1.JPA是什么?

实际上是规范

类似于JDBC规范

Java Persistence API：用于对象持久化的 API

2. JPA和Hibernate的关系

JPA 是 hibernate 的一个抽象（就像JDBC和JDBC 驱动的关系）：

– JPA 是规范：JPA 本质上就是一种 ORM 规范，不是 ORM 框架 —— 因为 JPA 并未提供 ORM 实现，它只是 制订了一些规范，提供了一些编程的 API 接口，但具体 实现则由 ORM 厂商提供实现

– Hibernate 是实现：Hibernate 除了作为 ORM 框架之外， 它也是一种 JPA 实现

3.为什么用？

1.少量修改，只要都是JPA框架就都可以通用

2. 在JPA 框架下创建实体和创建 Java 类一样简单，只需要使用 javax.persistence.Entity 进行注释；

3. JPA的查询语言是面向对象的，JPA定义了独特的JPQL

4.支持面向对象的高级特性

4.什么构成了JPA？

1.映射关系-ORM 映射元数据

什么意思？

就是数据库中的数据和实体类怎么一一对应？

表中的列名对应哪个属性？

2. JPA 的 API：用来操作实体对象，执行CRUD操作，框架 在后台完成所有的事情，开发者从繁琐的 JDBC和 SQL代 码中解脱出来。

JPA在java中提供的各种方法，只需调用，不需要写复杂的SQL语句，就可以实现CURD

3. 查询语言（JPQL）

JPA自身划分的查询语言

5.JPA的使用流程

1. JPA自身的配置文件+映射关系----xml文件

核心---创建 persistence.xml

• 创建 persistence.xml, 在这个文件中配置持久化单元

– 需要指定跟哪个数据库进行交互;

– 需要指定 JPA 使用哪个持久化的框架以及配置该框架的基本属性

2.Java中的实体类，实体类就是数据

创建实体类, 使用 annotation 来描述实体类跟数据库表之 间的映射关系.

核心---------@ annotation注解

3. 使用 JPA API 完成数据增加、删除、修改和查询操作

调用Java中方法即可

**JPA独特的操纵类为：EntityManager**

– 创建EntityManagerFactory (对应 Hibernate 中的 SessionFactory);

– 创建EntityManager (对应 Hibernate 中的Session);

6.具体使用

1.导入依赖



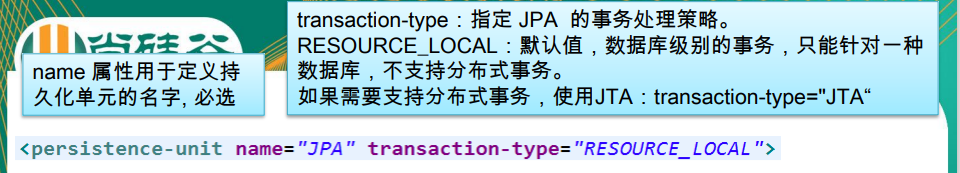
1. persistence.xml-------JPA自身+映射关系配置类

核心

1. 固定位置-JPA 规范要求在类路径的 META-INF 目录下放置 persistence.xml
2. 文件的名称是固定的persistence.xml

配置文件详解：

1. 持久化谁？用什么策略？



2.Java中对应的实体类在哪里？

类的绝对地址



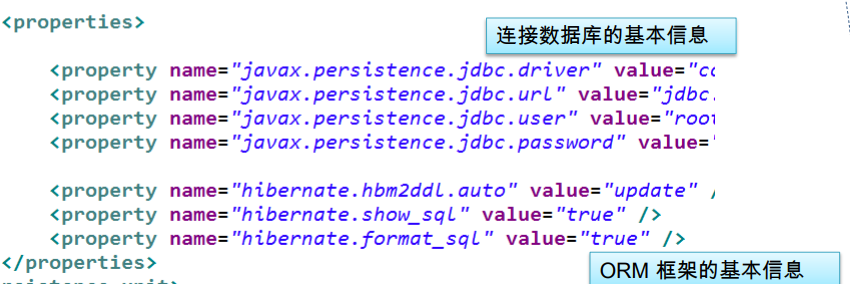
3.属性

数据库的基本信息

驱动 url 用户 密码

框架的设置信息

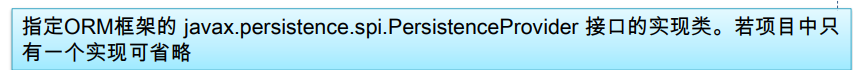
Hibernate的策略 是否显示sql语句 是否格式化语句



1. 指定实现类

提供一个实现类—用来实现此配置文件

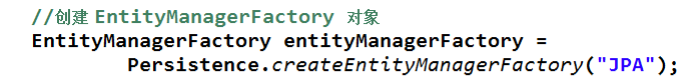




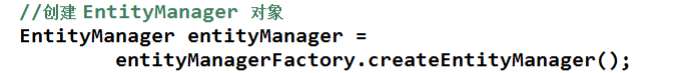
3. 执行持久化操作----数据库交互

1. EntityManagerFactory

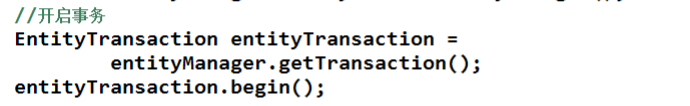
实体类管理工厂---产生实体类



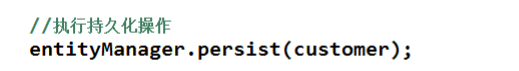
1. EntityManager对象-----JPA的核心类----数据库交互从这



