基于 JDBC 数据库时间获取方法

Time Achievement Methods Based on JDBC

晏　立　阎蔚明　陶跃华　兰美辉　（云南师范大学 计算机科学与信息技术学院 云南 昆明 650092）

摘　要： JDBC API 是 JAVA 访问关系数据库的编程接口，但各个数据库开发商对接口的实现并不完全相同，本文研究并测试了不同数据库及其 JDBC 驱动程序在不同时区获取数据库时间的情况；对在不同时区不能正确读、写数据库时间的数据库及其 JDBC 驱动，提出了调整算法，使用调整算法进行调整后，都得到正确的结果。

关键词： 数据库 JDBC 时区 时间

1 引言

在用 JAVA 开发报表工具时，发现这样一个问题，不能通过 JDBC API 从某些数据库中获取正确的时间[1,2]。比如：SQLServer 2000 的数据库系统，假设系统有一条销售记录：A 部门在时间 2008－06－22 10:00:00 GMT ＋9:00 销售了商品 X，若在 A 时区（GMT－8:00）取这条销售记录，得到的销售时间是 2008－06－22 10:00:00 GMT －8:00；在 B 时区（GMT＋8:00）取这条销售记录，得到的销售时间是 2008－06－22 10:00:00 GMT ＋8:00。显然数据库中的时间和分别在两个时区获取的时间是不相等的。

为解决上面这个问题需要做两方面的工作，一是研究、检测常用的数据库产品及其 JDBC 驱动程序，以发现在不同的时区中，使用 JDBC API 获取的时间不正确的数据库产品；二是提供一种调整时间的算法，该方法要能够把从数据库中取到的不正确的时间调整为正确的时间，调整后的时间仅用于在报表中使用，不涉及修改数据库时间和 JDBC 驱动程序。

1.1 JDBC API

JDBC API[1,2]（Java Data Base Connectivity，java 数据库连接）是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一访问，它由一组用 Java 语言编写的类和接口组成。JDBC 提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。

JDBC API 只是一个规范，各个开发商的接口并不完全相同。

1.2 时区

时区，我们知道地球是有自转的。因此地球上不同地区的人民为了更方便的表达时间，通常都是以自己所处地区的昼夜变化来定义 24 小时的（我们有时直接使用 12 个小时再加上日夜来表达时间）。通常地区所处的政治实体会定义对时间描述的标准，这通常用时区的概念来表达。因此，我们有了一个基准时区，它所处的时间被称为"格林威治标准平时"（Greenwich Mediation Time）。其他地区的计时往往是相对于这个时间的一个小时整数倍数偏差（offset）。从地理上而言，地球就有 24 个时区。但地理上区域的定义往往与政治实体的区域定义不一致，所以在实际的生活中我们所讨论的时区是基于政治实体的区域界定。另外，由于"夏"令时（Daylight Saving Time）的引入，人们会人为地调整时区的定义，以至于一个地区在一年中的不同时间相对于 GMT 的偏差是不同的。

1.3 JAVA 时间类

java.util.Date 这个类保存的是一个 long 值[3]，这个值代表了当前系统的时间离 1970/1/1 0:0 的毫秒差，而且是在 GMT 0:00 时区下的毫秒差。如果我们指定的时间是在此之前的，那它将返回一个负数值。本文提出的时间调整方法就是调整该毫秒差。

java.sql.Date 是 java.util.Date 的子类[3]。

java.util.Calendar 是一个抽象类[3]，它为特定瞬间与一组诸如 YEAR、MONTH、DAY_OF_MONTH、HOUR 等 日历对象之间的转换提供了一些方法，并为操作日历对象（例如获得下星期的日期）提供了一些方法。GregorianCalendar 是它的一个具体实现。

java. util. TimeZone 表示时区偏移量[3]，这个类保存的偏移量是一个 long 值，表示当前时区和 GMT 0：00 之间的毫秒差。

2　不同数据库及 JDBC 在不同时区获取时间情况测试分析

测试分为两个部分：从 java. sql. ResultSet 中读取时间对象和用时间作为数据查询条件。

首先构造数据库测试环境，插入时间到数据库，设置 JAVA 虚拟机默认的时区为 A（GMT－8：00），用 JDBC API 插入时间对象到数据中。

2.1　读取时间

java. sql. ResultSet 提供了两类方法从数据库中读取时间对象[3]：

读方法 1：直接获取时间对象

getDate（int columnIndex）

getTime（int columnIndex）

getTimestamp（int columnIndex）

读方法 2：获取使用 calendar 参数调整的时间对象

getDate（int columnIndex，Calendar cal）

getTime（int columnIndex，Calendar cal）

getTimestamp（int columnIndex，Calendar cal）

2.2　用时间作为查询条件

用时间作为数据查询条件，查询满足条件的数据，为此需要写时间到数据库，JDBC 规范提供三种不同的构造时间查询条件的的方法[3]：

写方法 1：使用带有输入参数的 SQL 语句，并用直接设置时间对象的方法设置时间参数。

PreparedStatement . setDate（int idx，Date x）

写方法 2：使用带有输入参数的 SQL 语句，并用设置时间对象及 calendar 调整参数方法设置时间参数。

PreparedStatement . setDate（int idx，Date x，Calendar cal）

写方法 3：不使用带有输入参数的 SQL 语句，直接把时间对象用默认时区格式化成时间字符串，并添加到 SQL 中的 Where 子句。

2.3　测试

调整 JAVA 虚拟机默认时区为 B（GMT＋8：00），分别使用读方法 1，读方法 2，写方法 1，写方法 2，写方法 3 测试不同的数据库及 JDBC，结果如表 1 所示，读取时能正确得到时间的标记为 OK，能正确查询到数据的标识为 OK。

