ……….
        <set inverse="true" name ="dsSinhVien" fetch="join"
lazy="false" cascade="delete" >
            <key>
                <column length="4" name="MALOP" not-null="true" />
            </key>
            <one-to-many class="hibernateexample.SinhVien" />
        </set>
    </class>
</hibernate-mapping>
```

one to one

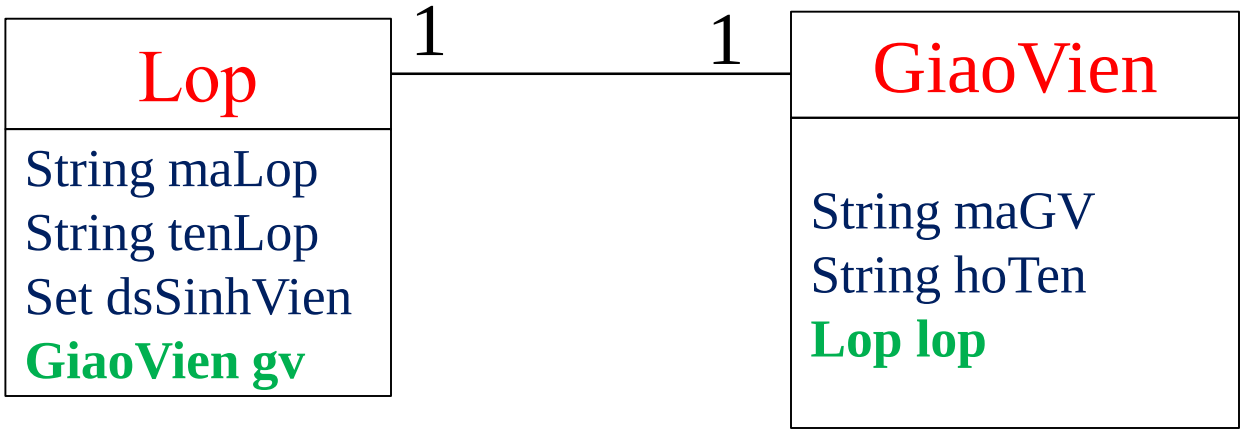
- Mối quan hệ một một (one to one)





one to one (tt)

- Mối quan hệ nhiều một (many to one)



one to one (tt)

- Mapping mối quan hệ một-một giống như mapping trong mối quan hệ nhiều-một
  - Nhưng thêm thuộc tính **unique="true"**
- Có thể khai báo sử dụng thuộc tính
  - lazy
  - fetch
  - cascade

Tạo file mapping Lop.hbm.xml

```
<hibernate-mapping>
    <class name="hibernateexample.Lop" table="lop">
        ……..
      <one-to-one name="gv" class="hibernateexample.GiaoVien"
property-ref="GiaoVien" cascade="save-update,delete"/>
    </class>
</hibernate-mapping>
```

Tạo file mapping GiaoVien.hbm.xml


```
<hibernate-mapping>
  <class name="hibernateexample.GiaoVien" table="giaovien"/>
        <id name="maGV" type="string">
            <column name="MAGV" length="4" />
            <generator class="assigned" />
        </id>
        <property name="hoten" type="string">
            <column name="HOTEN" length="25"  />
        </property>
        <many-to-one class="hibernateexample.Lop" name="lop"
fetch="join" lazy="false" cascade="save-update,delete">
            <column name="MALOP" length="4" not-null="true"
unique="true" />
        </many-to-one>
</hibernate-mapping>
```

Hibernate

- Lợi ích khi dùng Hibernate (Hibernate Query Language)
  - Tìm kiếm và sắp xếp nhanh.
  - Làm việc được với dữ liệu lớn.
  - Làm việc trên nhóm dữ liệu.
  - Joining, aggregating.
  - Chia sẽ nhiều người dùng và nhiều vùng.
  - Giải quyết tương tranh (Transaction)
  - Hỗ trợ cho nhiều ứng dụng.
  - Bảo đảm toàn vẹn.
  - Ràng buộc nhiều cấp độ.
  - Tách biệt giao tác.

Q & A