基于 Berkeley DB JE 的简易通用数据库的设计和实现

Design and Programming of Simple Common DB Based On Berkeley DB JE

张少龙　(武汉大学 信息管理学院　430072)

摘要:这篇论文主要阐述以嵌入式数据库 Berkeley DB 的 java 版本为基础,利用 java 语言的接口,串行化,动态加载类等特性设计一个具有可配置、通用跨平台的简易数据库系统的可行性。本文通过一个简单的考试成绩查询为示例进行了系统设计,并通过 java 程序加以实现。

关键词:嵌入式数据库　Berkeley DB java　动态加载类　接口　串行化

1　嵌入式数据库 Berekley DB 简介

Berkeley DB 是 sleepycat 公司(www. sleepycat. com)开发的开放源代码的内嵌式数据库管理系统,能够为应用程序提供高性能的数据管理服务。应用它程序员只需要调用一些简单的 API 就可以完成对数据的访问和管理。与常用的数据库管理系统(如 MySQL 和 Oracle 等)有所不同,在 Berkeley DB 中并没有数据库服务器的概念。应用程序不需要事先同数据库服务建立起网络连接,而是通过内嵌在程序中的 Berkeley DB 函数库来完成对数据的保存、查询、修改和删除等操作。

1.1　基本概念

Berkeley DB 简化了数据库的操作模式,同时引入了一些新的基本概念,从而使得访问和管理数据库变得相对简单起来。在使用 Berkeley DB 提供的函数库编写数据库应用程序之前,有必要先了解以下这些基本概念。

关键字(Key)和数据(Data)是 Berkeley DB 用来进行数据库管理的基础,由这两者构成的 Key/Data 对(见表 1)组成了数据库中的一个基本结构单元,而整个数据库实际上就是由许多这样的结构单元所构成的。通过使用这种方式,开发人员在使用 Berkeley DB 提供的 API 来访问数据库时,只需提供关键字就能够访问到相应的数据。

如果想将第一行中的"sport"和"football"保存到 Berkeley DB 数据库中,可以调用 Berkeley DB 函数库提供的数据保存接口。此时"sport"和"football"将分别当成关键字和数据来看待。之后如果需要从数据库中检索出该数据,可以用"sport"作为关键字进行查询。此时 Berkeley DB 提供的接口函数会返回与之对应的数据"football"。

表 1　Key/Data 对

Key	Data
sport	football
Fruit	orange
Drink	beer

BDB 提供了相对简单的数据访问服务。BDB 只支持对记录所做的几种逻辑操作。它们是:

(1) 在表中插入一条记录。

(2) 从表中删除一条记录。

(3) 通过查询键(key)从表中查找一条记录。

(4) 更新表中已有的一条记录。

1.2　Berkeley DB JE(JAVA Edition)

尽管 Berkeley DB 提供了其他语言(包括 java)的接口,但 Berkeley DB 最初的版本是用 C 写成。而 Berkeley DB JE(JAVA Edition)的出现使 java 开发者也能够使用 Berkeley DB 的各种优秀特性,并通过对 java 数据类型的支持使之更好的整合入 java 世界中而避

免使用 JNI 带来的性能损失。

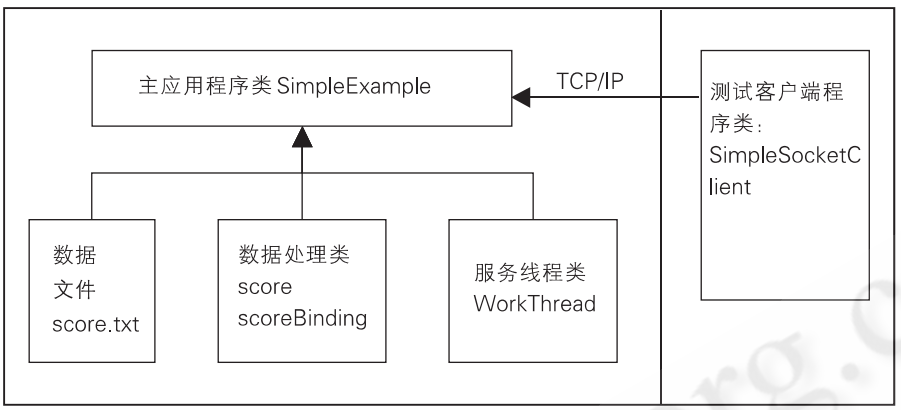

图 1　系统结构图

像 C 的先辈一样，JE 也很小（小于 435k）。但它能提供完备的事务支持（原子操作、一致性、单独性、持久性）和高并发情况下的记录级的锁操作。同时它支持单表上百 TB 的数据和单条 2G 的记录，并能提供数据的高速缓存。