1. 开启事务



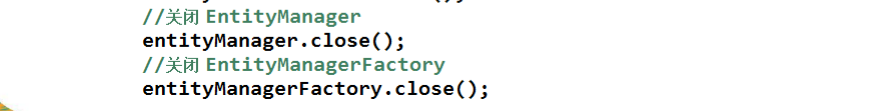
1. 交互数据库



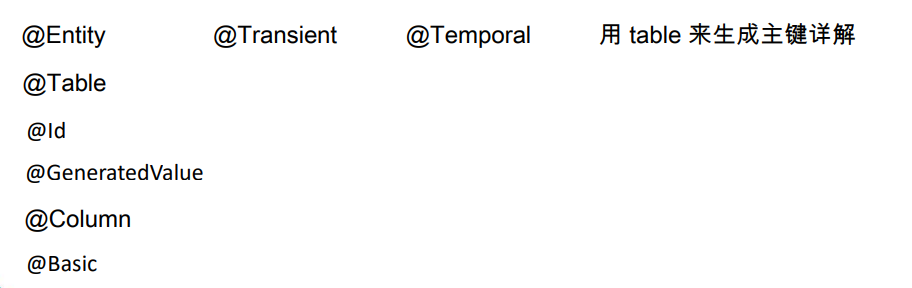
1. 提交事务



1. 关闭操作类+关闭操作类工厂

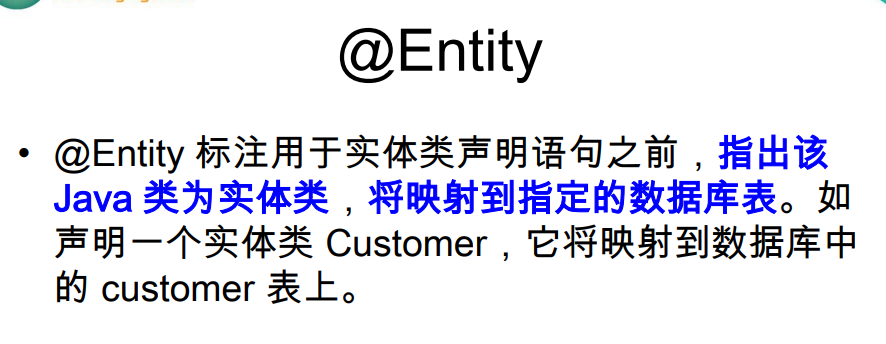


6.JPA 基本注解



用于标识特殊字段或者属性

[1.@Entity](mailto:1.@Entity)



[2.@Table](mailto:2.@Table)

干什么用？

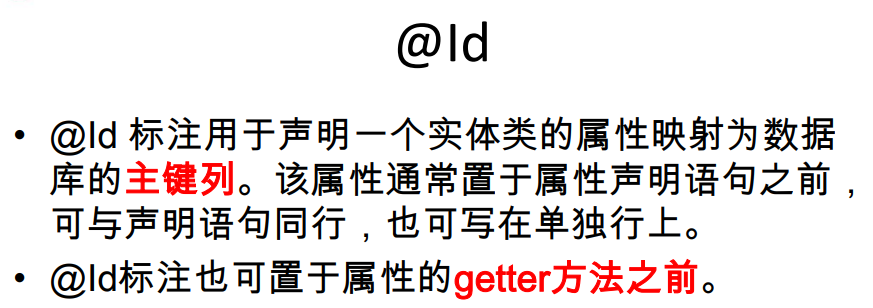
用于将实体类和不同名的表对应

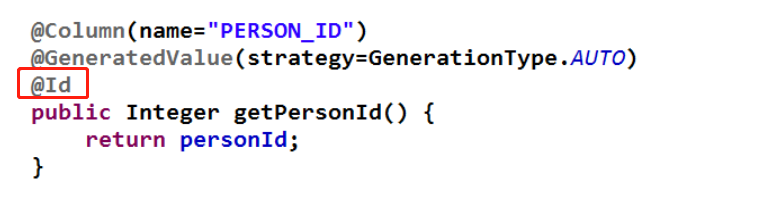


[3.@Id](mailto:3.@Id)

干什么用？

属性用于主键列---这个是主键

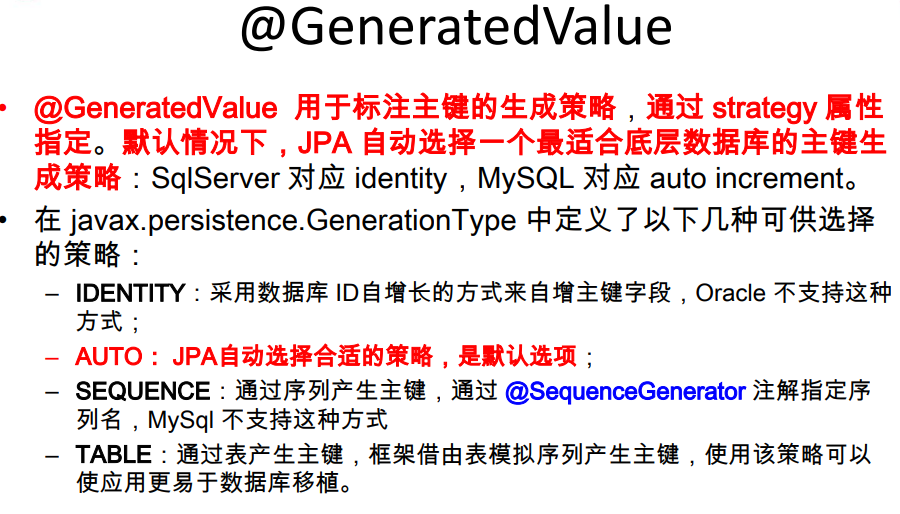




4.@GeneratedValue

干什么用？

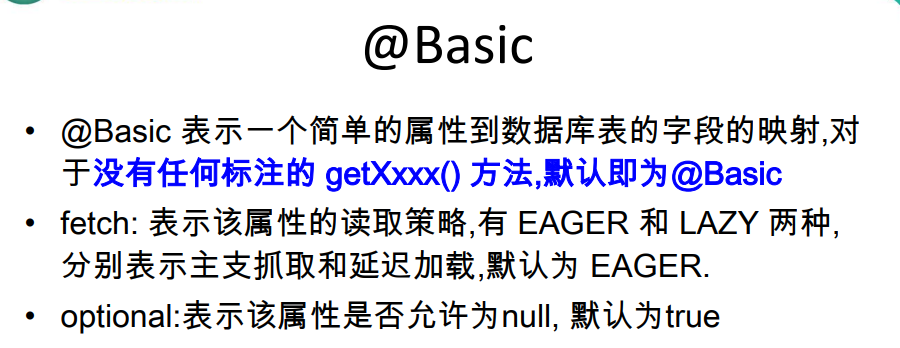
主键自增策略



[5.@Basic](mailto:5.@Basic)

干什么用的？

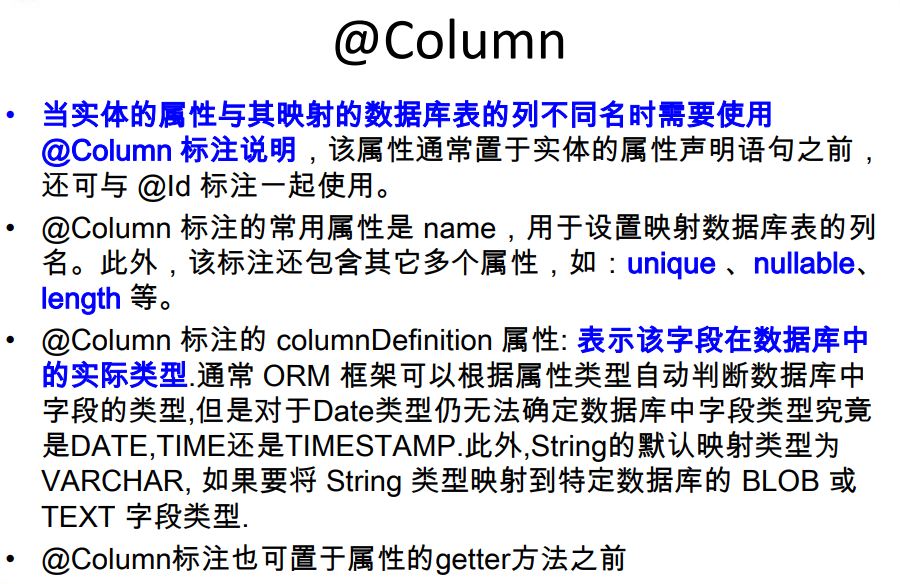
简单属性---没有get此类的方法---为Basic



6.@Column

干什么用的？

Java类的属性与数据库中的表的列名不是对应的----指定和哪个列对应



7. @Transient

干什么用的？

忽略这个属性---表示数据库中没有这个字段

重要，如果存在这样的属性，一定要标注，否则就会被默认未Basic



[8.@Temporal](mailto:8.@Temporal)

