# ORACLE7 数据库锁的研究及其应用

谭太龙 高红梅 (北方交通大学自动化所 100044)

**摘要**:数据库锁技术为保证数据并发性和数据一致性的有效手段,是 RDBMS 的关键技术之一。本文以 ORACLE7 数据库在规范化编组站中的应用为例,对关系数据库管理系统的数据库锁的种类及其应用进行了研究。

**关键词**:并发控制 数据库锁 DML 锁

## 1. 引言

我们知道,数据库锁是数据库研究中的一个较活跃的领域。现在各种不同的并发控制方法在集中式与分布式的数据库中得到广泛的应用。例如:封锁、时间标志与验证等方法互相混用,其中封锁技术是最常见和有效的并发控制技术。比如在规范化编组站管理信息系统(SYIS)的应用中,其核心就是各种类型的锁定。由于 SYIS 是一个比较大的管理信息系统,用户较多,因此,数据库锁的地位尤其重要,特点是没有安全的死锁检测机制。为了解决这个问题,通过在现场不断的摸索,提出了二阶段非等待锁机制,从而解决了死锁的问题,提高了程序运行的效率。

## 2. ORACLE 数据库中的各类锁

ORACLE 在各种不同的开发应用环境中,根据封锁对象和封锁方式的不同,可划分出多种不同类型的锁。如下表 1 所示:

| 封锁者 | 锁的类别 | 封锁对象 | 封锁方式 |
|---|---|---|---|
| 用户 | DML 锁 | 表/视图 | 由用户显示或隐式封锁,并由用户解锁 |
| | ROW 琐 | 记录 | 由用户隐式封锁,用户自行解锁 |
| 系统 | DDL 锁 | 数据字典/表 | 封锁/解锁不由用户干涉 |
| | 内部锁 | 内部结构 | 视图 系统自动封锁解锁 |

实际上,ORACLE 中任何一种锁的操作最终都是调用封锁管理程序来完成,而它是 ORACLE 内核数据库中的一个组成部分,具有 LOCK 与 UNLOCK 两大功能。下面给出它们的大致算法。

(1) LOCK 的算法。一般调用格式:LOCK(s _ jd, locktype, lockmode, Tid, NOWAIT)

其中:s _ jd 是封锁对象的内部标识

locktype 是封锁类型

lockmode 是封锁模式

Tid 是事务对象标识号

NOWAIT:当为"N"时等待,为"Y"时不等待

(2) UNLOCK 算法。一般调用格式:UNLOCK(Tid)

(3)死锁检测算法。在 ORACLE 中提供了有效的死锁检测机制。在检测死锁中存在两个问题:一是死锁判定,即等待回路的判定;二是选择退出事务。选择退出事务的原则是执行修改数目最小者,即以记录为单位进行增、删和改记录个数最少的事务被选为退出事务。下面给出死锁处理的基本算法:

·TD.1—当新的等待锁到来时,即把 Tid 放入集合 S 中,然后取 WAIT _ POINT 赋于 P;

·TD.2—判断 P.Tid∈S 吗? 若是,则进行调用回收资源,并唤醒相应事务,然后返回,否则进入 TD.3;

·TD.3—在等待队列中寻找 P.Tid 所拥有的等待块,若有,则把 P.Tid 并入集合 S 中,同时将 WAIT _ POINT 赋于 P,然后转到 TD.2 继续循环;否则,不发生死锁,返回。

从以上锁的分类以及 LOCK 与 UNLOCK 的算法中,可以看出 ORACLE 核心实现的封锁管理程序与死锁诊断是简便灵活及有效的。

## 3. SYIS 的并发控制技术

SYIS 在并发控制上提供了三种方法:事务控制、被动并发性、记录锁定。

(1)事务控制。SYIS 为事务控制提供了 3 种调用方法:开始事务、中止事务、结束事务。开始事务用以标志一组逻辑上有关的 SYIS 的开始;结束事务用来完成一个事务并对受影响的数据文件做适当的修改;中止事务用来删除自一个活动的事务开始以来执行的所有操作并终止这个事务。任何时候当应用访问事务内的基表时,ORACLE 就为该应用锁定该基表,直到事务结束或异常



