Friday, March 22, 2019

Lecture 10 - Entities and Association Mapping

Hibernate

- Overview
  - Hibernate的目的是允许您将数据库视为存储Java object。
    - 但是，实际上database不存储object， 它们将data存储在table和column中。
    - 遗憾的是，没有简单的方法关联data（in database）与data（Java object表示）
- Association
  - Oriented-Object association

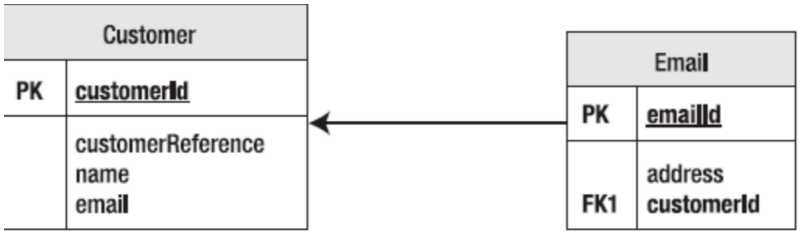
Figure 5-1. *An object-oriented association*

    - 有一个名为User的Class和一个名为Email的Class。User object 包含引用Email Object的field。此关联具有给定User对象的方向，您可以确定associated Email。
    - User user = new User();
    - Email email = user.Email;
    - 如果反过来，用email得到user，就不对
  - Relational Association
    - Association的方向和OO的是完全相反的。在Database中，如果给定一行Email，可以立刻知道在数据库中它属于哪个User。这种关系是被foreign key constrain授权的。
    - 另一个不同的地方是，这种关系在数据库中可以通过使用合适的SQL语句非常方便地反转。

Figure 5-2. *A relational association*

如何手动干预——将java class表现为database table

- Why Mapping Cannot Be Automated
  - 没有引用其他类的时候：允许参数为空的情况吗，如何处理将一个对象存两次
  - 引用其他对象时——4 relationships：一个user可以拥有n个email吗，一个email可以属于n个user吗
- 设计mapping的时候要考虑的问题
  - 一个customer被customer ID确定还是被customer reference确定
  - 一个email address可以被多个customer使用吗
  - relationship应该是Customer table中表达吗
  - relationship应该是Email table中表达吗
  - relationship应该是third table中表达吗
- Example
  - customer是通过customerId确定的，Email只能被一个customer使用，relationship被宰Email table中保持



- primary keys
  - 大部分可以通过SQL访问的relational DB允许访问没有提前定义PK的table
  - 但是Hibernate不允许你这样做，即使你的table已经被created为没有PK的，那么hibernate也会强制你选定PK，会从列集合中生成主键。如果没有PK，就没办法选定table中的特定一行了
- Why does Hibernate need to uniquely identify entries when SQL doesn't?
  - Hibernate表示Java对象，它们始终是唯一可识别的。
  - 区分内容相同的两个String对象的引用和两个对同一String对象的引用。
- Lazy Loading
  - hibernate loads ENTITIES and COLLECTIONS lazily by default
    - 在User 和Email的例子中，只有UserId被实例化。
    - 然而，只要对象以合适的方式associated，在entity和collection被访问的时候，合适的对象将从database被实例化
  - hibernate loads VALUES eagerly
  - 当你想把一个class从database加载到memory的时候，并不希望一次性把所有的信息都加载出来
    - 因为全部加载会浪费很对memory
    - 会花费很多时间来加载信息
  - SQL的解决方法是：SELECT from, to, date, subject FROM email WHERE username = 'someUser';
  - hibernate 的解决方法是lazy load。某些关系可以标记为"lazy"，并且在实际需要之前不会从磁盘加载它们。

Association

- 构建association之前要考虑两个问题
  - Q1:Can 1 Student have more than 1 advisor?
  - Q2:Can 1 Advisor have more than 1 student?

Q1	Q2	Relationship	Hibernate mapping
Y	Y	Many-to-Many (Many students will have many advisors)	<many-to-many>
Y	N	One-to-Many (1 student will have many advisers, but one adviser can not be shared)	<ont-to-many>
N	Y	Many-to-One (Many students will have one advisor)	<many-to-one>
N	N	One-to-One (One student will have one advisor)	<one-to-one>

- <one-to-one>
  - 表示两个class之间的关系，其中第一个class的每个instance与第二个类的单个 instance 相关，反之亦然。这种一对一的关系可以表达为
    - 通过为每个table提供相同的PK，或（通过PK相关联）
    - 通过使用一个table中的FK约束到另一个表的PK（通过FK相关联）
  - 当你不希望额外的table column去关联两个实体时，选择PK association。
  - 主表（master）采用normal PK generator，并且其<one-to-one> mapping entry仅具有指定的 attribute name 和 associated class 。
  - 从属实体（slave entity ）将以类似方式进行mapping，但必须应用约束属性设置以确保关系可以被识别。因为从属类的PK必须与分配给主类（master）的PK相同，所以它被赋予foreign类型的id generator。
  - slave entity：

<id name="id" column="product"> <generator class="foreign">
<param name="property">campaign</param> </generator> <id>
<one-to-one name="campaign“ class="Campaign“ constrained="true"/>

