Universe Scientific Publishing
Computer System Networking and Telecommunications
计算机系统网络和电信，2019,1(2)
ISSN:2661-3719 (Print); 2661-3727 (Online)

Analysis of Database Access Technology used in Java

Junhui LI Zhengde BAO Yawen TANG
School of Computer and Software, Jincheng College, Sichuan University, Chengdu, Sichuan 611731

Abstract

Java is widely used in Android, server, program, website and so on. At present, the main application of Java is JDBC database access mechanism. Based on the Java language environment, this paper analyzes the method of accessing database in Java, the connection technology, and so on. By comparing different connection techniques to understand the differences between different methods, this paper speculates how to improve the efficiency of database access in Java language.

Key Words

Java Language, Database Access, Connection Technology, Access

DOI:10.18686/jsjxt.v1i2.665

浅析在 Java 中使用的数据库访问技术

李俊辉 鲍正德 唐娅雯
四川大学锦城学院计算机与软件学院，四川成都，611731

摘 要

Java 在 Android、服务器、程序、网站等方面都有广泛的应用。目前在 Java 中主要应用的是 JDBC 数据库访问机制，本文在 Java 语言环境的根基之下，浅析了在 Java 中数据库的访问的办法，连接技术，并通过对比不同的连接技术了解不同方法之间的差异，推测如何提升在 Java 语言中数据库的访问效率。

关键词

Java 语言；数据库访问；连接技术；访问效率

1.引言

Java 语言在现有软件行业中应用的十分广泛，因为其具有简单性，健壮性等特点，因而，在大型的软件开发中应用广泛，每一个软件系统都不可避免的会和数据打交道目前主要使用数据库技术来处理数据，数据库技术是一项基本的软件开发技术，特别是，现今时代，大数据和人工智能产业的发展得到极大的提升，数据库技术应用的更加广泛。伸缩性以及健壮性是现今数据库访问技术所拥有的特性，保证了其使用的广泛性。在 Java 中通过与数据库进行连接可以更加高效和安全的处理数据，需要不断提高 Java 开发中数据库的访问效率，本文简略的论述在 Java 环境下，使用数据库的技术，分析各个方法的不同之处。

2. Java 语言中常用数据库访问技术

在 Java 语言环境下，不同的数据库访问方式有着不同的特点和其适用的场景，从不同的方面有着千丝万缕的联系或是巨大的差别，在了解基本的数据库访问方法之下，才能够展开对比和分析。在 Java 中运用相比宽泛的数据库访问技术主要有四种，这四种访问方式各有特点，其中一种是基础，剩下三种是在基础之上的延伸和扩展，本文会就这四种访问技术进行大体的阐述。

2.1 JDBC 访问技术

JDBC 数据库访问技术，别名也称为 Java 数据库连接技术，在 Java 中 JDBC 是一种标准的 API，即应用设计接口，这种接口能够用来直接的执行 SQL 语句，而不需要进行语言的转换。当然，JDBC 中的这一组类和接口是由 Java 语言编写的，主要是为 Java 的开发提供便利，它可以提供统一的访问服务，针对多个相互关

联的数据库。在 JDBC 出现之前，只有 DBMS 就是数据库管理系统，这种系统由开发人员针对自己的项目开发出来，只能使用到自己的项目之中，移植性非常的差，同时也相应提高了软件开发的成本。是各有特色的一种管理系统，不太方便软件开发人员的使用。在 JDBC 出现之后，通过使用其提供的标准化的接口数据库的开发工程师以及与前台相关的工具的开发者可以很方便使用 Java 语言编写与系统有关联的数据库应用程序[1]。使用 JDBC 技术可以很方便向各个数据库管理系统，发送相应的 SQL 语句，来对其中的数据进行相应的操作。通过利用 Java 的跨平台的特性以及其优秀的可移植性，开发人员只需完成一次应用的代码。当应用程序所部署到的平台或者时运行的操作系统支持 Java 虚拟机的运行，支持 Java 语言的编译，那么该应用程序就可以在任何的 Java 平台上平稳的运行，而不需要重新的进行配置。这种方式极大的表现了可移植性这一 Java 语言的优势。

