

# Oracle 数据库死锁问题研究

姜文<sup>1</sup>, 刘立康<sup>2</sup>

(1. 文思海辉技术有限公司西安分公司, 陕西 西安 710075;  
2. 西安电子科技大学通信工程学院, 陕西 西安 710071)

**摘要:**数据库是网络环境下多用户使用的共享资源,数据库在处理多线程大量数据存取过程中很可能出现死锁现象。文中介绍了 Oracle 数据库锁机制和死锁发生的原因,Oracle 数据库检测死锁采用的相关视图。详细叙述了 SQL 语句检测死锁、死锁的定位方法和解决数据库死锁的方法。编写了在 Linux 环境中死锁检测脚本,模拟 Oracle 数据库死锁测试了脚本的正确性。实际应用表明该方法可以有效监测程序中的主要模块是否发生死锁,而且简单有效。

**关键词:**Oracle 数据库;数据库锁;死锁

**中图分类号:**TP311

**文献标识码:**A

**文章编号:**1673-629X(2014)03-0097-05

**doi:**10.3969/j.issn.1673-629X.2014.03.035

## Research on Deadlock of Oracle Database

JIANG Wen<sup>1</sup>, LIU Li-kang<sup>2</sup>

(1. The Pactera Technology Co., Ltd. Xi'an Branch, Xi'an 710075, China;  
2. School of Telecommunication Engineering, Xidian University, Xi'an 710071, China)

**Abstract:** The database is a shared resource of multi-user in network environment, database deadlock phenomenon is likely to occur in multi-threaded process by large amounts of data access. Introduce the Oracle database locking mechanism and the causes the deadlock occurs, the Oracle database detects the related views deadlock uses. A detailed description is conducted for the SQL states the deadlock detection, the positioning method of deadlock and the resolution of the database deadlock. Compiled the deadlock detection script in the Linux environment, simulate the Oracle database deadlock to test the correctness of the script. Practical application shows that the script can be effective monitoring whether the program's main module is deadlock or not, with simplicity and effectiveness.

**Key words:** Oracle database; database locks; deadlock

## 0 引言

大型数据库应用系统的并发量是很大的,可能有数十万甚至数百万用户同时访问数据库。这就会导致数据库引擎同时处理大量的锁定请求,如果设计不合理,则有可能频繁产生死锁,从而严重影响系统的性能。但是按照一定的程序设计规范来设计数据库应用系统,完全可以尽量减少死锁情况的发生,从而提高数据库系统的性能。这表明死锁问题是数据库领域一个重要的内容。

Oracle 数据库的死锁<sup>[1-7]</sup>问题大多是由于程序设计不准确所导致。一旦发现是程序设计的问题所导致的死锁,经过对设计方案调整之后,基本上可避免死锁的发生。通常 Oracle 数据库发生死锁时是可以自动

检测死锁的,但是分析警告日志文件和跟踪文件非常耗时,通常利用 Oracle 数据库提供的相关视图,采用 SQL 语句检测死锁。文中详细叙述了 SQL 语句检测死锁、死锁的定位方法和解决数据库死锁的方法。

## 1 数据库锁机制

由于 Oracle 数据库<sup>[8-12]</sup>是网络环境下允许多个用户同时对数据库进行操作的数据库系统,当多个用户同时进行数据存取时,就会在 Oracle 数据库中发生多个事务同时对同一数据进行操作的情况,这种情况可以称为并发操作。若 Oracle 数据库已经出现了并发操作这一现象,而对该现象不加以控制就可能造成用户所读取和存储的数据不正确,导致数据库的一致

收稿日期:2013-05-19

修回日期:2013-08-27

网络出版时间:2014-01-08

基金项目:国家部委基础科研计划;国防预算基金项目(D1120060967)

作者简介:姜文(1986-),女,陕西西安人,硕士研究生,CCF 会员,研究方向为图像处理与分析、文字信息分析处理;刘立康,副教授,研究方向为数字通信、图像传输与处理、图像分析与图像识别等。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20140108.0818.018.html>

