C/S 结构数据库的数据完整一致性研究

孙光咏 田忠和 (华中理工大学 430074)

摘要:本文从分析客户机/服务器方式下破坏数据库的完整一致性的问题出发,结合大型数据库管理系统 SYBASE 在公安部门人口管理应用的实例,说明了如何维护数据完整一致性的几种方法。

关键词:客户机/服务器(client/server) 触发器 事务 并发

一、问题的提出

随着计算机网络技术的日益发展和应用,各企事业单位纷纷构建企业或部门内部的网络体系。作为企业局域网的一个重要的应用组成部分,企业数据库技术一直是业界关注的焦点,各部门单位纷纷将原来单机方式的数据库管理系统扩展成适应网络环境的客户机/服务器(client/server)结构的应用系统,以实现资源和数据的共享。

在客户机/服务器体系结构中,客户机是直接面向用户的,它接收并处理任务,将任务转化为 SQL 请求交给服务器执行。数据库服务器可以连接多台终端客户机,主要负责数据的存取管理,处理客户机的查询修改等操作请求,并将数据和结果返回给客户机。

由于在网络系统中,可能存在着许多客户同时并发地访问存取数据库中的相同的数据,如果数据库服务器不对这些并发执行的事务进行处理和控制,就有可能造成不正确的结果和数据库的不一致状态。同时由于网络或机器的故障(如断电、强行关机等)使正在进行的数据存取操作中断,加上一些人为的误操作,均可能使数据库的完整一致性受到破坏。因此,要开发好网络客户/服务器方式下的应用系统,保证数据操作的正确性和可靠性,必须对数据库的数据完整一致性有较深入的认识和考虑。

二、问题的分析

在网络客户/服务器体系结构下,应用的处理是分布在各个客户机上的,数据库服务器集中处理数据的修改、分类和检索,管理客户的事务和事务逻辑所涉及到的安全性。在这种体系结构下,造成数据库的完整一致性受到破坏的原因有多种,大致可以分为以下几种:

1. 由网络或机器故障造成

网络环境的不稳定,机器故障(如断电、硬盘损坏等)造成正在存取的数据发生错误,导致不正确的结果。

2. 由人为的对数据的操作造成

此类操作可能是由数据库管理员 DBA 不经过应用程序,而是通过一些数据库的操作平台如 ISQL, Microsoft SQL/W 等工具直接对数据库进行删除、修改和插入等操作,使得一些不符合企业规则的数据进入数据库,破坏了数据的完整性。如在人口管理系统中,人为地将某人的记录从基本信息表删除(注销),却没有将此人的记录转入历史信息表中。

3. 由应用程序的不完善造成

有些应用程序可能处在调试或试用阶段,或程序设计考虑不周,也可能造成非法数据进入数据库,破坏数据库的完整一致性。如数据的合法性校验不完全,造成制出的某人身份证的年龄小于合法年龄(如 18 岁)。

4. 由多个事务并发执行造成

事务是由数据库管理的最小的逻辑工作单元。单个事务单独执行可能是正确的,但多个事务同时并发交错地执行,造成相互干扰,使客户得不到正确的结果。尤其是在多用户环境中,数据库必须避免同时进行的查询和更新发生冲突所造成的数据存取时的"脏读"(Dirty Data)对数据完整性的破坏。

三、问题解决的方法

下面结合 SYBASE 数据库在公安部门人口管理中的应用实例,介绍几种有利于维护数据完整一致性的方法:

1. 使用数据库完整性约束

数据库完整性约束是用于维护数据库完整性的一种机制,这种约束是一系列预先定义好的数据完整性规划和业务规则,这些数据规则存放于数据库中,防止终端用户输入错误的数据,以保证所有数据库中的数据是合法的、完整的。

具体来说,数据库的完整性约束有以下几种:非空约束(NOT NULL);缺省值约束(DEFAULT VALUE);唯

一性约束(UNIQUE);主键约束(PRIMARY KEY);外部键约束(FOREIGN KEY);规则约束(CHECK)。这种约束是加在数据库的表的定义上的,它与应用程序中维护数据库的完整性不同,它不用额外地书写代码,代价小而且性能高。