干什么用？

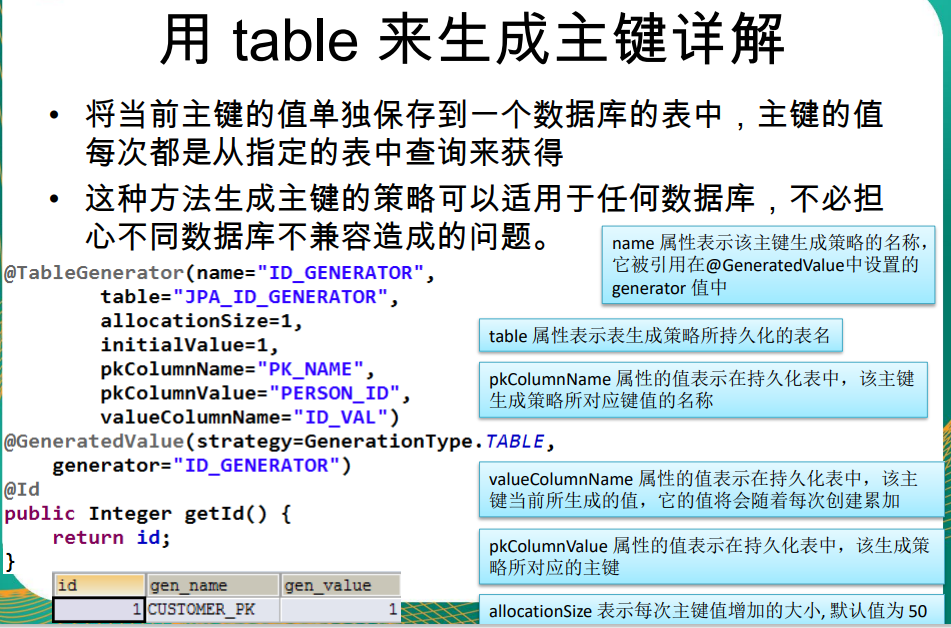
用来标注特殊字段

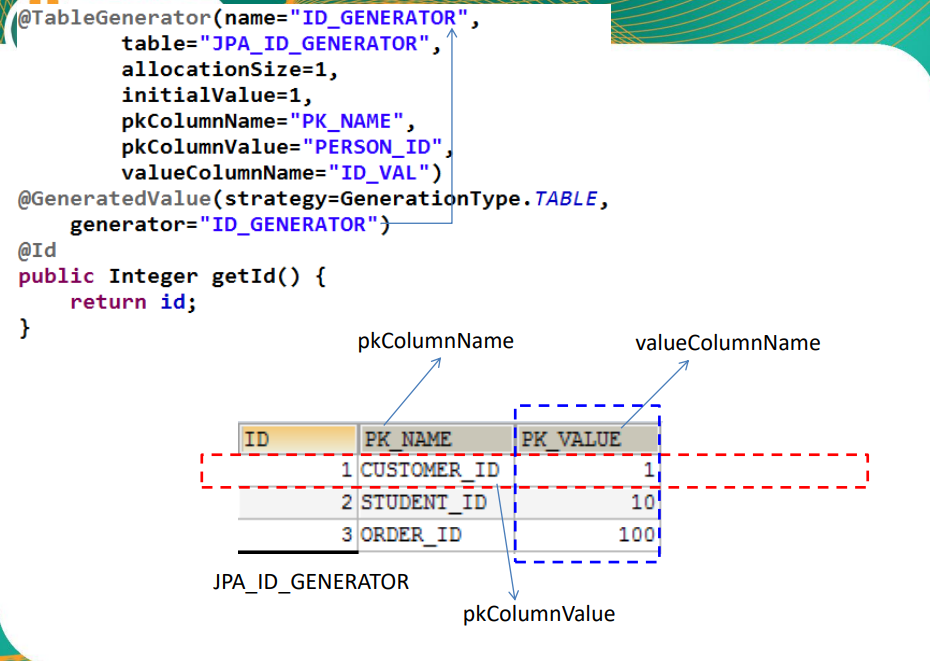


7.用Table来生成主键详解

是什么？

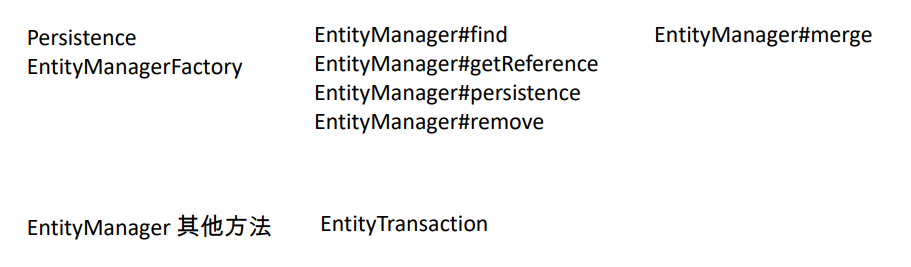
1. 将当前主键的值单独保存到一个数据库的表中，主键的值 每次都是从指定的表中查询来获得
2. 这种方法生成主键的策略可以适用于任何数据库，不必担 心不同数据库不兼容造成的问题。





8.JPA的API

就是JPA提供的方法----增删改查啦



1. Persistence

Persistence 类是用于获取 EntityManagerFactory 实例。

该类包含一个名为 createEntityManagerFactory 的 静态方 法 。

Persistence .createEntityManagerFactory()

createEntityManagerFactory 方法有如下两个重载版本。

– 带有一个参数的方法以 JPA 配置文件 persistence.xml 中的持久化单元名为参数

– 带有两个参数的方法：前一个参数含义相同，后一个参数 Map类型，用于设置 JPA 的相关属性，这时将忽略其它地方设置的属性。

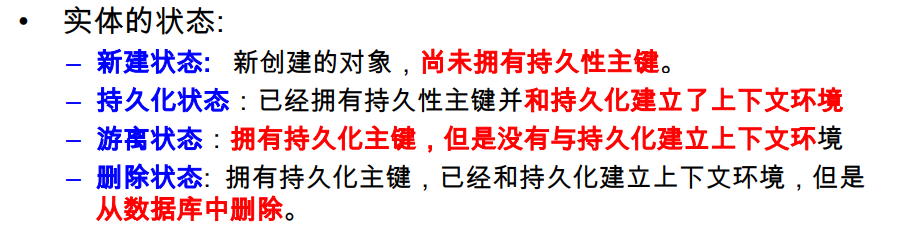
Map 对象的属性名必须是 JPA 实现库提供商的名字空间约定的属性名。

3.EntityManagerFactory



4. EntityManager

核心对象-----------这个用来操纵实体的形式



**具体方法**

• find (Class entityClass,Object primaryKey)：返回指定的 OID 对应的实体类对象，如果这个实体存在于当前的持久化环 境，则返回一个被缓存的对象；否则会创建一个新的 Entity, 并 加载数据库中相关信息；若 OID 不存在于数据库中，则返回一 个 null。第一个参数为被查询的实体类类型，第二个参数为待查 找实体的主键值。