性被破坏。

为了保持 Oracle 数据库的数据一致性,使用加锁机制是一种很好的解决方法。另外加锁机制也能够保持多个用户同时对 Oracle 数据库进行操作时,每一个用户的操作相对独立。加锁机制具体操作时,当用户对 Oracle 数据库中某个数据对象进行操作之前,需要先向系统发出请求,对相应的事务进行加锁。其他事务若要对这个事务进行更新操作,需要等到此事务完成解锁操作之后才能进行。完成了事务加锁后,该事务对相应的数据对象就有了相应的控制机制,能够较好地保持数据一致性以及用户操作的独立性。

2 关于各种锁的介绍

根据所保护对象的不同,Oracle 数据库锁<sup>[10,13-15]</sup>可以分为以下两类:

一类是 DML 锁(data locks,数据锁),用于保护数据的完整性。在 Oracle 数据库中,DML 锁又分为 TM 锁和 TX 锁,其中 TM 锁被称为表级锁,TX 锁被称为事务锁或行级锁。当 Oracle 系统中执行 DML 语句时,系统会自动在需要操作的表上申请 TM 锁。获得 TM 锁后,系统接着自动申请 TX 锁,并对锁定的数据行进行锁标志位置位。因此,事务加锁前就不必逐行检查 TX 锁相容性,而只需检查 TM 锁模式的相容性即可,大大提高了系统的效率。TM 锁包括的模式在数据库用 0~6 来表示,不同的 SQL 操作产生不同类型的 TM 锁,如表 1 所示。

表 1 Oracle 的 TM 锁模式

锁模式	锁描述	解释
0	NONE	
1	NULL	空
2	SS( Row-S)	行级共享锁,其他对象只能查询这些数据行
3	SX( Row-X)	行级排他锁,在提交前不允许做 DML 操作
4	S( Share)	共享锁
5	SSX( S/Row-X)	共享行级排他锁
6	X( Exclusive)	排他锁

X 锁(排他锁)只有在数据行上才有。在 Oracle 数据库中,当一个事务首次发起一个 DML 语句时就获得一个 TX 锁,该锁保持到事务被提交或回滚。

另一类是 DDL 锁(dictionary locks,字典锁),用于保护数据库对象的结构,如表、索引等的结构定义;内部锁和闩(internal locks and latches),保护数据库的内部结构。

3 死锁的原因以及解决方法

死锁是一种可能发生在任何多线程系统中的状

态。数据库系统中,当多个事务分别锁定了某资源,并又试图请求锁定其他事务已经锁定的资源,每个事务都等待其他事务释放自己需要的锁,这样就形成了一个锁定请求环,从而产生死锁。

3.1 死锁现象产生的原因

Oracle 数据库产生死锁的根本原因是,当用户之间希望持有对方的资源时就会发生死锁。具体表现有以下几种:

1)用户 A 访问表 A(对表 A 加锁),接着又访问表 B;而用户 B 访问表 B(对表 B 加锁),之后又企图访问表 A;这时用户 A 由于用户 B 已经先锁住表 B,必须等待用户 B 释放表 B 才能访问表 B,而用户 B 要等用户 A 释放表 A 才能访问表 A,因此产生了死锁现象。

2)用户 A 查询系统中某表中的一条记录,然后修改该条记录;而用户 B 同时修改该条记录,这时用户 A 事务的锁级别由共享锁准备上升为独占锁,而用户 B 事务的独占锁由于用户 A 事务有共享锁存在必须等 A 释放掉共享锁;而用户 A 由于用户 B 的独占锁而无法上升到独占锁因此无法释放共享锁,此时也出现了死锁现象。

3)存在执行多个不满足条件的 update 语句时,执行了全表扫描,把行级锁上升为表级锁的事务时,这样的操作很容易造成死锁现象。

4)当数据表中的数据量非常庞大而索引建的过少或不合适的时候,此时若经常进行全表扫描,最终也会造成死锁现象。

3.2 死锁现象的解决方法

1)数据库死锁的现象。

程序在执行的过程中,点击确定或保存按钮,程序没有响应,也没有出现报错。

2)死锁的检测。

(1) Oracle 自动检测死锁。

Oracle 自动检测死锁并解决它们(通过回滚一个包含在死锁中的语句实现),释放掉该语句锁住的数据,回滚的会话将会遇到 Oracle 错误“ORA-00060:等待资源时检测到死锁”。