2.2 JPA 技术

The full name of JPA is: Java Peristence API.JPA 在中文中译为 Java 持久层的标准化接口和类。是关于 Java 中的持久化数据存储的技术，现在已经渐渐独立，在前端开发中也有应用到，经过，对象-映射关系的 XML 文档来对 JPA 注解完成相干的描述，软件专家组为了方便的使用持久化的数据存储就开发了 JPA。其框架和接口相对的比较的简单，因此便于开发人员掌握。JPA 通过运用面向对象的语言来对数据库中的相干数据进行查询，实体对象在运行的时候能够持久化存储到相应的数据库中，JPA 对于实体对象的这一特性，不再对变量进行操作，转而对实体对象进行操作，通过这样的方式去实现持久化存储的目的。同时这种方式能够提供数据批量的查询功能，修改功能，在很大的程度上避免了直接和 SQL 语句的接触，以防止应用程序的耦合度过高。在使用 JPA 的时候要用到一些注解：@Entity,@Table,@Id,@Generated,@Basic,@Column,@Transient,@Temproal,@Enumerated,@Embedded,@ElementCollection

2.3 JNDI 技术

JNDI 以 JAVA 进行命名，作为一个接口可实现 JAVA 程序与命名服务、JAVA 程序与目录服务相交互的功能。通过 JNDI，利用统一和标准的 JNDI 接口，开发人员为了完成查询和访问的目录和命名服务能够利用这种接口方法。其主要用于访问数据库，通过 JNDI，可以将多个命名服务以及目录服务同步连接，通过 JNDI 可以实现命名和对象的真正关联，因为 JNDI 中所包含的接口和方法可以将名称和具体的 Java 对象关联起来，提高数据库的访问效率。但在使用时一定要注意，使用的目录服务是否有相关的提供者，如果没有将不能够使用该服务。相关实现的代码如下：

更改配置文件：

```
<resource-ref>
<description>Oracle DB Connection</description>
<res-ref-name>oracleDataSource</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<res-auth>Container</res-auth>
</resource-ref>
```

接下来对数据库进行操作即可

2.4 ORM 技术

Object Relational Mapping 是 ORM 的英文全名，其的中文名译为对象关系映射，ORM 是一种技术用来处理程序，其通过完成在面向对象的编程语言中将不同类型系统的数据进行转换[2]来完成其功能，因此 ORM 更像是一个中介的系统。通过 ORM 可以将不同的关系对象进行描述，同时其具有很强的适应性和灵活性通过 ORM 允许重用对象的特点，能够在数据库结构中完成数据模型的正确映射。ORM 的出现主要是为了弥补 JDBC 中的不可持久化存储的特点，通过 ORM 可以实现更快的数据查询，利用 ORM 的关系映射可以实现对数据库进行操作。

3.实现 Java 中数据库的访问

3.1 创建数据库，导入数据源

在 Java 中需要使用到数据库，在使用之前，需要先新建一个数据库，之后需要导入与系统或者应用关联的数据进入到数据库中，以便于后期的测试和管理。现在使用的比较广泛的开源数据库是 MYSQL，本文就以它为例进行相关的阐述；首先通过 NaVicat 软件或者是在操作界面，进入 MySql 数据库，打开界面之后，需要建立一个完整的数据库，在创建的时候需要记住相应的用户名和唯一的密码，这两种信息是进入数据库的钥匙。

之后可以在数据库中建立数据表即 Table，数据表可以是若干个，彼此之间有着相互联系，在表中可以存入相应的数据，并通过表之间的联系来对数据进行管理和支配。在数据库表创建完成之后，需要打开 NaVicat 软件，在数据库中确认并查看表中的数据是否存在，且数据是否正确。

