HIBERNATE

ID

GVHD: Thầy Nguyễn Hoàng Anh

Sinh viên: 0812463 – Hồ Văn Tấn
0812505 – Sử Bá Thuần

Các vấn đề trình bày

1. Giới thiệu.
2. ID gán trực tiếp.
3. ID tự động sinh.
4. Demo + Hỏi đáp.

GIỚI THIỆU

IDENTIFIER (ID)

- Các lớp ánh xạ phải được khai báo cột khóa chính của bảng CSDL tương ứng.
- Dùng để xác định các đối tượng một cách rõ ràng.
- Trong Hibernate hỗ trợ 2 loại ID chính:
  - ID được gán trực tiếp.
  - ID tự động sinh.

ID GÁN TRỰC TIẾP

ID GÁN TRỰC TIẾP

- Ít được sử dụng.
- Thường có ý nghĩa trong nghiệp vụ.
- Có thể thay đổi trong một số trường hợp.
- Nếu quên nhập sẽ gây ra lỗi ràng buộc.

ID GÁN TRỰC TIẾP

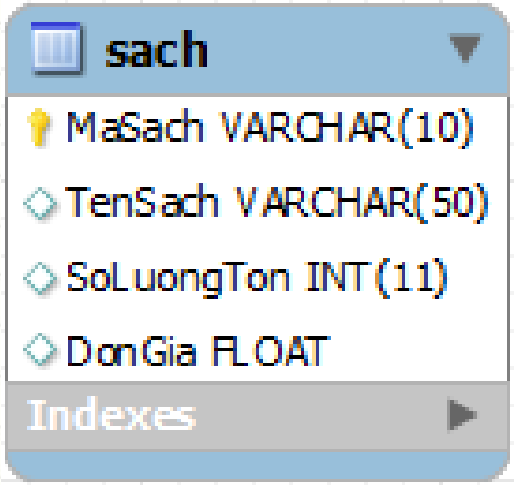
- Có 2 loại chính:
  - ID đơn.
  - ID phức hợp (Composite).

ID ĐƠN

- **Mapping**

ID Đơn

- Là ID chỉ chứa 1 phần tử.
- Ví dụ: Có csdl và SachPojo như sau:




```
1  private String maSach;
2  private String tenSach;
3  private int soLuongTon;
4  private float donGia;
```

Mapping ID – Sach.hbn.xml

```
1  <id name="maSach" type="string" length="10" column="MaSach"/>
```

- name: thuộc tính trong pojo.
- type: kiểu của thuộc tính.