当 Oracle 数据库检测到死锁时,相应的消息就写入到警告日志文件(alert.log)中,也会在 USER\_DUMP\_DEST 目录下创建一个详细描述死锁相关信息的跟踪文件。

在日常工作中,定位 Oracle 数据库死锁问题时,时常需要通过查看警告日志文件(alert.log)来完成定位。定位时,如果在告警日志文件(alert.log)中查看到“ORA-00060”的相关错误信息,则表明 Oracle 数据库有死锁现象产生。这时还需要查看 USER\_DUMP\_DEST 目录下对应的跟踪文件,由跟踪文件中所记录的

信息进一步定位分析产生死锁的原因。分析警告日志文件和跟踪文件是非常耗时的,由于数据繁杂并不容易得到有用的相关信息。通常采用 SQL 语句检测死锁。

(2) 采用 SQL 语句检测死锁。

①SQL 语句检测死锁的相关视图<sup>[10]</sup>。

Oracle 数据库采用 SQL 语句检测死锁需要五个常用的相关视图。

V \$ SESSION 视图:V \$ SESSION 是基础信息视图,查询会话的信息和锁的信息。

V \$ PROCESS 视图:通过查询 V \$ PROCESS 视图,可以找到对应操作系统的每个进程的信息。V \$ PROCESS 视图包含当前系统 Oracle 运行的所有进程信息。常被用于将 Oracle 或服务进程的操作系统进程 ID 与数据库 session 之间建立联系。

V \$ LOCKED\_OBJECT 视图:显示被加锁的数据库对象,通过与 DBA\_OBJECT 进行连接查询,可以显示具体的对象名及执行加锁操作的 Oracle 用户名。只包含 DML 的锁信息,包括回滚段和会话信息。

DBA\_OBJECTS 视图:列出数据库中所有的对象,是常用的数据视图,可以获得数据库中任意的对象信息。

V \$ LOCK 视图:用于显示锁的信息,通过与 V \$ SESSION 进行连接查询,可以显示占有锁的会话,以及等待锁的会话信息。

②SQL 语句检测死锁。

可以通过以下语句来检测死锁

SELECT

a. SPID, d. SESSION\_ID, c. TYPE, c. LMODE, c. BLOCK, d. PROCESS, d. OBJECT\_ID, e. OBJECT\_NAME, e. OBJECT\_TYPE FROM v \$ process a, v \$ session b, v \$ locked\_object d, dba\_objects e, v \$ lock c WHERE a. ADDR=b. PADDR and b. SID=c. SID and b. SID=d. SESSION\_ID and d. OBJECT\_ID=e. OBJECT\_ID and (c. BLOCK=1 or c. REQUEST=6);

其中,查询语句中的字段分别表示:

b. SID:SESSION 标识,常用于连接其他列。

a. SPID:操作系统进程 ID。

d. SESSION\_ID:持有锁的 session 标识。

d. PROCESS:操作系统进程号。

d. OBJECT\_ID:被锁对象 ID。

e. OBJECT\_NAME:对象名称。

e. OBJECT\_TYPE:对象类型。

e. OBJECT\_ID:对象 ID。

c. TYPE:加锁的类型。包括 TM 锁、TX 锁等。

c. LMODE:会话等待的锁模式信息。锁模式信息

通常用数字 0~6 表示。

c. BLOCK:该锁是否阻塞其他锁的申请。

该语句可以查询到死锁的操作系统进程 ID,锁的 SESSION\_ID,锁的类型,会话等待的锁模式信息,该锁是否阻塞其他锁的申请,操作系统进程号,被锁对象 ID、被锁对象名称和被锁对象类型。通常关注比较多的是锁的 SESSION\_ID 和操作系统进程号。

如果仅查询数据库死锁的 SESSION\_ID 和操作系统进程号,查询语句可以简化如下:

SELECT

d. SESSION\_ID, d. PROCESS FROM v \$ process a, v \$ session b, v \$ locked\_object d, dba\_objects e, v \$ lock c WHERE a. ADDR=b. PADDR and b. SID=c. SID and b. SID=d. SESSION\_ID and d. OBJECT\_ID=e. OBJECT\_ID and (c. BLOCK=1 or c. REQUEST=6);

