x 数据访问层技术应用

历年真题

在信息系统的开发与建设中，分层设计是一种常见的架构设计方法，区分层次的目的是为了实现“高内聚低耦合”的思想。分层设计能有效简化系统复杂性，使设计结构清晰，便于提高复用能力和产品维护能力。一种常见的层次划分模型是将信息系统分为表现层、业务逻辑层和数据访问层。信息系统一般以数据为中心，数据访问层的设计是系统设计中的重要内容。数据访问层需要针对需求，提供对数据源读写的访问接口；在保障性能的前提下，数据访问层应具有良好的封装性、可移植性，以及数据库无关性。

请围绕“论数据访问层设计技术及其应用”论题，依次从以下三个方面进行论述。

1．概要叙述你参与管理和开发的与数据访问层设计有关的软件项目，以及你在其中所担任的主要工作。

2．详细论述常见的数据访问层设计技术及其所包含的主要内容。

3．结合你参与管理和开发的实际项目，具体说明采用了哪种数据访问层设计技术，并叙述具体实施过程以及应用效果。

一、首先用 400-600 字的篇幅简要叙述作者参与开发的软件系统的概要和所担任的工作。

二、数据访问层的技术主要在于数据映射的问题如写 Hibernate 或 iBATIS 的应用。

相对 Hibernate“O/R”而言，iBATIS 是一种“Sql Mapping”的 ORM 实现。

Hibernate 是一个开放源代码的**对象关系映射框架**，它对 JDBC 进行了非常轻量级的对象封装，它将 POJO 与数据库表建立映射关系，是一个全自动的 orm 框架，hibernate 可以自动生成 SQL 语句，自动执行，使得 Java 程序员可以随心所欲的**使用对象编程思维来操纵数据库**。

Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 **Java 的客户端**程序使用，也可以在 Servlet/JSP 的 **Web 应用**中使用，最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 **CMP**，完成数据持久化的重任。

Hibernate 的调优方案：

制定合理的缓存策略；

尽量使用延迟加载特性；

采用合理的 Session 管理机制；

使用批量抓取，设定合理的批处理参数（batch_size）；

进行合理的 O/R 映射设计。

写作要点

摘要

正文

Hibernate

概念：Hibernate 是一个开放源代码的**对象关系映射框架**，它对 JDBC 进行了非常轻量

级的对象封装，它将 POJO 与数据库表建立映射关系，是一个全自动的 orm 框架，hibernate 可以自动生成 SQL 语句，自动执行，使得 Java 程序员可以随心所欲的**使用对象编程思维来操纵数据库**。

适用场合：Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用，最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。

JDBC:

ORM 对象/关系映射，表中的每行对应于类的一个实例，而每列的值对应于该实例的一个属性.

缺点：不同类型代码混淆，可读性差,维护难/ SQL 不支持面向对象思维/ 错误运行时才发现，调试难

HQL

HQL 语言以面向对象的操作方式替代了关系语言(SQL).

HQL 语言是对持久化类进行操作的语言，将持久化类的对象看作 SQL 语言中操作的表名，将对象的属性看作是 SQL 语言中操作的字段.

例：灵活构造对象，进行分组、排序等基本 SQL 语句的功能.

1. session 管理机制

SessionFactory 中保存了对象当前数据库配置的所有映射关系，同时负责维护当前的二级数据库缓存和 statement pool，由于 SessionFactory 创建过程中系统的开销非常大，因此在一个应用中针对一个数据库设计一个 SessionFactory 实例. sessionFactory 是线程安全的，多个并发线程可以同时访问一个 SessionFactory 并从中获取 Session 实例.

Session 是 hibernate 持久化操作的基础，提供了 Hibernate 的众多持久化方法. Hibernate 通过 session 来完成对对数据库的操作. 由于 Session 并非线程安全，如果多个线程同时使用一个 Session 实例进行数据存取，则将会导致 Session 数据存取逻辑混乱。因此我们在设计过程中，严格保证一个 session 只可由一个线程使用.

例：本交通信息包含 oracle 和 postSQL 两个数据库，因此分别为这两个数据库单独创建一个 SessionFactory. 爬取高德路况 API 时，建立多个线程，每个线程使用一个独立的 session.

2. 缓存管理

一级缓存：Session 缓存，它是属于事务范围的缓存，这一级别的缓存由 Hibernate 管理的，一般情况下无需干预。显式执行 flush 之前，所有的持久层操作的数据都缓存在 session 对象处，位于缓存中的对象称为持久化对象，它和数据库中的相关记录对应。Session 缓存可减少 Hibernate 应用程序访问数据库的频率。clear()将会清理掉 session 的缓存。

二级缓存: sessionFactory 缓存，它是属于进程范围或群集范围的缓存，这一级别的缓存可以进行配置和更改，并且可以动态加载和卸载。当 Hibernate 根据 ID 访问数据对象的时候，首先从 Session 一级缓存中查；查不到，如果配置了二级缓存，那么从二级缓存中查；查不到，再查询数据库，把结果按照 ID 放入到缓存。对于经常使用的查询语句，如果启用了查询缓存，当第一次执行查询语句时，Hibernate 会把查询结果存放在第二缓存中。以后再次执行该查询语句时，只需从缓存中获得查询结果，从而提高查询性能。适用于很少被修改的数据.

例：高德 API 抓取的路况数量达到数百万，因此当使用完对象后，即时使用 clear 清空缓 session 缓存. 由于人口、用地等数据比较固定，因此启用二级缓存功能，将这些不变的数据放入 sessionFacotry 缓存，从而提高查询性能.

3. 延迟加载

hibernate 支持延迟加载，也称为懒加载，在 hibernate 设置延迟加载后，hibernate 返回给我们的对象（要延迟加载的对象）是一个代理对象，并不是真实的对象，该对象没有真实对象的数据，只有真正需要用到对象数据（调用 getter 等方法时）时，才会触发 hibernate 去数据库查对应数据。当调用 Session 上的 load()方法加载一个实体时，会采用延迟加载。

例: 在本平台中, 定义了一些双向关系的类, (many to one) , 如一条道路名称对应多条短的路段, 这些路段具有相同的道路名称, 但有着不同的车道数量、车道宽度等道路属性, 如未使用懒加载, 访问某道路对象时, 会将所有对应的路段记录的数据也一并查询后返回, 而后期只需要用到某些路段数据, 这就造成了资源的浪费. 因此使用懒加载, 当访问道路对象时不进行 SQL 查询, 只有在使用对象数据时才访问数据库进行查询.

4. 批处理: fetch_size/batch_size

当查询的记录很多时，系统不会一次性取出所有的数据，而只会去取 fetch size 条数，当遍历完这些记录后，再取同样的条数. 这样可以大大节省了无谓的内存消耗. 当 Fetch Size 设的越大，读数据库的次数越少，速度越快，也不是越大越好，经过测试，取 50 时能取得较好的查询性能. (mysql 不支持)

当对数据库进行批量插入、更新、删除时, 调整 batch_size 以控制向数据库发送 SQL 的次数. 向数据库发送 sql 的次数越少，速度就越快. 经测试, 取 45 时性能较好.

5. O/R 映射设计

主键生成策略/ 多对一/