• getReference (Class entityClass,Object primaryKey)：与 find()方法类似，不同的是：如果缓存中不存在指定的 Entity, EntityManager 会创建一个 Entity 类的代理，但是不会立即加载 数据库中的信息，只有第一次真正使用此 Entity 的属性才加载， 所以如果此 OID 在数据库不存在，getReference() 不会返回 null 值, 而是抛出EntityNotFoundException

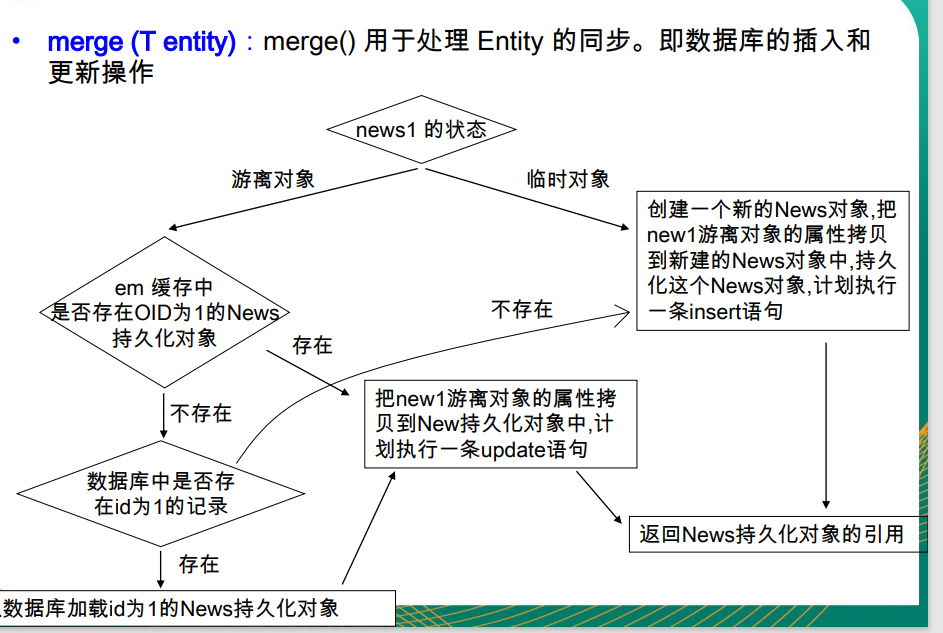
• persist (Object entity)：用于将新创建的 Entity 纳入到 EntityManager 的管理。该方法执行后，传入 persist() 方法 的 Entity 对象转换成持久化状态。

– 如果传入 persist() 方法的 Entity 对象已经处于持久化状态，则 persist() 方法什么都不做。

– 如果对删除状态的 Entity 进行 persist() 操作，会转换为持久化状态。

– 如果对游离状态的实体执行 persist() 操作，可能会在 persist() 方法 抛出 EntityExistException(也有可能是在flush或事务提交后抛出)

• remove (Object entity)：删除实例。如果实例是被管理的，即 与数据库实体记录关联，则同时会删除关联的数据库记录。



• flush ()：同步持久上下文环境，即将持久上下文环境的所有未 保存实体的状态信息保存到数据库中。

• setFlushMode (FlushModeType flushMode)：设置持久上下文 环境的Flush模式。参数可以取2个枚举 – FlushModeType.AUTO 为自动更新数据库实体， – FlushModeType.COMMIT 为直到提交事务时才更新数据库记录。

• getFlushMode ()：获取持久上下文环境的Flush模式。返回 FlushModeType类的枚举值。

• refresh (Object entity)：用数据库实体记录的值更新实体对象的状态，即更新 实例的属性值。

• clear ()：清除持久上下文环境，断开所有关联的实体。如果这时还有未提交 的更新则会被撤消。

• contains (Object entity)：判断一个实例是否属于当前持久上下文环境管理的 实体。

• isOpen ()：判断当前的实体管理器是否是打开状态。

• getTransaction ()：返回资源层的事务对象。EntityTransaction实例可以用于 开始和提交多个事务。

• close ()：关闭实体管理器。之后若调用实体管理器实例的方法或其派生的查 询对象的方法都将抛出 IllegalstateException 异常，除了getTransaction 和 isOpen方法(返回 false)。不过，当与实体管理器关联的事务处于活动状态时， 调用 close 方法后持久上下文将仍处于被管理状态，直到事务完成。

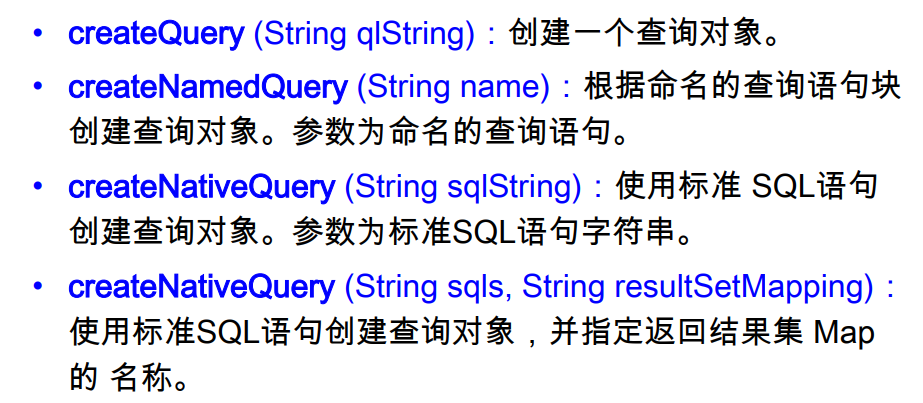
**增删改查方面**

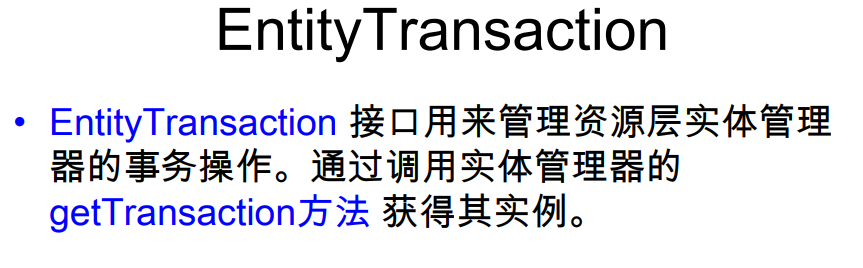
• createQuery (String qlString)：创建一个查询对象。