现在 JE 的最新版本是 1.5.3，需要 JDK1.4 以上支持。在以下的示例中均以此版本为基础。

2　示例数据库设计和实现

2.1　示例说明

尽管 Berkeley DB JE 为我们实现一个简单、通用的，具有可移植的数据库查询系统提供了基础，但这是远远不够，还需要充分利用 java 语言各种特性包括对象的串行化、类的动态加载、接口（Interface）、多线程等才能够真正的实现。

下面就一个简单的数据库查询系统的示例来说明。该数据库为一考生成绩数据库，字段包括考生考号、考试成绩，考号为唯一键值。客户通过考号查询考试成绩。

考生成绩数据文件为 score.txt（以#作为分割符：考号#成绩）：

2001#95

2002#99

2003#65

……

2.2　系统设计和实现

2.2.1　系统架构设计（如图 1）

根据系统要求，设计了下面几个类：

（1）Score.class：用来实现数据对象的存储和串行化。

（2）ScoreBinding.class：用来将一个 Score 类的对象转换为 Berkeley DB 中的 DatabaseEntry 对象。

（3）SimpleExample.class：主程序，实现数据导入或数据查询服务。

（4）SimpleSocketClient.class：客户端程序通过 TCP/IP 协议发送请求，SimpleExample 调用 WorkThread 线程类来进行数据查询并返回数据给 client。

下面分别就这些类的实现来进行对系统实现的要点阐述如下：

2.2.2　Score 类和 ObjectInterface 接口

该类完成学生考号和成绩数据对象的存储和串行化，串行化是 java.io 包中的一部分，它被用来将对象转换成一串字节。在串行化的过程中，将一个实体转变成一系列表示对象的字节，这些字节可以写入文档以备后用，通过网络连接传输到其他程序，用来对初始对象进行拷贝等等，需要串行化一个类，那么就必须对这个类执行 java.io.Serializable，所以 Score 类实现了 Serializable 的接口。

另外考虑到通用的目的，又设计了一个 ObjectInterface 的接口，该接口提供了统一的数据导入和存取数据的方法，Score 类同样也实现了该接口。

```
public interface ObjectInterface {
/* 通过数组倒入数据，sArray[0]为键值，sArray[1]为数据对象 */
public void loadData(String[] sArray);
/* 返回键值 */
public String getKey();
/* 返回数据内容 */
public String getData();
}
/* 实现 Serializable 和 ObjectInterface 接口 */
public class Score
  implements Serializable, ObjectInterface {
```

```
public Score( ) {
}
/ * 学生考号 * /
private String testCode;
/ * 学生成绩 * /
private String testScore;
/ * 实现统一的数据导入接口 * /
public void loadData( String[ ]sArray) {
 setTestCode( sArray[0] ) ;
 setTestScore( sArray[1] ) ;
}
public String getKey( ) {
 return getTestCode( ) ;
}
public String getData( ) {
 return getTestScore( ) ;
}
/ * 下面实现对象串行化 * /
public void setTestCode( String data) {
 testCode = data;
}
public void setTestScore( String data) {
 testScore = data;
}
public String getTestCode( ) {
 return testCode;
}
public String getTestScore( ) {
 return testScore;
}
}
```

2.2.3 ScoreBinding 类

ScoreBinding 类将一个 Score 类的对象转换为 Berkeley DB 中的 DatabaseEntry 对象，只有转换为 DatabaseEntry 对象后，才能存入 BerkeleyDB 中。该类继承 TupleBinding 类并实现 entryToObject（从 DatabaseEntry 到 Score 类的对象）和 objectToEntry（从 Score 类的对象到 DatabaseEntry）两个方法。

```
public Object entryToObject( TupleInput input) {
/ * 读入 DatabaseEntry 对象中的数据 * /
 String testcode = input. readString( ) ;
 String testscore = input. readString( ) ;
/ * 将数据写入 score 对象中并返回 * /
 Score score = new Score( ) ;
 score. setTestCode( testcode) ;
 score. setTestScore( testscore) ;
 return score;
}
public void objectToEntry( Object object, TupleOut-
  put output) {
  Score score = ( Score) object;
  / * 将 score 对象的数据写入 DatabaseEntry * /
  output. writeString( score. getTestCode( ) ) ;
  output. writeString( score. getTestScore( ) ) ;
 }
}
```