© 中国科学院软件研究所 http://www.c-s-a.org.cn

终止，而其他用户可以在非事务内和非锁定状态下读该基表。

一般情况下，当开始事务时，用户可以指定为等待锁定或非等待锁定。当指定为等待锁定时，如果有两台终端试图访问同一个事务内的文件，ORACLE 将锁定第二台终端，直到第一个用户结束或中止事务，若指定为非等待锁，则第二个用户可以检测到死锁的信息，这时系统管理员作出相应的处理。由于事务操作是对基表级锁定，因此很容易产生死锁。

(2)被动并发性。被动并发性控制方法允许读取与更新记录而不用执行任何锁定或事务，如果一个记录在用户的程序读取的时间或试图更新的时间之间有变动，那么返回错误信息。

(3)记录锁定。当 SYIS 的用户访问一个基表内的记录时，用户可以锁定希望封锁的记录。一旦被用户指定封锁的记录，其他用户可以不带锁定地读取记录，但是，没有任何用户可以锁定、更新或删除该记录，除非保持封锁的原用户释放它。

记录锁定有两种不同的记录封锁：单个和多个锁定。对这两类记录可指定为等待锁定或非等待锁定。具体办法是把一个值与任一“取”、“异步”、“打开”或者“开始事务”的操作码相加，列出下表的锁定偏置值：

| 值 | 锁定类型 |
|---|---|
| + 100 | 单个记录等待锁定 |
| + 200 | 单个记录非等待锁定 |
| + 300 | 多个记录等待锁定 |
| + 400 | 多个记录非等待锁定 |

对于单个记录锁定，在某一时刻只能锁定基表中的一个记录，若对这个基表发出带有一个锁定的另一个“取”的操作、更新或删除这个被锁定的记录，或者发出一个“解锁”命令，于是 ORACLE 就解放这个锁定。

对于这个多个记录锁定，允许一个应用在一个基表中锁定多个记录，并随后在必要时对那些记录进行更新或删除，这与单个记录不同的是更新记录或发出另一个带有多记录锁定的“取”操作时，不解锁该记录，只有通过显式“解锁”操作，这个锁才被解除。

这种等待解锁不返回状态码，直到该锁定成功，而非等待锁定，当另一个用户保持记录锁定时，立即给用户返回一个忙的状态。

由于 DML 语句会自动获得所需要的锁，因此，在应用程序中，可以直接使用 DML 语句，而不必考虑数据库锁，但是，由于数据库锁是自动获得的，因此用户无法对其进行判断，尤其是在有长的大事务或者对基表 DML 操作较多的应用中，容易出现长时间等待的情况，如图所示：

| 事务 1 | 时间 | 事务 2 |
|---|---|---|
| UPDATE scott.dept<br>SET loc = 'beiling'<br>WHERE deptno = 10;<br>1 row processed. | 1 | |
| | 2 | UPDATE scott.dept<br>SET loc = 'shanghai'<br>WHERE deptno = 10; |
| Rollback;<br>Rollback complete. | 3 | |
| | 4 | 1 rowprocessed. |

事务 2 在时间 2 对 EDPT 表进行更新，但由于事务 1 对基表 DEPT 有锁，故须等待，直到时间 3 时事务 1 回退或提交，事务 2 的语句才能执行，如果时间 1 与时间 3 之间间隔较长，则事务 2 将出现较长的等待时间。

为了避免自动使用数据库锁造成的事务长时间等待其他事务释放锁的情况，可以采用手工获得数据库锁的办法，手工获得数据库锁的方法是在对数据更新操作(INSERT、UPDATE、DELETE)之前，先利用获得锁的 DML 语句来检查所要操作的基表是否有数据锁，再决定是否进行进一步的数据更新操作。如图 2 所示：(见下页)

由于在时间 1 时事务 1 对基表 DEPT 加锁，因此，在时间 2 时事务 1 手工获得锁失败(产生 ORACLE 例外 ORA - 00054)，表明对基表 DEPT 的 deptno = 10 的记录的更新操作与其他事务冲突，而当时间 3 事务 1 进行回退或提交后，事务 2 就可以实现对 DEPT 基表的手工加锁(时间 4)。这样，便可以根据手工获得锁的情况在决定下一步的操作而不必一直等待其他事务释放所需要的锁。