如下面的例子,在 SYBASE 中创建一个数据表 phone-tab 用于存放各人的电话号码:

```
CREATE TABLE phone-tab(
  id-code char(20) PRIMARY KEY , name char(20) NOT NULL ,
  sex char(1) NULL CHECK ( sex = 'M' or (sex = 'F')),
  age char(3) NULL , area-code char (5) NULL ,
  phonenumber char(20) NULL UNIQUE, address char (50) NULL )
```

在创建的这个表中,主键 PRIMARY KEY 来约束列 id-code 不能重复,而且不能为空,姓名字段 name 不能为空(NOT NULL),唯一性约束 UNIQUE 约束列 phonenumber 是唯一的,使用规则 CHECK 约束性别列 sex 的取值在'M'和'F'间。

2.使用数据库存储过程

存储过程是由流控制和 SQL 语句书写的过程,它是一组经编译和优化后存储在数据库服务器的 SQL 语句,使用时用户只要调用即可。这种已经编译好的过程可以极大地改善 SQL 的性能,而且执行速度快,可以大大减少网络 I/O 流量,提高应用系统的性能。尤其是在多网络用户的客户/服务器体系下,需要对多表进行插入、删除、更新等操作时,使用存储过程可以有效地防止多客户同时操作数据库时带来的"死锁"和破坏数据完整一致性的问题。

如在公安部门暂住人口系统中,当某人从基本库中注销后(删除 base-tab 表中相应记录),也应将他携带的儿童从表 child-tab 中删除,同时将其基本记录转入历史表(history-tab)中,可以写如下存储过程:

```
CREATE PROCEDURE sp-del-base
  @id-code char(20) AS
BEGIN
  DELETE FROM child-tab WHERE id-code = @id-code
  INSERT INTO history-tab SELECT * FROM base-tab
WHERE base-tab.id-code = @id-code
  DELETE FROM base-tab WHERE base-tab.id-code = @id-code
END
```

这里,存储过程将对这三个表的删除和插入操作当作一个事务来处理,这三个操作要么全成功(提交),要么全失败(回滚),保持了数据的一致性。

3.使用数据库触发器

触发器是一种特殊的存储过程,不同的是这种存储过程不是由应用程序的调用来执行,而是通过对数据库表进行插入、更新和删除操作自动地"触发"执行。它的主要优点是不管什么原因造成数据库变化时(无论是客户通过应用程序还是数据库管理员 DBA 绕过应用程序直接在后台操作数据库)都自动响应,回退那些破坏数据库完整性的操作。

具体来说,触发器通过引用列或其他数据库对象,产生比数据库规则更为复杂的完整性检查和约束,如触发器可以回退任何企图在库存量为零时增加系统出库记录的操作。同时,它可以对数据库中相关的表进行连环更新,来实现非标准的数据库相关完整性规则,保持数据的相关完整性,如在某一表上的更新触发器可以导致更新其他表中与它的主键值匹配的记录的内容。

例如,上面例子也可以通过在 base-tab 表上的删除触发器来实现,代码如下:

```
CREATE TRIGGER trg-base-tab-del ON base-tab
  FOR DELETE AS
BEGIN
  DELETE FROM chid-tab
    FROM deleted WHERE child-tab.id-code = deleted.id-code
  INSERT INTO history-tab SELECT * FROM deleted
END
```

值得说明的是,触发器并非越多越好,触发器建立的时候须特别小心,因为它不便于调试,而且过多的触发器也给维护带来麻烦,设计不好容易导致连环触发,造成"死锁"。

4.并发事务的处理

在客户机/服务器结构下,数据库集中数据的管理和共享。客户机是通过事务这种机制来操作数据库,事务将多个 SQL 语句当作一个工作单元来处理,这组 SQL 语句执行后,要么全部成功,要么全部失败。通过对事务的控制,数据库可以控制并发执行的查询和更新操作,也可以在系统出现故障后,数据库自动地从事务日志中恢复。

在多用户环境中,可能存在多个事务同时并发地存取相同的数据,若不进行处理和控制,会造成从数据库读出的数据与实际数据不一致(即"脏读 Dirty Data")的现象,甚至可能造成数据库的"死锁"。控制这种并发事务的最好方法是利用数据库的封锁机制。