3) 死锁的定位方法。

通过检查数据库表,能够检查出是哪一条语句被死锁,产生死锁的机器是哪一台。

(1) 用 dba 用户执行以下语句。

select username, lockwait, status, machine, program from v \$ session where sid in (select session\_id from v \$ locked\_object)

如果有输出的结果,则说明有死锁,且能看到死锁的机器是哪一台。字段说明:

username:死锁语句所用的数据库用户;

lockwait:死锁的状态,如果有内容表示被死锁;

status:状态,active 表示被死锁;

machine:死锁语句所在的机器;

program:产生死锁的语句主要来自哪个应用程序。

(2) 用 dba 用户执行以下语句,可以查看到被死锁的语句。

select sql\_text from v \$ sql where hash\_value in (select sql\_hash\_value from v \$ session where sid in (select session\_id from v \$ locked\_object))

4) 解决数据库死锁的方法。

解决数据库死锁主要有两类方法<sup>[8-15]</sup>,一类是避免死锁的发生,保证系统不进入死锁状态;另一类是允许发生死锁,定期检测系统中有无死锁,若有则解除。

(1) 规避死锁。

Oracle 数据库死锁问题由于程序设计不准确所导致的情况比较常见。一旦发现是由于程序设计的问题所导致的死锁,经过对设计方案的调整之后,基本上都可以规避死锁现象。检测到死锁后,应查明死锁的原因,定位死锁在程序中的位置。将程序关闭后修改程序设计,从而尽量规避死锁发生。这是一种长远的处

理方法。

## (2)解除死锁。

为了应对当前的死锁状态,解除死锁,可根据具体情况采取以下处理方法:

①提交(COMMIT)或者回滚(ROLLBACK)某个事务;

②通过 DBA 撤销事务来解锁;

③通过 DBA 在线/离线操作解锁或者加锁事务;

④DBA 用 ALETER DBATABSE KILL SESSION ‘SID,SERIAL#’来杀掉该进程(SID,SERIAL#均为 v \$ session 的字段,由于 SID 有可能会重复,当两个 session 的 SID 重复时,SERIAL#用来区别 session);

⑤系统管理员用 KILL - 9 PROCESS 来杀掉死锁进程,. PROCESS 为操作系统进程号。

这些解除方法虽然能够解决问题,但是通常在大型应用系统这样做有一定的风险。因此,应该尽量从应用程序设计方面来调优,减少出现死锁这种情况。

### 3.3 Oracle 数据库死锁检测脚本的测试

文中测试环境为:Suse Linux 10 x86/ Suse Linux 10 x86-64,Oracle 11g 由于 Oracle 数据库死锁并不是每时每刻都存在的现象,需要模拟 Oracle 数据库死锁的现象来测试死锁检测脚本的正确性,测试过程如下所示:

1)模拟 Oracle 数据库死锁的现象。

1. 建表用于存放演示数据:

```
CREATE TABLE DeadLockTest
```

```
(
```

```
Id NUMBER,
```

```
Name VARCHAR2(50)
```

```
);
```

2. 向表中插入数据,Id 等于 1 的记为记录 1,Id 等于 2 的记为记录 2:

```
INSERT INTO DeadLockTest VALUES(1,'Test1');
```

```
INSERT INTO DeadLockTest VALUES ( 2,'Test2');
```

```
COMMIT;
```

3. 分别打开 2 个数据库操作窗口,将这两个操作窗口分别记为“窗口 A”与“窗口 B”。并将在“窗口 A”中进行操作的事物记为“事务 A”;在“窗口 B”中进行操作的事物记为“事务 B”。

4. 在窗口 A 执行 UPDATE 语句,更新表 DeadLockTest 中 Id 等于 1 的记录(相当于给记录 1 上申请了 X 锁):

```
UPDATE DeadLockTest SET Name = 'Test1'
WHERE Id=1;
```

5. 在窗口 B 执行 UPDATE 语句,更新表 DeadLock-

Test 中 Id 等于 2 的记录(相当于给记录 2 上申请了 X 锁):

```
UPDATE DeadLockTest SET Name = 'Test2'
WHERE Id=2;
```