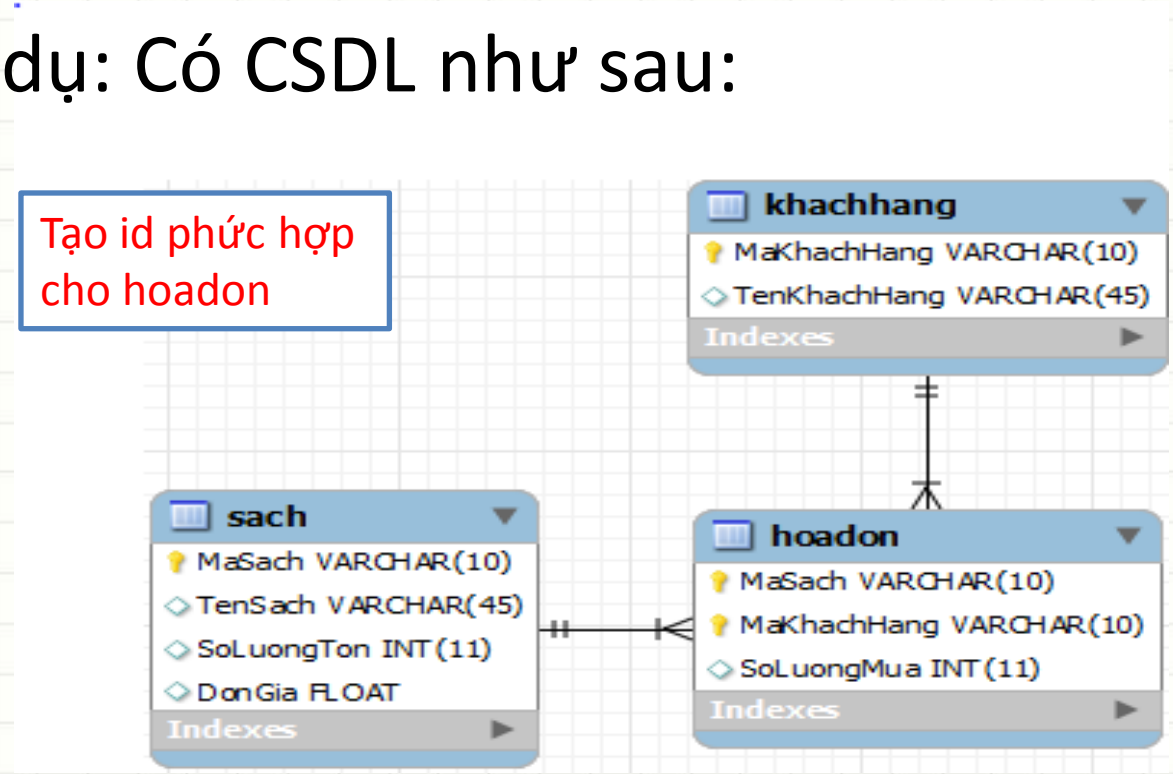
  (Là kiểu của Hibernate)
- length: độ dài của dữ liệu.
- column: cột khóa chính tương ứng trong CSDL.

ID PHỨC HỢP (COMPOSITE ID)

- Chiến lược
- Mô hình
- Mapping
- Ưu – nhược điểm

ID Phức hợp

- Là ID được tạo nên bao gồm nhiều thành phần.
- Ví dụ: Có CSDL như sau:



Các bước tạo 1 ID phức hợp

- Gom tất cả các thành phần làm khóa lại tạo thành 1 class.
- Lớp này bắt buộc phải implements java.io.Serializable

```
1  public class IDHoaDon implements java.io.Serializable {
2      private String maSach;
3      private String maKhachHang;
4      //get & set & constructor
5  }
```

Các bước tạo 1 ID phức hợp

- Chỉnh lại HoaDonPOJO
- Lớp này không bắt buộc phải implements java.io.Serializable.

```
1  public class HoaDonPOJO implements java.io.Serializable {
2      private IDHoaDon maHoaDon;
3      private int soLuongMua;
4      //get & set & constructor
5  }
```

Các bước tạo 1 ID phức hợp

- Mapping trong HoaDon.hbm.xml

```
1   <hibernate-mapping>
2     <class name="pojo.HoaDonPOJO" table="hoadon">
3       <composite-id name="maHoaDon" class="pojo.IDHoaDon">
4         <key-property name="maSach" column="MaSach" length="10"
5   type="string" />
6         <key-property name="maKhachHang" column="MaKhachHang"
7   length="10" type="string" />
8       </composite-id>
9       <property name="soLuongMua" type="integer" column="SoLuongMua" />
10    </class>
11  </hibernate-mapping>
12
```

ID TỰ ĐỘNG SINH

ID tự động sinh

- Được sử dụng phổ biến.
- Tự động sinh ra 1 giá trị duy nhất cho mỗi đối tượng.
- Không có ý nghĩa trong nghiệp vụ.
- Không thay đổi.
- Hạn chế lỗi ràng buộc.