• createNamedQuery (String name)：根据命名的查询语句块 创建查询对象。参数为命名的查询语句。

• createNativeQuery (String sqlString)：使用标准 SQL语句 创建查询对象。参数为标准SQL语句字符串。

• createNativeQuery (String sqls, String resultSetMapping)： 使用标准SQL语句创建查询对象，并指定返回结果集 Map 的 名称。







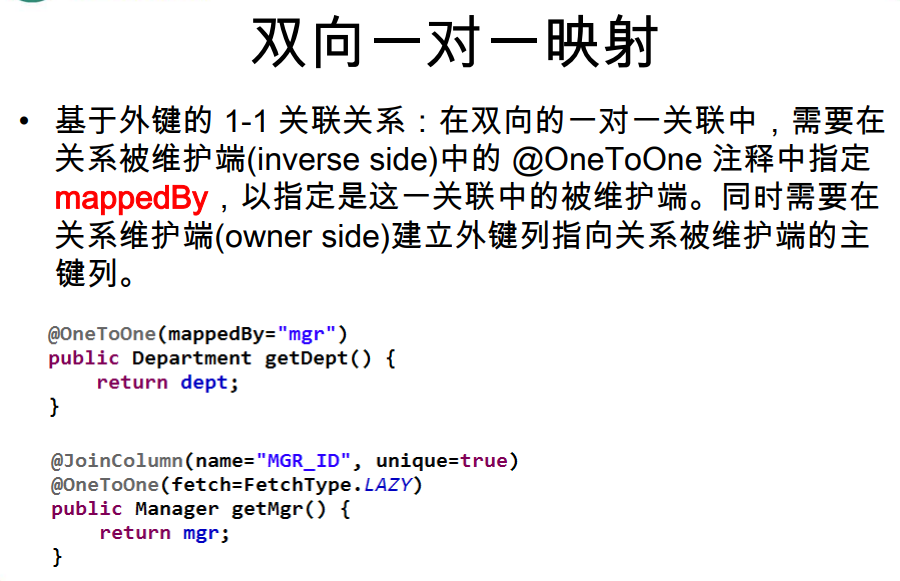
9.映射关联关系

一对一 一对多 多对一 多对多

1. **双向一对多及多对一映射（重要）**



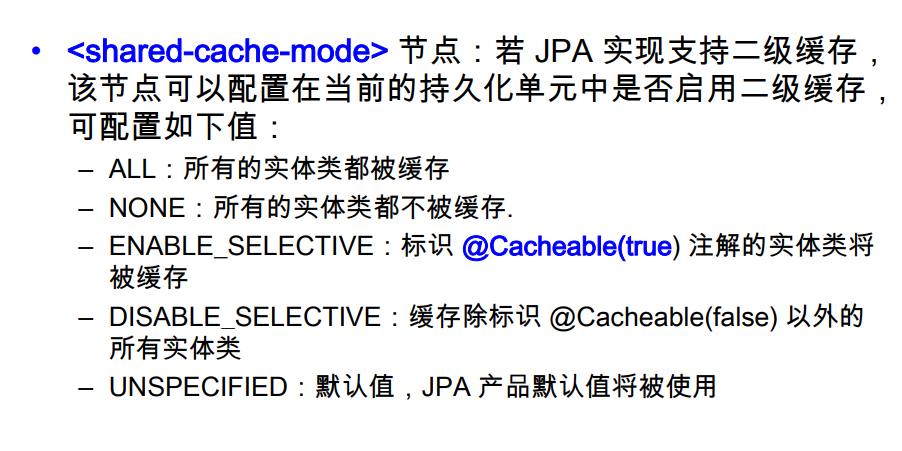
1. **双向一对一映射**



1. **双向多对多关联关系**



10.二级缓存-----与Hirbelt类似



11. JPQL



12. JPQL语言

JPQL是什么？

SQL啦！！！

特征：JPQL语言的语句可以是 select 语句、update 语句或 delete语句，它们都通过 Query 接口封装执行

重点：只有Select Update 和 Delete ------全部通过Query执行

13. javax.persistence.Query

是什么？

接口！

干什么用的？

封装了执行数据库查询的相关方法。

怎么用？

调EntityManager 的 createQuery、create NamedQuery 及 createNativeQuery 方法可以获得查询对象，进而可调用 Query 接口的相关方法来执行查询操作。