……

2.2.4 SimpleExample 类

该类为该示例程序的入口类，提供了 main( ) 方法，主要是配置数据环境并通过输入参数的选择完成数据文件的导入或数据查询服务。

（1）数据导入

……

```
/ * 建立一个新的提供事务处理的数据环境 * /
 EnvironmentConfig envConfig = new Environ-
 mentConfig( ) ;
 envConfig. setTransactional( true) ;
 envConfig. setAllowCreate( true) ;
/ * envDir 为数据库环境目录，数据库将建立在该
 目录下 * /
 Environment exampleEnv = new Environment
 ( envDir, envConfig) ;
 / * DatabaseEntry 表示 每条记录的主键和数据
 * /
 DatabaseEntry keyEntry = new DatabaseEntry
 ( ) ;
 DatabaseEntry dataEntry = new DatabaseEntry
 ( ) ;
 / * 在该数据环境下建立一个数据库 * /
 Transaction txn = exampleEnv. beginTransaction
 ( null, null) ;
```

```
/ * 建立数据库配置选项 * /
DatabaseConfig dbConfig = new DatabaseConfig( ) ;
/ * 设置数据库支持事务提交 * /
dbConfig. setTransactional( true) ;
/ * 设置允许新建数据库 * /
dbConfig. setAllowCreate( true) ;
/ * 设置数据库支持重复键值和键值排序 * /
dbConfig. setSortedDuplicates( true) ;
/ * 建立数据库,数据库名为"simpleDb" * /
Database exampleDb = exampleEnv. openDatabase( txn,
                                    " simpleDb" ,
                                     dbConfig) ;
txn. commit( ) ;
/ * 打开 ClassCatalogDB ,这个数据库用于类串行化的优化 * /
DatabaseConfig catdbConfig = new DatabaseConfig( ) ;
catdbConfig. setTransactional( true) ;
catdbConfig. setAllowCreate( true) ;
catdbConfig. setSortedDuplicates( false) ;
txn = exampleEnv. beginTransaction( null, null) ;
Database classCatalogDb =
   exampleEnv. openDatabase( txn,
            " ClassCatalogDB" ,
            catdbConfig) ;
txn. commit( ) ;
StoredClassCatalog classCatalog = new StoredClassCatalog( classCatalogDb) ;
…….
/ * 将数据文件中的学生成绩数据导入 simpleDb 中,loadFile 方法是将一个文件数据的数据放入一个数组列表中,方法第一个参数为文件名,第二个参数是文件中的字段数,在这个示例中只有两个字段学生考号和成绩
 * /
ArrayList scores = loadFile( exampleFile, 2) ;
// Now load the data into the database. The vendor's name is the
// key, and the data is a Vendor class object.
// 对数据串行化绑定类实例化
EntryBinding dataBinding =
    new SerialBinding ( classCatalog, Score.class) ;
TupleBinding binding = null;
binding = ( TupleBinding) Class. forName( "ScoreBinding" ) . newInstance( ) ;
//将数据从数组列表中取出并存入 score 对象(以下代码删略)
……
```

(2)查询服务

通过监听端口接收客户端发来的请求,并通过 WorkThread 类实现进行多线程处理。

```
//开始监听
    server = new ServerSocket ( SimpleExample.PORT) ;
    ssocket = server. accept( ) ;
//多线程处理
    WorkThread t = new WorkThread( exampleDb, ssocket) ;
    t. start( ) ;
```

3 小结

读者可以注意到在示例程序中许多变量都是写定在程序中,如动态加载库的类名,文件名等。因为首先这只是个简单示例程序;其次使用配置文件或者通过 XML 文件方式进行配置等是比较简单的,也不是示例的重点,重点在于如何使用 Berkeley DB 的 API,希望这篇文章能够起到抛砖引玉的作用,为广大的开发人员对嵌入式数据库的使用提供一种新的思路。

参考文献

1 Anton Okmianski 编译/ 林崇亮,嵌入式数据库 – 朴素但实用的数据库选择,《程序员》,2003.1。

2 http://www. sleepycat. com

3 http://developer. ccidnet. com/pub/article/c310 _ a101733_p1. html,开源嵌入式数据库 Berkeley DB (1) – 嵌入式应用 – 技术天地 – 赛迪网。