**4. 两阶段非等待锁定机制技术及实现方法**

在 SYIS 中，称交易为事务，现在将有关概念术语列述如下：

事务：在批处理或远程批处理中的一个作业或作业步；或一个工作站与完成一个特殊动作或结果的其他设



© 中国科学院软件研究所 http://www.c-s-a.org.cn

备间的一次数据交换;或对数据库文件的一组更改。

| 事务 1 | 时间 | 事务 2 |
|---|---|---|
| UPDATE scott.dept<br>SET loc = 'beijing'<br>WHERE deptno = 10;<br>1 row processed. | 1 | |
| | 2 | SELECT loc<br>From scott.dept<br>Where deptno = 10<br>For update of loc<br>NOWAIT<br>ERROR:ORA - 00054:<br>resource<br>Busy and acquire with-<br>NOWAIT<br>Specified<br>No row selected. |
| ROLLBACK;<br>Rollbace complete. | 3 | |
| | 4 | SELECT loc<br>From scolt.dept<br>WHERE deptno = 10<br>For update of loc<br>NOWAIT<br>Loc<br>- - - - - - - -<br>BOSTON |

图 2

事务粒度:一次事务中所占有的共享资源量的多少。

事务回滚:事务失败时,要恢复事务所执行的所有动作到事务初始状态,就象事务没有发生过一样。

事务封锁:为执行事务中的操作,对需要的共享资源进行的封锁。

事务提交:事务成功时,使操作及结果数据能有效地操作。

在 ORACLE 数据库中,数据的一致性、完整性、可恢复性和并发性,使事务成为数据库系统的一个基本的工作单位,具体说是对访问的不可再分性的操作系列。具有以下四个基本性质:

(1)事务的不可再分性;

(2)事务的持久性;

(3)事务的串行性;

(4)事务的隔离性;

二阶段非等待封锁机制的实现是在 SYIS 提供的事务和封锁调用基础上,在应用程序级解决了可能出现死锁的问题,无论数据库系统是否提供了死锁检测和保护功能,此机制都能很好地处理各进程之间的事务并发机制。

定义:对所有事务都满足下述两条件的协议成为二阶段封锁协议:

(1)在对任何数据单元执行操作之前,事务首先对该数据单元加锁。

(2)若对一个数据单元开锁后,事务不再对任何其他单元进行加锁。

定理:满足二阶段封锁协议的事务的并发执行是可串行化的,即保证了事务的串行性。

二阶段非等待锁定机制按以下步骤实现:

LN.0 初值 V.delay = 0;

LN.1 延时 2 * V.delay;

LN.2 非等待开始事务;

LN.3 读所有需要修改记录;

LN.4 如果不成功,则中止事务,V.delay + +,返回 LN.1 继续循环;

LN.5 修改记录;

LN.6 如果失败,则中止事务,V.delay + +,返回 LN.1 循环;

LN.7 否则事务结束。

其中引入 V.delay + + 是为了减少立即循环导致的重复冲突的概率。

到现在可以看出,以上步骤是符合二阶段封锁协议的,同时因为采用了非等待读记录,则应用必然能得到 SYIS 需要的状态,从而决定了事务的继续或中止,因此不可能发生死锁,这是一种基于释放最近冲突进程的共享资源解决死锁的策略,不同于 ORACLE 的死锁的有向图检测及释放占用最少资源的事务的策略。

## 5.结束语

数据库锁是提高数据库并发性的有效手段,在 SYIS 应用中合理地使用数据库锁,可以极大地提高数据库并发性,减少不必要的等待时间,提高效率,并可以获得不同的数据读一致性。因此,在应用系统开发中,应充分重视数据库锁的,合理利用数据库锁。

## 参考文献

[1] 《ORACLE7 使用技巧》北京:科学出版社

[2] 《计算机工程与应用》1998.8 华北计算机出版社

[3] 《铁路计算机系统》1998.7 铁道部电子计算中心

(来稿时间:1998 年 11 月)



© 中国科学院软件研究所 http://www.c-s-a.org.cn