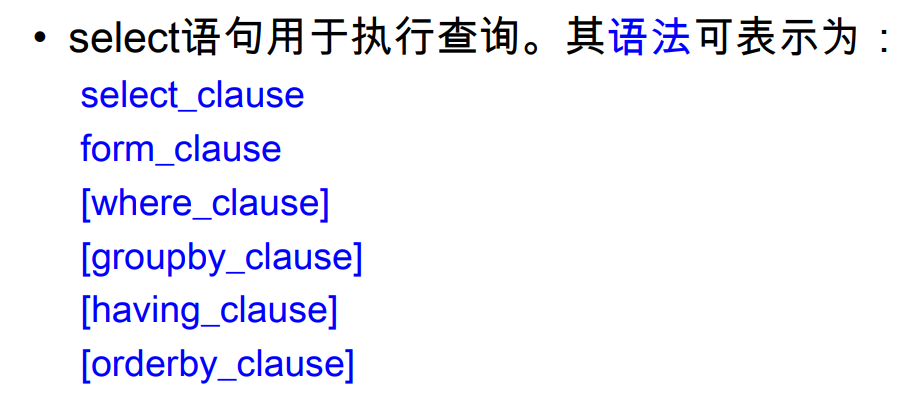
14. Query接口的主要方法



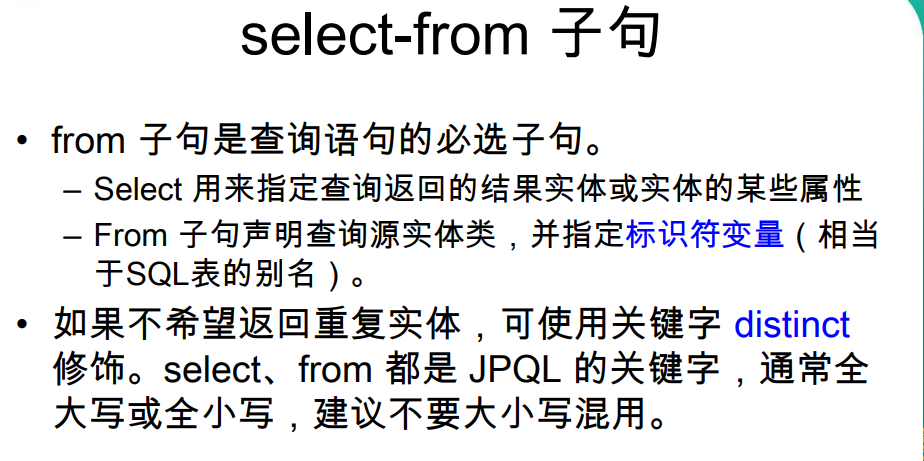


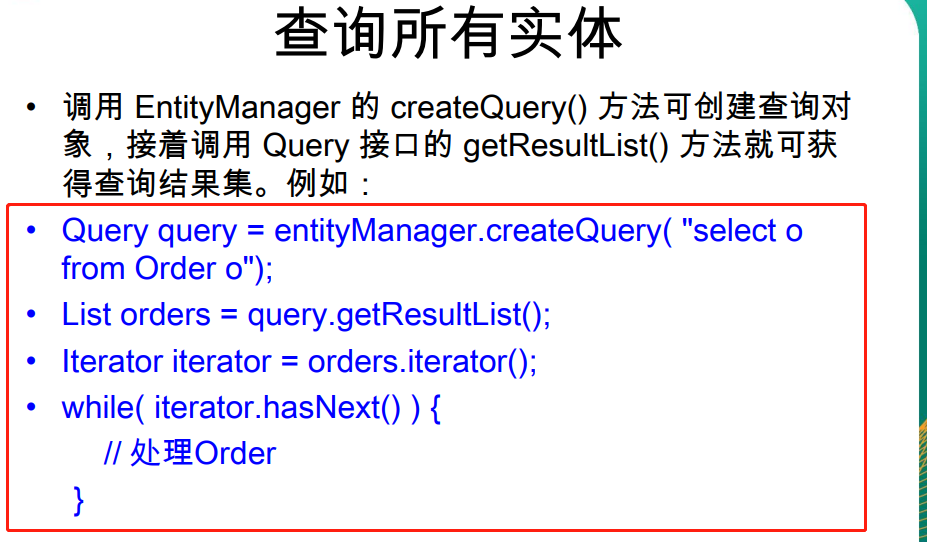
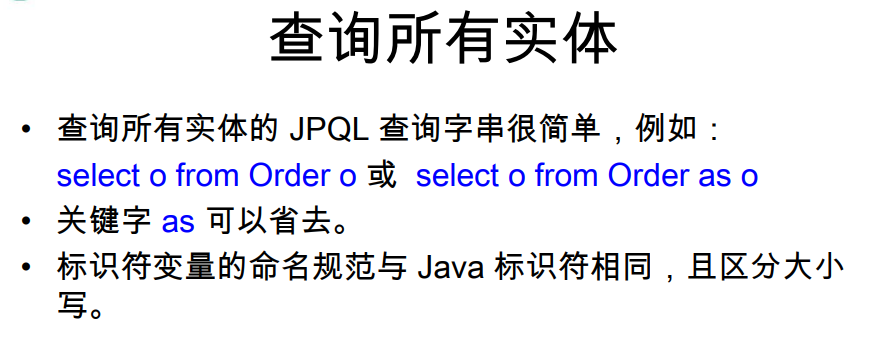
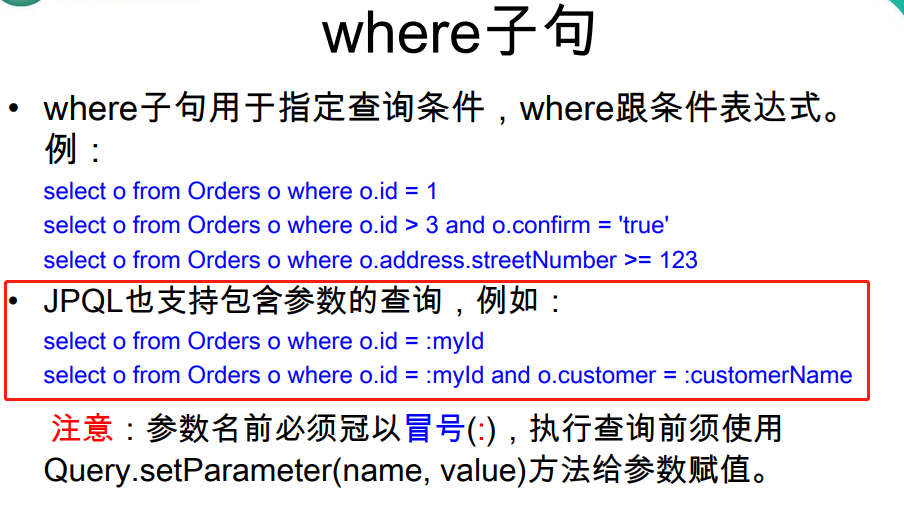