3.2 注册并加载数据驱动

若要在 Java 程序中连接到数据库，就必需要注册数据驱动，注册时需要使用到一种桥驱动程序，英文简写是 JDBC-ODBC，在注册时可能需要用到如下的一些方法 ClassforName()、registerDriver（）是包含在 DriverManager 中的，数据库的相关驱动文件需要填写到对应的方法中，通过这种方式来实现数据驱动的注册，其中 ClassforName()只是注册驱动程序类，DriverManager 中可以通过方法 registerDriver 来注册驱动程序的实例类。在注册成功之后，就可以开始连接数据库的步骤。相关实现的代码如下：

```
try{Class.forName("com.mysql.jdbc.Driver") ;
}catch(Exception e){
System.out.println("出现问题的语句");
e.printStackTrace() ;
}
```

3.3 数据库的连接

通过利用 getConnection()方法来完成数据库的连接，这个方法属于 DeiverManager 类，举例为：Connection coc = DriverManager.Get connection(URL, user, pass),其中 URL 为数据资源的定位符，具体表明数据所在，USER 和 PASS 为数据库中的用户名和密码。当输入的用户名和密码正确，就可以与数据库获得连接，在连接之后，能够对数据库中的相关数据进行操作。相关的实现代码如下：

```
String url = "" ;
        String user = "用户名" ;
        String password = "密码" ;
try{
      Connection connection =
                  DriverManager.getConnection(url , user ,
password ) ;
        }catch(SQLException e){
        System.out.println("出现问题的语句");
        e.printStackTrace() ;
         }
```

3.4 通过数据库连接池进行连接

通过普通的数据库进行连接，在访问数量较小的时候，可以正常的完成任务。但是在数据访问量比较大的时候，普通的数据访问方式已经不再适宜，每一次连接数据库之后，关闭连接，再打开连接，如此反复，对于数据来说是很大的负担极易造成拥堵，降低了数据访问的效率。在连接池中要先存放合适的连接数目，在使用时，可以直接获取连接，且不使用时，释放连接即可，数据库的连接，交由连接池来掌管，极大的降低了数据库连接的压力[3]。在 Java 中可以选择合适的数据库连接池来进行使用完成驱动配置。部分实现代码如下：
读取 Properties 文件获取到 url、user、password 和连接数，之后进行连接。

```
Properties prop = new Properties();
FileInputStream fin = new FileInputStream(propFile);
prop.load(fin);
fin.close();
int InitSize =
Integer.parseInt(prop.getProperty("InitSize"));
```

3.5 访问数据库中数据，执行 SQL 语句

实现访问数据库。在 Java 中有三种不同的调用方式，第一种是 Statement 用于执行静态的 SQL 语句即在编译阶段就可以知道数据库的动作，SQL 语句的主体结构比较明确。第二种是 PreparedStatement（）方法，其来自 Statement（）方法的继承，且比父方法更加的有效，它执行的是动态的 SQL 语句，在编译时无法确定，因此能够从文本框中读取 sql 语句或者键盘输入的方式来确定 SQL 语句。第三种是 CallableStatement()方法，其是从 PrepareStatement 这个父方法中继承来的，用于执行 SQL 的存储过程，在使用的时候，您可以通过调用 Connection 对象的 upareCall()方法来获得 CallableStatement 对象。以便于使用。在这三种方法中使用比较多的是第二种方法，第二种方法能够很有效的防御 SQL 的注入攻击，其执行动态的 SQL 语句能够保证较高的安全性，而且更加的灵活。但是具体在使用的时候，需要根据得到结果的内容来选择相应的访问方

式。Execute()方法能够完成多个结果集的返回，删除和更新 SQL 语句的方法是 ExecuteUpdate()方法，通过执行特定的 SQL 语句，ExecuteQuery()方法可以或许返回单个结果集。