6. 切换到窗口 A 执行 UPDATE 语句,更新表 DeadLockTest 中 Id 等于 2 的记录(相当于给记录 2 上申请了 X 锁),而这个 X 锁被事务 B 所占用。只有事务 B 被提交或回滚之后,事务 A 才能得到记录 2 上的 X 锁:

```
UPDATE DeadLockTest SET Name = 'Test2'
WHERE Id=2;
```

7. 切换到窗口 B 执行 UPDATE 语句,更新表 DeadLockTest 中 Id 等于 1 的记录(相当于给记录 1 上申请了 X 锁),而这个 X 锁被事务 A 所占用。只有事务 A 被提交或回滚之后,事务 B 才能得到记录 1 上的 X 锁:

```
UPDATE DeadLockTest SET Name = 'Test1'
WHERE Id=1;
```

经过以上操作,事务 A 占有记录 1 上的 X 锁,正在申请记录 2 上的 X 锁;事务 B 占有记录 2 上的 X 锁,正在申请记录 1 上的 X 锁。双方互相占有对方需要的资源,同时又申请对方占有的资源,因此产生死锁现象。

2)死锁检测脚本的测试。

模拟出死锁现象之后,在 Linux 环境下执行死锁检测脚本( deadlockdetection. sh)来查找发生死锁的进程,死锁检测脚本( deadlockdetection. sh)的编写见 3.4,执行步骤为:

1. chmod +x deadlockdetection. sh;

2. ./ deadlockdetection. sh;

3. 切换到/tmp/sqldeadlockdetection. txt,取出日志文件与屏幕上的显示结果对比,查看结果是否相同。

测试结果表明,该脚本能够准确查找到出现死锁的进程。

### 3.4 在 Linux 环境中的应用

1)编写在 Linux 环境中死锁检测脚本( deadlock-detection. sh)。

(1)输入用户名,生成保存检测死锁信息的文件,以 dba 用户身份进入 sqlplus。

```
UESR = 'abc'
```

```
tmplogfile = "/tmp/sqldeadlockdetection. txt"
```

```
sqlplus -S/ as sysdba >> $ tmplogfile << EOF //
定向到输出文本
```

(2)定义输出文本中的字段排列格式。

```
set sqlblankline on //处理空行
```

```
set linesize 500 //设置行高
```

```
set pagesize 40 //设置每页显示条数
col SPID format a10
col SPID wrapped //设置列的回绕方式,为了使显示结果整齐
.....
```

(3)采用 SQL 语句检测死锁信息(见 3.2),并将检测结果写入输出文件(/tmp/sqldeadlockdetection.txt)。

(4)退出 sqlplus。

(5)将一些需要检查的模块进程号写入输出文件(/tmp/sqldeadlockdetection.txt)。同时将输出文件内容显示到屏幕上。

```
echo "模块_pid=`ps aux | grep 模块 | grep -v `
grep` | awk -F""{ print $2 }"" >> $ tmplogfile
cat $ tmplogfile
2)在 Linux 环境中的应用。
```

在环境中运行该脚本可以检测系统是否发生死锁。若发生死锁,通过比对进程号可以查明是哪个模块发生了死锁,从而及时采取方法处理。

## 4 结束语

死锁是大型数据库在处理多线程大量数据存取过程中很可能出现的现象,将造成系统不稳定甚至系统崩溃等不良后果。死锁只可能减少,但无法根本避免。实践经验表明,采取文中叙述的原则和方法能有效监测程序中的主要模块是否发生死锁,而且简单可靠有效。

### 参考文献:

[1] Gligor V D,Shattuc S H. On deadlock detection in distributed systems[J]. IEEE transactions on software engineering,1980,

(上接第 96 页)

[4] 李瑞芳,李仁发,罗 娟. 无线多媒体传感器网络 MAC 协议研究综述[J]. 通信学报,2008,29(8):111-123.

[5] 李 蓬. 支持 QoS 的无线视频传感器网络传输模型的研究[J]. 传感器与微系统,2011,30(3):7-9.

[6] 孙 岩,马华东. 无线多媒体传感器网络 QoS 保障问题[J]. 电子学报,2008,36(7):1412-1420.