封锁就是事务请求数据库管理系统对其所操作的数

据对象加锁(LOCK),其他事务必须等到此事务结束并释放锁(UNLOCK)后,才能对该数据对象进行操作。通过这一机制,可以避免多个事务并发执行存取同一数据时出现的数据不一致问题。

SYBASE 数据库的锁是加在数据页上,提供三种基本的锁类型:

(1)共享锁(Shared Lock):在读操作时加共享锁,缺省状态下,读操作完成后即释放共享锁。

(2)排它锁(Exclusive Lock):在更新操作时加排它锁,一旦一个事务对某一数据页加了排它锁后,其他任何事务都不能对该页进行读取和修改等操作。

(3)更新锁(Update Lock):在对要进行修改或删除的数据页进行操作前加更新锁。在表上加了更新锁后,还可以加共享锁,但不能再加排它锁或其他更新锁。在数据库进行修改或删除操作时,如果没有共享锁存在,此锁升级为排它锁(Ex)。这种锁主要是为了防止数据库的"死锁"。

下表说明了这三种封锁的相容关系:

请求锁 \ 原有锁	共享锁(Sh)	更新锁(Update)	排它锁(Ex)
共享锁(Sh)	是	是	否
更新锁(Update)	是	否	否
排它锁(Ex)	是	否	否

事务在读写数据时,必须先发出共享锁(Sh)或排它锁(Ex)的请求,直到获得必要的锁后才能进行读或写的操作。为解决"脏读"和"不可重复读"带来的数据一致性问题,SYBASE 数据库使用 HOLDBACK 关键字,此时共享锁并不是在读操作完成后即释放,而是要等到读事务完成(COMMIT 或 ROLLBACK)后才释放。例如,统计性别为'男(M)'的人的平均年龄:

```
SELECT AVG(age)
FROM base-tab HOLDBACK
WHERE base-tab.sex = 'M'
```

这里 HOLDBACK 关键字保证在计算列 age 的平均值时,表 base-tab 中的数据不会有任何改变,从而读取正确的数据。

对于"死锁",SYBASE 有一种自动检测机制,当"死锁"发生时,数据库自动回滚某进程中 CPU 时间最少的事务,允许其他事务继续执行。为避免"死锁"造成的数据库的不一致性,应用程序应当尽早地提交事务,少用 HOLDBACK 关键字,尽量使用存储过程。

5.软件设计时的特殊处理

在进行应用软件设计时,根据具体的功能要求,通过采取一些特殊的处理方法,也可以对并发执行的事务进行控制,维护数据库的完整一致性。如可以对数据设定标志位(flags)作相应的标记,或对不同的客户设定不同的优先级别,均可以实现对并发事务的控制。下面的例子通过设定标志为字段 k-flag 的值来实现对共享数据的并发存取。

在人口管理系统中,制证中心需要提取人口基本信息表 base-tab 中的制证标志 k-flag 为'N'的记录数据,但如果多台客户机同时提取,可能造成大家读取的数据相同,这样可能出现重复制证。

为避免这一问题,先对制证中心的这几台客户机编号,每次提取数据前先对数据进行"锁定",程序如下(假定@serial-no 为客户机编号):

```
UPDATE base-tab SET k-flag = @serial-no WHERE k-flag = 'N'
SELECT * FROM base-tab WHERE k-flag = @serial-no
.....
```

这样处理后,就好象每个客户机将共享数据按客户机编号进行了"划分",在 UPDATE 操作完成后就可以提交事务,从而也就有效地避免了多个客户读取相同数据的重复性问题。当然,具体情况具体对待,各种好的方法还须在实际中不断总结和摸索。

四、结束语

数据库的完整一致性问题是每一个数据库应用系统普遍存在的问题,本文提出的关于解决这一问题的方法,主要是从数据库服务器端数据字典的设计和系统的整体的角度来说明,这些方法经我们开发的浙江省公安人口管理应用系统中的证实,不仅可以较好地解决数据完整一致性问题,而且还可以大大提高应用系统的性能。

参考文献

[1] 王珊主编, Sybase 关系数据库系统原理和指南. 北京:中国科学技术大学出版社,1993

[2] System Administration Guide for Sybase SQL Server, Sybase Inc., 1992

[3] 珠联璧合 - - PowerBuilder 与数据库配合开发技术. 北京:晓通数据库研究所

(来稿时间:1998 年 10 月)