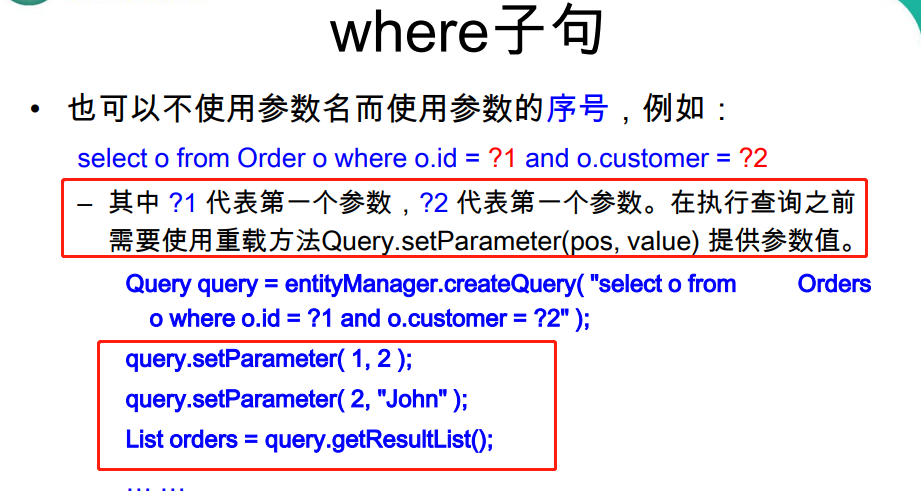
15.JPA的Select语句



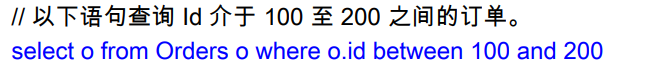
类似SQL 回忆

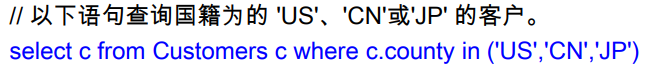


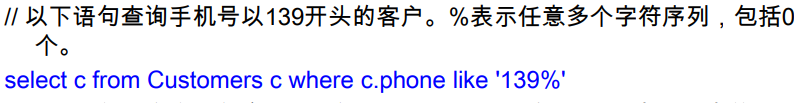
   


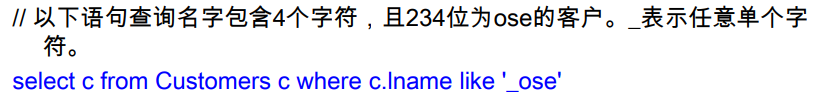


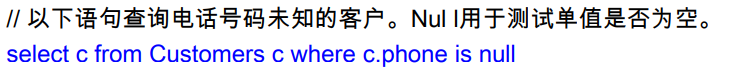
16.Where子句示例-----核心

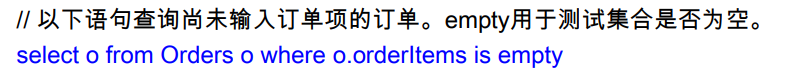






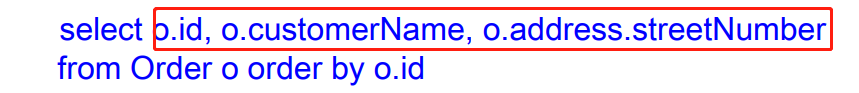


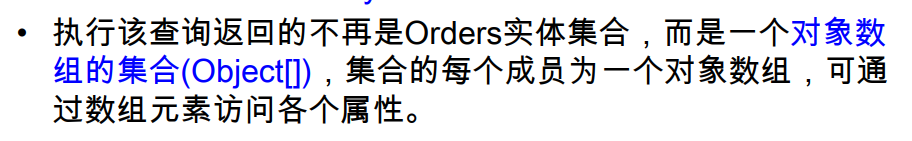




17. 查询部分属性

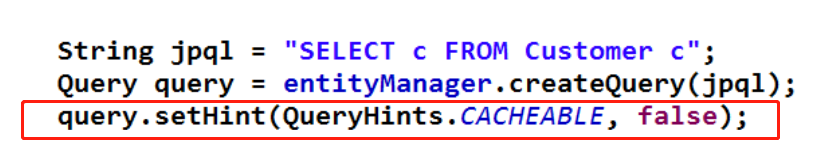
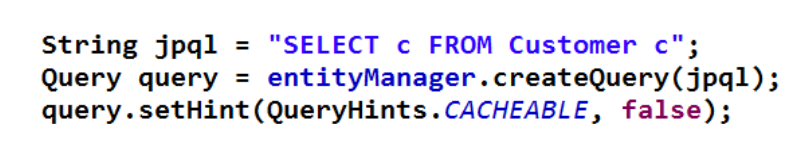
如果只须查询实体的部分属性而不需要返回整个实体。