3.6 得到数据库相应数据

针对数据库执行了正确的 SQL 语句之后，会对应返回所需的数据，一般情况下，会利用 ResultSet 对象来对返回的结果进行操作。

4. Java 语言中数据库访问技术比较

JDBC 作为 Java 语言环境下，连接数据库的基础，起到了非常重要的作用，在以 JDBC 为基础的情况下，能够构建出更加高级的接口和工具，提升数据库的访问效率和软件开发人员的开发速度，也正是因为如此，JDBC 在某些方面稍显落后，新的技术相较于 JDBC 有着更好的性能，同时也有着不足之处。JDBC 的一大缺点就是将 SQL 语句和逻辑控制语句杂糅在一起，新的技术针对这方面都进行了相应的改进[4]。

4.1 访问效率比较

JDBC 技术对数据库中相关数据进行获取是通过利用对象的方式，其能够直接的对数据库进行相关联的操作，但是随着访问次数的增加，会出现拥堵的现象，JNDI 是通过命名和对象相互关联的方式来对数据进行操作，其效率要比 JDBC 高。JPA 由于其操作的对象是实体且对于开发人员来说，更加便于掌握，查询效率较高，ORM 通过映射的方式对数据进行操作有很强的灵活性，其同时弥补了 JDBC 的持久化存储问题。

4.2 可移植性比较

在 JDBC 中利用 SQL 的特点是把 SQL 语句内嵌进入，Java 的编程语言里面，对数据库操作的语句往往和 Java 语句掺杂在一起，程序的调试难度相对加大，同样的可移植性并不是很好，移植之后也同样会存在开发环境等问题需要解决，ORM 由于使用不同的框架问题，可移植性不高，JNDI 则必要有服务来提供和其相对应的功能，JPA 技术直接对实体对象进行操作，不与 SQL 语句相糅合，大大提高了可移植性。

4.3 操作复杂度比较

很明显在操作复杂度方面，JDBC 的难度要更高一些，在编写业务逻辑的同时，需要嵌入 SQL 语句，更容易出现问题，JPA 直接操作实体对象，ORM 有相应的框架，JNDI 通过接口实现服务，在操作复杂度上相对合适。

5.结束语

Java 语言在软件行业占据着重要地位，尤其是在大型应用系统的开发之中表现的更加出色，但是大型的应用系统往往会涉及到大量的数据交互，这对数据库访问技术带来了很大的挑战。高效的数据连接和处理会给软件带来更高的优化，提高数据交互的速度，提升用户体验。Java 具有的良好的可移植性与数据库相结合会带来更快的数据处理和软件开发方式。在开发过程中根据具体情况选择相应的处理数据方式，会大大提高开发的效率。随着 Java 开发方式的进步和数据库技术的优化，将会带来更加高效，安全的开发方式。

参考文献

[1] 谷庆华,李成贵.基于 Java 语言实现数据库的访问[J].计算机技术与发展,2008(02):13-16.
[2] 梁文菲, 黄厚宽. 对象/关系映射技术与面向对象数据库技术比较分析[J]. 中国科技信息, 2006(21):154-156.
[3]董伟.Java 程序中访问数据库的常用技术的比较分析[J].黑龙江科技信息,2012(33):85.
[4]周哲韫.基于 JAVA 语言的数据库访问技术[J].电子技术与软件工程,2017(08):199.
[5]王建.基于 Java 语言的数据库访问技术应用研究[J].计算机产品与流通,2017(08):24-25.

作者简介

第一作者：李俊辉（1998-），男，汉，河南省信阳市，本科，四川大学锦城学院，研究方向：Java。
第二作者（通讯作者）：鲍正德（1989-），男，汉，黑龙江哈尔滨市，研究生，四川大学锦城学院，研究方向：电子商务。
第三作者：唐娅雯（1999-），女，汉，四川省资阳市，本科，四川大学锦城学院，研究方向：信息管理、J2EE