Java数据库程序中的存储过程设计 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/475/2021_2022_Java_E6_95_B0_E6_8D_AE_c67_475823.htm 本文阐述了怎么使用DBMS存储过程。阐述了使用存储过程的基本的和高级特性，比如返回ResultSet.本文假设你对DBMS和JDBC已经非常熟悉，也假设你能够毫无障碍地阅读其它语言写成的代码（即不是Java的语言），但是，并不要求你有任何存储过程的编程经历。 存储过程是指保存在数据库并在数据库端执行的程序。你可以使用特殊的语法在Java类中调用存储过程。在调用时，存储过程的名称及指定的参数通过JDBC连接发送给DBMS，执行存储过程并通过连接（如果有）返回结果。 使用存储过程拥有和使用基于EJB或CORBA这样的应用服务器一样的好处。区别是存储过程可以从很多流行的DBMS中免费使用，而应用服务器大都非常昂贵。这并不只是许可证费用的问题。使用应用服务器所需要花费的管理、编写代码的费用，以及客户程序所增加的复杂性，都可以通过DBMS中的存储过程所整个地替代。 你可以使用Java，Python，Perl或C编写存储过程，但是通常使用你的DBMS所指定的特定语言。Oracle使用PL/SQL，PostgreSQL使用pl/pgsql，DB2使用Procedural SQL.这些语言都非常相似。在它们之间移植存储过程并不比在Sun的EJB规范不同实现版本之间移植Session Bean困难。并且，存储过程是为嵌入SQL所设计，这使得它们比Java或C等语言更加友好地方式表达数据库的机制。 因为存储过程运行在DBMS自身，这可以帮助减少应用程序中的等待时间。不是在Java代码中执行4个或5个SQL语句，而只需要在服务器端

执行1个存储过程。网络上的数据往返次数的减少可以戏剧性地优化性能。 使用存储过程 简单的老的JDBC通过CallableStatement类支持存储过程的调用。该类实际上是PreparedStatement的一个子类。假设我们有一个poets数据库。数据库中有一个设置诗人逝世年龄的存储过程。下面是对老酒鬼Dylan Thomas（old soak Dylan Thomas，不指定是否有关典故、文化，请批评指正。译注）进行调用的详细代码：try{ int age = 39. String poetName = "dylan thomas". CallableStatement proc = connection.prepareCall(" { call set_death_age(?, ?) }"). proc.setString(1, poetName). proc.setInt(2, age). cs.execute().}catch (SQLException e){ // ....} 传给prepareCall方法的字串是存储过程调用的书写规范。它指定了存储过程的名称，？代表了你需要指定的参数。 和JDBC集成是存储过程的一个很大的便利：为了从应用中调用存储过程，不需要存根（stub）类或者配置文件，除了你的DBMS的JDBC驱动程序外什么也不需要。 当这段代码执行时，数据库的存储过程就被调用。我们没有去获取结果，因为该存储过程并不返回结果。执行成功或失败将通过例外得知。失败可能意味着调用存储过程时的失败（比如提供的一个参数的类型不正确），或者一个应用程序的失败（比如抛出一个例外指示在poets数据库中并不存在“ Dylan Thomas ”） 结合SQL操作与存储过程 映射Java对象到SQL表中的行相当简单，但是通常需要执行几个SQL语句；可能是一个SELECT查找ID，然后一个INSERT插入指定ID的数据。在高度规格化（符合更高的范式，译注）的数据库模式中，可能需要多个表的更新，因此需要更多的语句。Java代码会很快地膨胀，每一个语句的网络

开销也迅速增加。 将这些SQL语句转移到一个存储过程中将大大简化代码，仅涉及一次网络调用。所有关联的SQL操作都可以在数据库内部发生。并且，存储过程语言，例如PL/SQL，允许使用SQL语法，这比Java代码更加自然。下面是我们早期的存储过程，使用Oracle的PL/SQL语言编写：create procedure set_death_age(poet VARCHAR2, poet_age NUMBER)poet_id NUMBER.beginSELECT id INTO poet_id FROM poets WHERE name = poet.INSERT INTO deaths (mort_id, age) VALUES (poet_id, poet_age).end set_death_age. 很独特？不。我打赌你一定期待看到一个poets表上的UPDATE. 这也暗示了使用存储过程实现是多么容易的一件事情。set_death_age几乎可以肯定是一个很烂的实现。我们应该在poets表中添加一列来存储逝世年龄。Java代码中并不关心数据库模式是怎么实现的，因为它仅调用存储过程。我们以后可以改变数据库模式以提高性能，但是我们不必修改我们代码。 下面是调用上面存储过程的Java代码：public static void setDeathAge(Poet dyingBard, int age)throws SQLException{ Connection con = null. CallableStatement proc = null. try { con = connectionPool.getConnection(). proc = con.prepareCall("{ call set_death_age(?, ?) }"). proc.setString(1, dyingBard.getName()). proc.setInt(2, age). proc.execute(). } finally { try { proc.close(). } catch (SQLException e) {} con.close(). }} 为了确保可维护性，建议使用像这儿这样的static方法。这也使得调用存储过程的代码集中在一个简单的模版代码中。如果你用到许多存储过程，就会发现仅需要拷贝、粘贴就可以创建新的方法。因为代码的模版化，甚至也可以通过脚本自动生产调用存储过程的

代码。 Functions 存储过程可以有返回值，所以CallableStatement类有类似getResultSet这样的方法来获取返回值。当存储过程返回一个值时，你必须使用registerOutParameter方法告诉JDBC驱动器该值的SQL类型是什么。你也必须调整存储过程调用来指示该过程返回一个值。 下面接着上面的例子。这次我们查询Dylan Thomas逝世时的年龄。这次的存储过程使用PostgreSQL的pl/pgsql：create function snuffed_it_when (VARCHAR) returns integer declarepoet_id NUMBER.poet_age NUMBER.begin-- first get the id associated with the poet.SELECT id INTO poet_id FROM poets WHERE name = $1.-- get and return the age.SELECT age INTO poet_age FROM deaths WHERE mort_id = poet_id.return age.end. language pl/pgsql. 另外，注意pl/pgsql参数名通过Unix和DOS脚本的$n语法引用。同时，也注意嵌入的注释，这是和Java代码相比的另一个优越性。在Java中写这样的注释当然是可以的，但是看起来很凌乱，并且和SQL语句脱节，必须嵌入到Java String中。