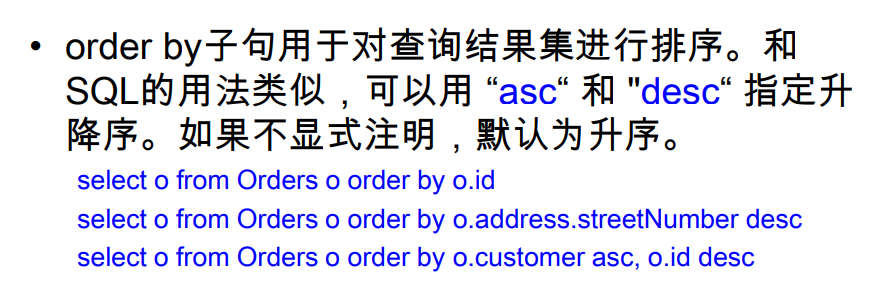
18. 使用 Hibernate 的查询缓存

核心是这里

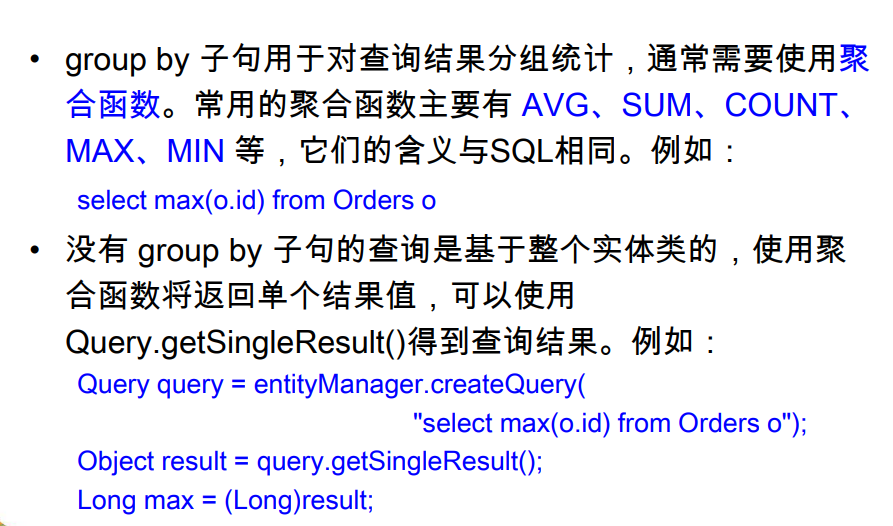
19.Order by子句

order by子句用于对查询结果集进行排序--------排序啦！



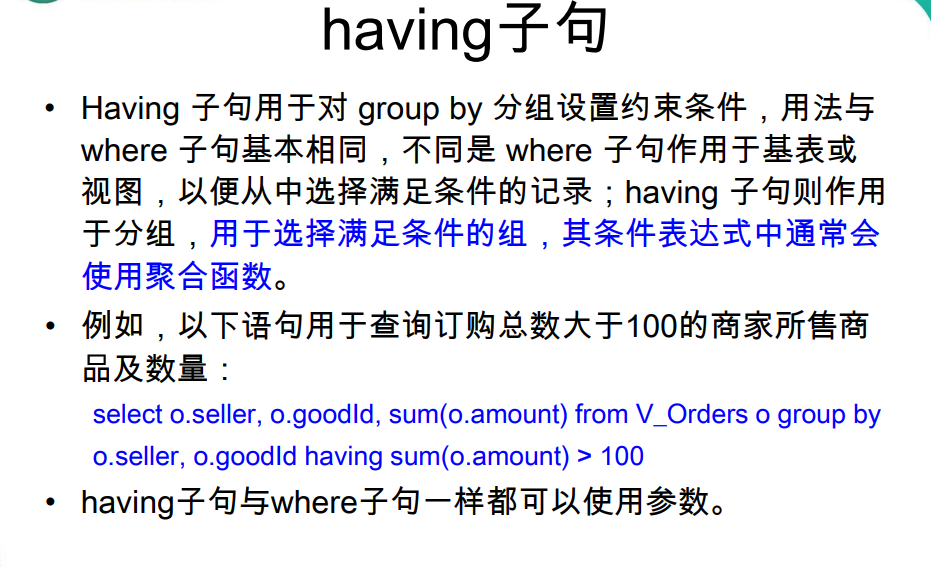
20. group by子句与聚合查询

**与SQL中的相同啦**

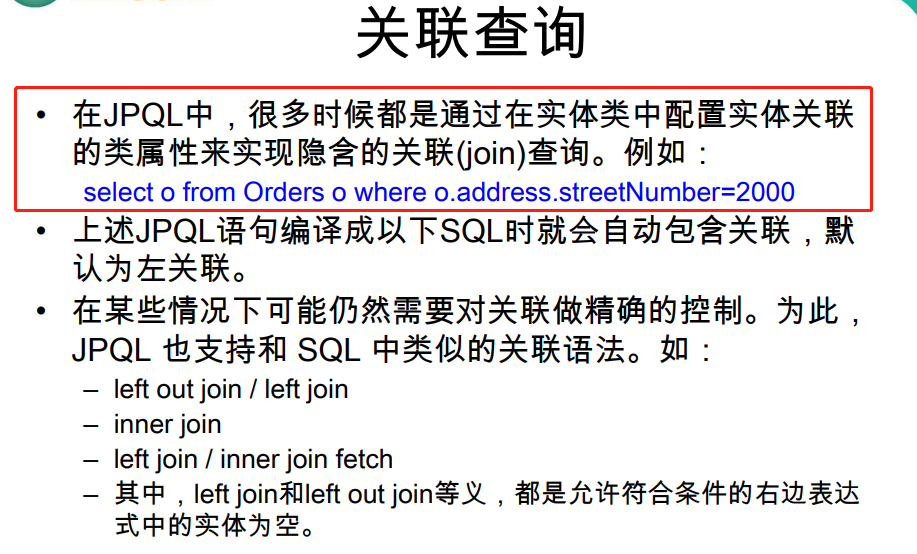


21. having子句

约束条件----------与Where基本相同啦



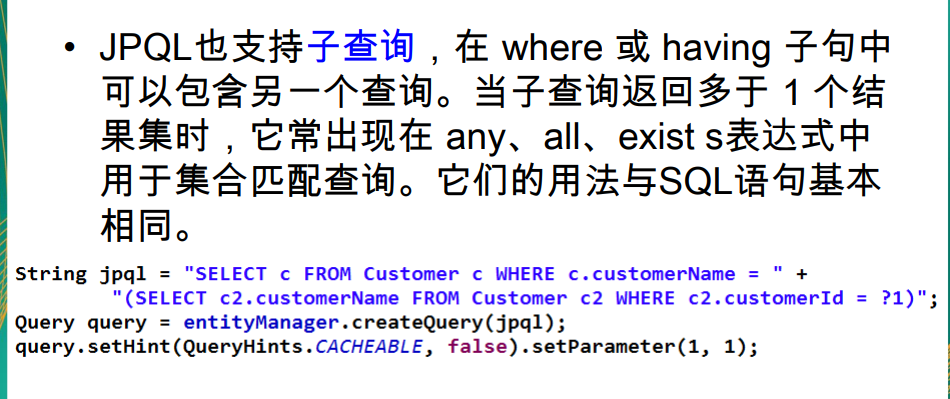
22.关联查询



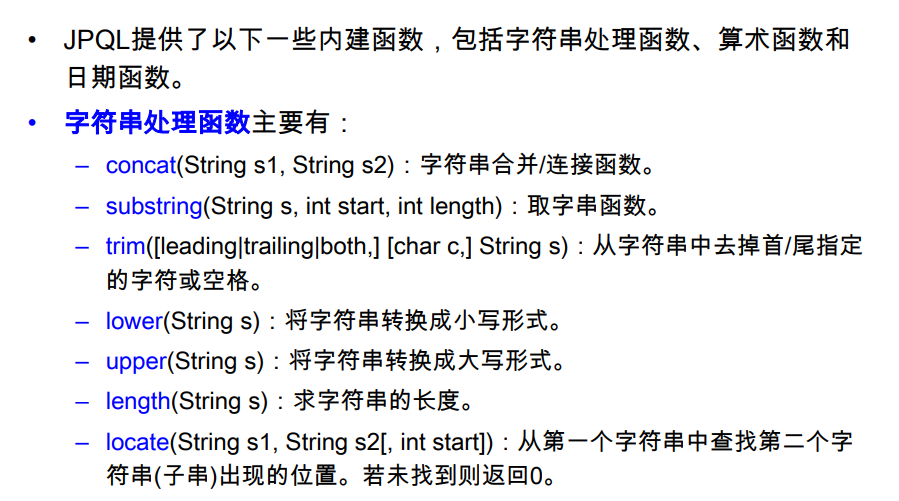
**关联查询示例：**

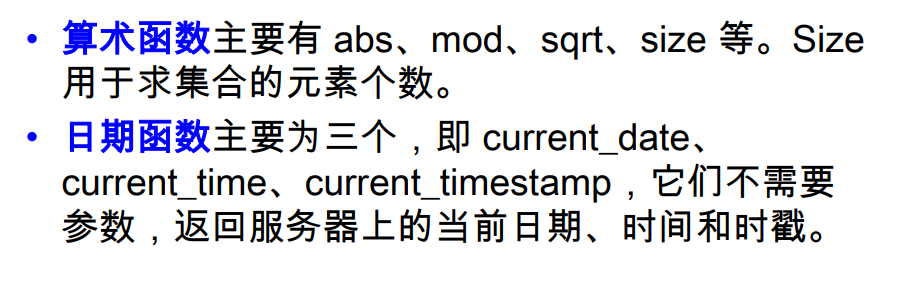


23.子查询



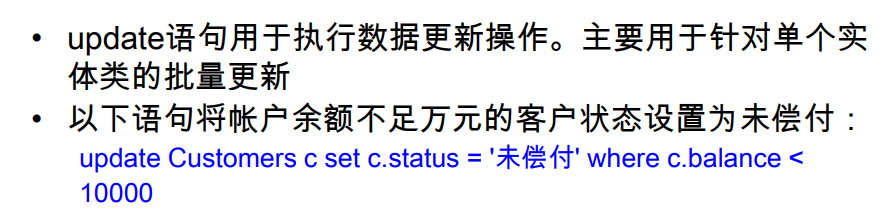
24.JPQL函数



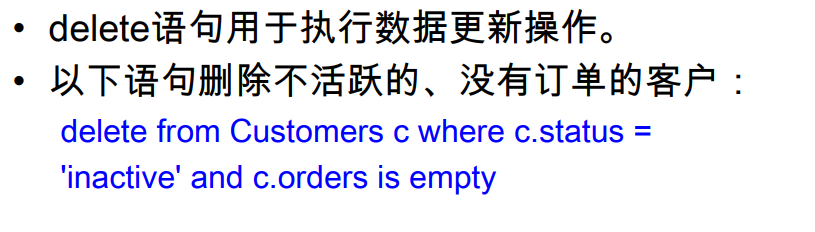


25.增删改查语句

1. update语句



2.delete语句



26.核心----整合Spring---具体使用！



**具体使用：**