Khai báo

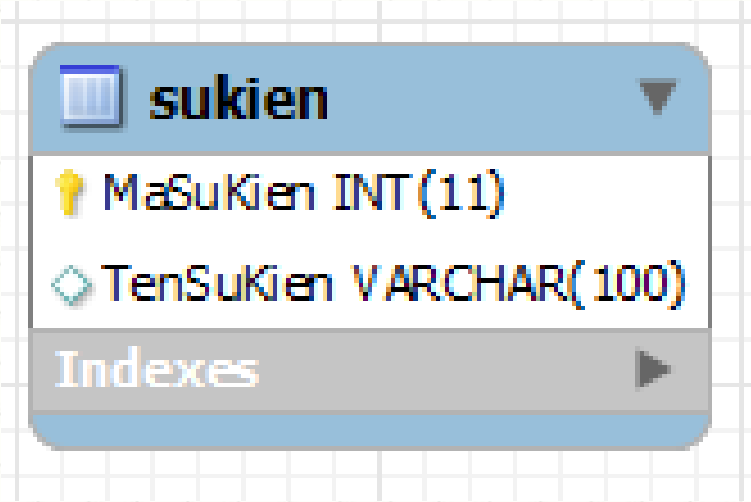
```
1  <id  name="…" type="…" >
2          <column name="…" not-null="true"/>
3          <generator class="[method]"/>
4  </id>
```

Các kiểu method thường dùng:

- increment
- identity
- native
- assigned
- hilo
- sequence
- uuid

increment

- Tự động sinh ra 1 giá trị duy nhất cho mỗi biến thể của class.
- Trả về kiểu int, long, hoặc short
- Ví dụ:



increment – SuKienPOJO.java

```
1  public class SuKienPOJO implements java.io.Serializable {
2      private int maSuKien;
3      private String tenSuKien;
4      //get & set & constructor
5      …
6  }
```

increment – SuKien.hbm.xml

```
1  <hibernate-mapping>
2     <class name="pojo.SuKienPOJO" table="sukien">
3        <id name="maSuKien" column="MaSuKien" type="integer" >
4           <generator class="increment"/>
5        </id>
6        <property name="tenSuKien" type="string" length="100"
7  column="TenSuKien" />
8     </class>
9  </hibernate-mapping>
```

increment – SuKienDAO.java

```
1   public static List<SuKienPOJO> layDsSuKien() {
2       List<SuKienPOJO> 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.SuKienPOJO").list();
9          trans.commit();
10      } catch (Exception ex) {
11         System.out.println(ex.getMessage());
12      }
13      return list;
14    }
```

increment – SuKienDAO.java

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

increment – App

```
1  List<SuKienPOJO> dsSuKien = dao.SuKienDAO.layDsSuKien();
2          for (int i = 0; i < dsSuKien.size(); ++i) {
3             System.out.println (dsSuKien.get(i).getMaSuKien() + "-" +
4  dsSuKien.get(i).getTenSuKien());
5          }
6
```

	MaSuKien	TenSuKien
▶	1	Ngày hội việc làm
	2	Ngày hội CNTT
*	NULL	NULL

```
1-Ngày hội việc làm
2-Ngày hội CNTT
```

increment – App

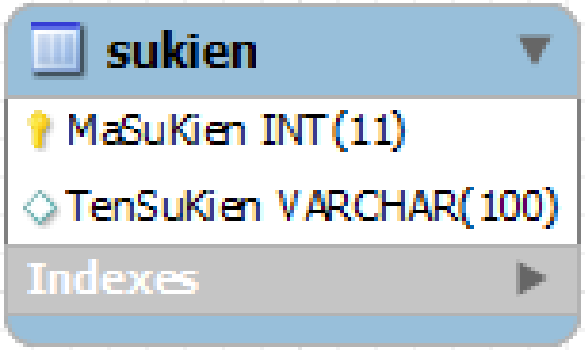
```
1  SuKienPOJO sk = new SuKienPOJO(“Hội chợ triển lãm");
2          boolean kq = dao.SuKienDAO.themSuKien(sk);
3          if (kq) {
4             System.out.print("Thêm thành công!");
5          } else {
6             System.out.print("Thêm thất bại!");
7          }
```

```
Thêm thành công!BUILD SUCCESSFUL (total time: 3 seconds)
```