从表中可以看出：部分数据库及 JDBC 在不同时区不能获取正确的时间，时间作为数据查询条件时不能查询到需要的记录。

表 1　在时区 B（GMT＋8：00）读、写时间的测试

数据库及 JDBC 驱动版本	读方法 1	读方法 2	写方法 1	写方法 2	写方法 3
DB2 8.1.7 JDBC：2.0 Type 3，08.02. 0000		OK		OK	
Informix：9.1.4 JDBC：2.21. JC5					
Oracle 8.1.7 JDBC：10.1.0.2.0	OK		OK		OK
Oracle 9.2.0.1.0 JDBC：10.1.0.2.0	OK		OK		OK
Oracle 10.1.0.2.0 JDBC：10.1.0.2.0	OK		OK		OK
SQL Server 2000 JDBC：2.2.0037		OK		OK	
SQL Server 2005 JDBC：2005 1.1.1501.101		OK		OK	
Sybase 11.5 JDBC：5.5					
PostgreSQL 8.0 JDBC：8.0	OK		OK		OK
MySQL 5.0 JDBC：5.0.4	OK	OK	OK	OK	OK

注：对读方法，读取时若能得到正确的时间的标记为 OK，对写方法，能正确查询到数据的标识为 OK。

读方法 2 和写方法 2 中，calendar 的时区设置为 A（GMT－8：00）（即：插入时间对象的时区）

3 调整不正确时间的算法及测试

3.1 算法

从表 1 中可以看出,使用读方法 2 和写方法 2 可以在部分的数据库及 JDBC 中处理正确的时间,但这两个方法在实际的开发中很难使用,这是因为不能正确的获得数据输入时区,比如:插入数据是由其他程序完成的;或插入数据的程序不是用 JAVA 开发的。

因此需要其它的方法来调整获取时间和构造时间查询条件,本文提出了基准调整量算法来解决获取时间和构造查询条件的问题,包括三个算法:获取基准调整量、取时间调整、写时间调整。获取基准调整量算法是基础,其它两个算法都需要使用基准调整量。基准调整量的基准时区是 GMT +00:00),这是为了得到的基准调整量可以在任何时区使用。

算法 1:获取基准调整量

1) 使用取方法 1 从数据库中取时间,记为 Timestamp1;

2) 显示 Timestamp1 给用户;

3) 用户输入期望的正确时间,记为 Timestamp2

4) 计算基准调整量(基准是 GMT +00:00),offset = Timestamp2 - Timestamp1 - (JAVA 虚拟机默认时区的偏差)

算法 2:取时间调整

1) 使用取方法 1 从数据库取得时间,记为 Time1

2) 计算正确的时间,CorrectTime = Time1 + offset + (JAVA 虚拟机默认时区的偏差)

算法 3:写时间调整

1) 把要输出到数据库时间对象记为 Time1

2) 计算输出时间,OutputTime = Time - (offset + (JAVA 虚拟机默认时区的偏差))

3) 用 JAVA 虚拟机默认时区把 OutputTime 格式化成字符串,用写方法 3 生成的 SQL 的 Where 子句。

3.2 调整算法测试结果

经过测试,在原本不能得到正确结果的数据库及 JDBC 中,使用获取调整算法后得到了正确的时间;使用输出调整算法后也查询到了正确的记录,测试结果如表 2 所示。

表 2 在时区 B(GMT +8:00)使用调整算法调整时间后的测试结果

数据库及 JDBC 驱动版本	算法 2	算法 3
DB2 8.1.7 JDBC: 2.0 Type 3, 08.02。0000	OK	OK
Informix: 9.1.4 JDBC: 2.21. JC5	OK	OK
SQL Server 2000 JDBC: 2.2.0037	OK	OK
SQL Server 2005 JDBC: 2005 1.1.1501.101	OK	OK
Sybase 11.5 JDBC: 5.5	OK	OK

4 结论

综上所述,我们得出以下结论:

Oracle8.1.7, Oracle9.2.0.1.0, Oracle 10.1.0.2.0, PostgreSQL 8.0, MySQL 5.0 在不同时区读、写数据库时间时均能得到正确的结果,在需要跨时区处理数据时建议优先使用他们。

对在不同时区不能正确读、写数据库时间的数据库及 JDBC 使用本文提供的方法进行调整后,都得到正确的结果。

参考文献

1 JDBC 3.0 http://www.jcp.org/en/jsr/detail?id=54

2 JDBC 4.0 http://www.jcp.org/en/jsr/detail?id=221

3 JAVA API http://java.sun.com/javase/6/docs/api/