[7] Durmaz I O,Jansen P G,Mullender S J. MC-LMAC: A multi-channel MAC protocol for wireless sensor networks[R]. Enschede, The Netherlands: University of Twente, 2008.

[8] Yahya B, Ben-Othman J. Towards a classification of energy aware MAC protocols for wireless sensor networks[J]. Journal of wireless communication and mobile computing, 2009, 9(12):1572-1607.

6(5):435-440.

[2] Obermarck R. Distributed deadlock detection algorithm[J]. ACM transactions on database systems, 1982, 7(2):187-208.

[3] Chandy K M, Mism J, Hass L M. Distributed deadlock detection[J]. ACM transactions on computer systems, 1983, 1(2):144-156.

[4] Sinha M K, Natarajan N. A priority based distributed deadlock detection algorithm[J]. IEEE transactions on software engineering, 1985, 11(1):67-80.

[5] Choudhary A N, Kohler W H, Stankovic J A, et al. A modified priority algorithm for distributed deadlock detection and resolution[J]. IEEE transactions on software engineering, 1989, 15(1):10-17.

[6] Knapp E. Deadlock detection in distributed database systems[J]. ACM computing surveys, 1987, 19(4):303-327.

[7] 郝朝辉, 麦中凡. 面向对象数据库死锁检测方法研究[J]. 软件学报, 1996, 7(12):722-727.

[8] 李晓黎, 陈艳莲, 张如昌. Oracle 学习笔记: 日常应用、深入管理、性能优化[M]. 北京: 人民邮电出版社, 2010.

[9] 龙华飞, 唐月华, 邓 千, 等. “军卫一号”系统中“死锁”的分析及处理方法[J]. 中国数字医学, 2011, 6(1):31-32.

[10] 李雁敏. 并发访问 ORACLE 数据库的数据死锁分析和解决措施[J]. 内蒙古科技与经济, 2012(13):79-80.

[11] 陈 鹏. 分布式数据库死锁检测算法研究[D]. 重庆: 重庆大学, 2004.

[12] 才科扎西. 数据库设计中封锁和并行控制技术[J]. 苏州科技学院学报(工程技术版), 2009, 22(2):69-73.

[13] 薄 宏. 基于 SQL Server 锁定数据技术的探讨[J]. 计算机与数字工程, 2006, 34(5):69-70.

[14] 曹玉林, 胡 飞, 崔 键. Oracle 封锁技术在试飞系统设计中的应用[J]. 微处理机, 2006, 27(3):67-71.

[15] 雷 亮, 杨国权. ORACLE 数据库系统加锁问题的研究[J]. 承钢技术, 2005(1):60-62.

[9] Ye W, Heidemann J, Estrin D. Medium access control with coordinated adaptive sleeping for wireless sensor networks[J]. IEEE/ACM transactions on networking, 2004, 12(3):493-506.

[10] Incel O D, Krishnamachari B. Enhancing the data collection rate of tree-based aggregation in wireless sensor networks[C]//Proceedings of IEEE communications society conference on sensor, mesh and Ad Hoc communications and networks. Salt Lake City, UT, USA: [s. n.], 2008:569-577.

[11] 李建中, 李金宝, 石胜飞. 传感器网络及其数据管理的概念、问题与进展[J]. 软件学报, 2003, 14(10):1717-1727.

[12] 陆海明, 刘学军, 钱江波. 基于有线长链解决传感器网络的能量空洞[J]. 计算机研究与发展, 2010, 47(z2):1-4.

作者：[姜文](#)，[刘立康](#)，[JIANG Wen](#)，[LIU Li-kang](#)  
作者单位：[姜文, JIANG Wen\(文思海辉技术有限公司西安分公司, 陕西 西安, 710075\)](#)，[刘立康, LIU Li-kang\(西安电子科技大学 通信工程学院, 陕西 西安, 710071\)](#)  
刊名：[计算机技术与发展](#)

---

英文刊名：[Computer Technology and Development](#)

---

年，卷(期)：2014(3)

本文链接：[http://d.wanfangdata.com.cn/Periodical\\_wjfz201403025.aspx](http://d.wanfangdata.com.cn/Periodical_wjfz201403025.aspx)