	MaSuKien	TenSuKien
▶	1	Ngày hội việc làm
	2	Ngày hội CNTT
	3	Hội chợ triển lãm
*	NULL	NULL

identity

- Sử dụng trên các hệ quản trị có hỗ trợ identity như: MySQL, SQL Server, …
- Khi thiết kế CSDL, cần tạo một cột khóa chính tự động tăng.
- Trả về kiểu int, long, hoặc short
- Ví dụ: Lấy lại vd cũ với MaSuKien là 1 cột INT tự động tăng.



identity – SuKienPOJO.java

```
1  public class SuKienPOJO implements java.io.Serializable {
2      private int maSuKien;
3      private String tenSuKien;
4      //get & set & constructor
5      …
6  }
```

identity – SuKien.hbm.xml

```
1  <hibernate-mapping>
2     <class name="pojo.SuKienPOJO" table="sukien">
3        <id name="maSuKien" column="MaSuKien" type="integer" >
4           <generator class="increment"/>
5        </id>
6        <property name="tenSuKien" type="string" length="100"
7  column="TenSuKien" />
8     </class>
9  </hibernate-mapping>
```

identity – SuKienDAO.java

```
1   public static List<SuKienPOJO> layDsSuKien() {
2       List<SuKienPOJO> 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.SuKienPOJO").list();
9          trans.commit();
10      } catch (Exception ex) {
11         System.out.println(ex.getMessage());
12      }
13      return list;
14    }
```

identity – SuKienDAO.java

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

increment – App

```
1  List<SuKienPOJO> dsSuKien = dao.SuKienDAO.layDsSuKien();
2          for (int i = 0; i < dsSuKien.size(); ++i) {
3             System.out.println (dsSuKien.get(i).getMaSuKien() + "-" +
4  dsSuKien.get(i).getTenSuKien());
5          }
```

	MaSuKien	TenSuKien
▶	1	Ngày hội việc làm
	2	Ngày hội CNTT
	3	Hội chợ triển lãm
*	NULL	NULL

```
1-Ngày hội việc làm
2-Ngày hội CNTT
3-Hội chợ triển lãm
```

identity – App

```
1  SuKienPOJO sk = new SuKienPOJO("Dell day");
2          boolean kq = dao.SuKienDAO.themSuKien(sk);
3          if (kq) {
4             System.out.print("Thêm thành công!");
5          } else {
              System.out.print("Thêm th?t b?i!");
           }
```

```
Thêm thành công!BUILD SUCCESSFUL (total time: 3 seconds)
```

	MaSuKien	TenSuKien
▶	1	Ngày hội việc làm
	2	Ngày hội CNTT
	3	Hội chợ triển lãm
	4	Dell day
*	NULL	NULL

identity – App

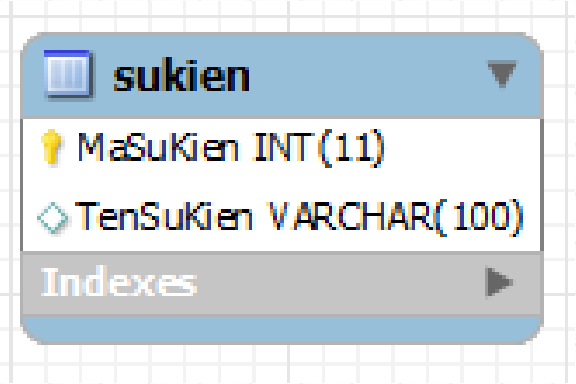
- Thực hiện xóa dòng 4 – Dell day trong CSDL rồi chạy lại đoạn code thêm ở trên.

```
Thêm thành công!BUILD SUCCESSFUL (total time: 3 seconds)
```