- <many-to-one>
  - 一个类的多个实例可以引用另一个类的单个实例。
  - “many class”有一个FK是“one class”的PK。
  - 两种实现方法
    - 需要两个table，和FK dependency
      - <many-to-one name="email“ class=“Email“ column="email“ cascade=“all" unique="true"/>
    - 使用link table将两个entities组合起来。link table中包含两个表的FK，可以引用与关联中的两个实体关联的两个表。
      - <join table="link_email_user" inverse="true" optional="false">
        <key column="user_id"/>
        <many-to-one name="email" column="email_id" not-null="true"/> </join>

  - 使用link table的缺点是：需要join 3个table，而不是只使用一个table，会降低性能
  - 使用link table的优点是：当关系改变的时候，不需要做很多实质性的改动，只需要改变link table即可
- <many-to-many> relationships required a 3rd table

Unidirectional & Bidirectional

- bi-directional 被建立的时候，一方必须被设置为owner(in a one- to-many or many-to-one association, it must always be the “many” side) ，另一方被设置为 inverse
- 单向关联一般在一方配置多方不进行配置。如：一对多 单向关联在“一”的一方配置文件里进行配置,”多"的一方不进行配置。双向关联两方都要配置

Collection	Association
<set name="titles" table="nameset"> <key column="titleid"/> <element type="string" column="name" not-null="true"/> </set>	<set name="phoneNumbers"> <key column="aduser"/> <one-to-many class="sample.Phone"/> </set>
	<set name="phoneNumbers" inverse="true"> <key column="aduser"/> <one-to-many class="sample.Phone"/> </set>

- **UNIDIRECTIONAL**
  - 单向关联是指只有一方有另一方的关联信息而另一方没有关联信息。我们可以通过A对象中B的关联信息查询或修改B对象的信息，但无法通过B对象来查询修改A对象的信息

```
<class name = “Item” table = “ITEM”>
<list name = “bids”>
        <key column = “ITEM_ID” not-null = “true”/>
        <list-index column = “BID_POSITION”/>
        <one-to-many class = “Bid”/>
</list>
</class>
```

  - 在这个例子中，Item里面包含ArrayList<Bid> bids = new ArrayList<Bid>()
  - ITEM_ID是FK，它必须是非空的，因为一个bid必须被一个item引用。在mapping中要指明这个约束。
    - collection是noninverse的时候，mapping是unidirectional，要假设没有opposite side 映射到相同的FK列，在这种情况下要声明非空的约束

BID

BID_ID	ITEM_ID	BID_POSITION	AMOUNT	CREATED_ON
1	1	0	99.00	19.04.08 23:11
2	1	1	123.00	19.04.08 23:12
3	2	0	433.00	20.04.08 09:30

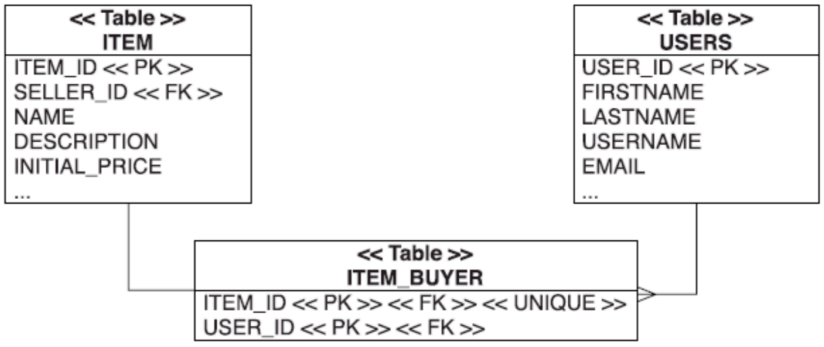
- 需要注意的是，attributes应该在<key>mapping里面，而不是在嵌套的<column>里面。只要你有一个noninverse collection of entity references（大部分情况下是通过<one-to-many>实现的list，set，array）和在target table中通过FK join column 是不可为空的（not nullable），你必须告诉Hibernate。因为Hibernate需要这样的提示来正确排序Insert和Update语句，避免约束冲突（constraint violation）。

- **BIDIRECTIONAL**
  - 双向关联是指两方都有另一方的关联信息。我们可以通过A对象中B的关联信息查询或修改B对象的信息也可以通过B对象来查询修改A对象的信息

```
<class name = "Bid" table = "BID">
    <many-to-one name = "item" column = "ITEM_ID" class = "ITEM"
        not-null = "true" insert = "false" update = "false"/>
</class>
```

  - Bid 对象中有一个item属性，多个bid对应一个item，所以在ITEM_ID 的FK列添加<many-to-one>的关系 & inverse = "true", 使得mapping是双向的（多个bid可以索引到一个item）
  - Hibernate会忽略inverse collection的状态
  - collection包含正确更新数据库的信息：elements的位置
  - 如果只有每个Bid实例的状态被认为是synchronization的，collection是inverse和ignored的，Hibernate在BID_POSITION这一列中是没有值的
  - 如果你mapping一个有index的collection的双向<one-to-many>关系，需要转化inverse side。
  - 你不能让indexed collection的inverse = "true"，这个collection负责state synchronization，一端（Bid），必须inverse。然而，在<many-to-one>中，没有inverse = "true"，所以在<many-to-one>中你必须模拟（simulate）这个属性

- <many-to-many>
- Join table 可以避免空的FK 列



- <set name = "boughtItem" table = "ITEM_BUYER">
  <key column = ""USER_ID/>
  <many-to-many class = "ITEM" column = "ITEM_ID" unique = "true"></set>
- 将set设置为collection table，collection table是一个join table，一般的<one-to-many>不知道join table中的信息