	MaSuKien	TenSuKien
▶	1	Ngày hội việc làm
	2	Ngày hội CNTT
	3	Hội chợ triển lãm
	5	Dell day
*	NULL	NULL

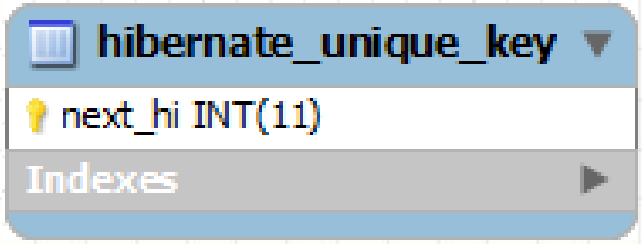
hilo

- Sử dụng thuật toán Hi/Lo để sinh ra các giá trị
- Cần cung cấp 1 table và column làm nguồn giá trị
- Chỉ sử dụng được cho các database chuyên biệt.
- Ví dụ: Lấy lại vd cũ với MaSuKien là 1 cột INT không tự tăng.



hilo

- Thêm 1 table có tên: hibernate_unique_key
- Trong table này tạo 1 column có tên: next_hi
- Column next_hi chỉ có 1 dòng dữ liệu.



hilo – SuKienPOJO.java

```
1  public class SuKienPOJO implements java.io.Serializable {
2      private int maSuKien;
3      private String tenSuKien;
4      //get & set & constructor
5      …
   }
```

hilo – SuKien.hbm.xml

```
1  <hibernate-mapping>
2     <class name="pojo.SuKienPOJO" table="sukien">
3        <id name="maSuKien" column="MaSuKien" type="integer" >
4           <generator class="hilo" />
5        </id>
6        <property name="tenSuKien" column="TenSuKien" type="string"
7  length="100" />
8     </class>
9  </hibernate-mapping>
```

hilo – SuKienDAO.java

```
1   public static List<SuKienPOJO> layDsSuKien() {
2       List<SuKienPOJO> 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.SuKienPOJO").list();
9          trans.commit();
10      } catch (Exception ex) {
11         System.out.println(ex.getMessage());
12      }
13      return list;
14    }
```

hilo – SuKienDAO.java

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

hilo – App

```
1  List<SuKienPOJO> dsSuKien = dao.SuKienDAO.layDsSuKien();
2          for (int i = 0; i < dsSuKien.size(); ++i) {
3             System.out.println (dsSuKien.get(i).getMaSuKien() + "-" +
4  dsSuKien.get(i).getTenSuKien());
5          }
```

MaSuKien	TenSuKien
1	Ngày hội việc làm
2	Ngày hội CNTT
3	Hội chợ triển lãm
5	Dell day
NULL	NULL

```
1-Ngày hội việc làm
2-Ngày hội CNTT
3-Hội chợ triển lãm
5-Dell day
```

hilo – App

```
1  int iThanhCong = 0, iThatBai = 0;
2          for (int i = 0; i < 100; ++i) {
3             SuKienPOJO sk = new SuKienPOJO(i +" day");
4             boolean kq = dao.SuKienDAO.themSuKien(sk);
5             if (kq) iThanhCong++; else iThatBai++;
6          }
7          System.out.println("Thêm thất bại: " + iThatBai);
8          System.out.println("Thêm thành công: " + iThanhCong);
```

```
Thêm thất bại: 0
Thêm thành công: 100
BUILD SUCCESSFUL (total time: 9 seconds)
```

sequence

- Dùng cho các hệ quản trị CSDL: DB2, PostgreSQL, Oracle, SAP DB, …
- Trả về kiểu int, long, hoặc short
- Tự động tăng dần dãy số

native

- Tự lựa chọn 1 trong các kiểu identity, hilo hay sequence phụ thuộc vào khả năng CSDL bên dưới.

uuid

- Sử dụng thuật toán 128bit UUID
- Sinh ra dữ liệu ở dạng chuỗi.

Demo

QUESTIONS?