DB2和 Oracle的并发控制（锁）比较

在实际的生产运行环境中，笔者在国内很多客户现场都看到开发人员和系统管理人员遇到很多有关于锁而引起的性能问题，进而被多次问起DB2和Oracle中锁的区别比较问题，笔者根据自己在工作中对DB2和Oracle数据库的使用经验积累写下这篇文章。

牛新庄博士是IBM官方高级培训讲师，于2002年获IBM杰出软件专家奖，是《程序员》，《电脑编程与维护》等杂志数据库专栏作家，是很多公司的技术顾问，他拥有OCP，AIX，DB2，HP-UX，MQ，CICS和WebSphere等二十多项国际认证。曾经帮助工农商建招交六大行、上海移动、青岛海尔、云南红塔、江苏电力公司等公司做过问题诊断、性能调优和技术支持，他经常往返于国内大中城市解决数据库技术难题，有着丰富的理论和实践经验。

2005 年 12 月 26 日

1 引言

在关系数据库（DB2，Oracle，Sybase，Informix和SQL Server）最小的恢复和交易单位为一个事务（Transactions）,事务具有ACID(原子性，一致性，隔离性和永久性)特征。关系数据库为了确保并发用户在存取同一数据库对象时的正确性（即无丢失更新、可重复读、不读"脏"数据，无"幻像"读），数据库中引入了并发（锁）机制。基本的锁类型有两种：排它锁（Exclusive locks记为X锁）和共享锁（Share locks记为S锁）。

排它锁：若事务T对数据D加X锁，则其它任何事务都不能再对D加任何类型的锁，直至T释放D上的X锁；一般要求在修改数据前要向该数据加排它锁，所以排它锁又称为写锁。

共享锁：若事务T对数据D加S锁，则其它事务只能对D加S锁，而不能加X锁，直至T释放D上的S锁；一般要求在读取数据前要向该数据加共享锁，所以共享锁又称为读锁。

2 DB2 多粒度封锁机制介绍

2.1 锁的对象

DB2支持对表空间、表、行和索引加锁（大型机上的数据库还可以支持对数据页加锁）来保证数据库的并发完整性。不过在考虑用户应用程序的并发性的问题上，通常并不检查用于表空间和索引的锁。该类问题分析的焦点在于表锁和行锁。

2.2 锁的策略

DB2可以只对表进行加锁，也可以对表和表中的行进行加锁。如果只对表进行加锁，则表中所有的行都受到同等程度的影响。如果加锁的范围针对于表及下属的行，则在对表加锁后，相应的数据行上还要加锁。究竟应用程序是对表加行锁还是同时加表锁和行锁，是由应用程序执行的命令和系统的隔离级别确定。

2.2.1 DB2表锁的模式

DB2在表一级加锁可以使用以下加锁方式：

表一：DB2数据库表锁的模式

名称缩写	全名	描述
IN	无意图锁（Intenet None）不需要行锁	该锁的拥有者可以读表中的任何数据，包括其他事务尚未提交的数据，但不能对表中的数据进行更改。
IS	意图共享锁（Intent Share）需要行锁配合	该锁的拥有者在拥有相应行的上的 S 锁时可以读取该行的数据。但不能对表中的数据进行更改。
IX	意图排它锁(Intent eXclusive)需要行锁配合	该锁的拥有者在拥有相应行的 X 锁时可以更改该行的数据。
SIX	共享携意图排它锁(Share with Intent exclusive）需要行锁配合	锁的拥有者可以读表中的任何数据，如果在相应的行上能够获得 X 锁，则可以修改该行。SIX 锁的获得比较特殊，它是在应用程序已经拥有 IX 锁的情况下请求 S 锁或者是在应用程序已经拥有 S 锁的情况下请求 IX 锁时生成的。
S	共享锁（Share）不需要行锁配合	锁的拥有者可以读表中的任何数据，如果表上被加上 S 锁，该表中的数据就只能被读取，不能被改变。
U	更新锁（Update）不需要行锁配合	锁的拥有者可以读表中的任何数据，如果在升级到 X 锁之后，可以更改表中的任何数据。该锁是处于等待对数据进行更改的一种中间状态。
X	排它锁（eXclusive）不需要行锁配合	锁的拥有者可以读取或更改表中的任何数据。如果对表加上 X 锁，除了未提交读程序，其他应用程序都不能对该表进行存取。
Z	超级排它锁（Super Exclusive）不需要行锁配合	该锁不是通过应用程序中的 DML 语言来生成的。一般是通过对表进行删除（Drop）和转换(Alter)操作或创建和删除索引而获得的。如果对表上加上 Z 锁，其他应用程序，包括未提交读程序都不能对该表进行存取。

下面对几种表锁的模式进一步加以阐述：

IS、IX、SIX方式用于表一级并需要行锁配合，他们可以阻止其他应用程序对该表加上排它锁。

如果一个应用程序获得某表的IS锁，该应用程序可获得某一行上的S锁，用于只读操作，同时其他应用程序也可以读取该行，或是对表中的其他行进行更改。

如果一个应用程序获得某表的IX锁，该应用程序可获得某一行上的X锁，用于更改操作，同时其他应用程序可以读取或更改表中的其他行。

如果一个应用程序获得某表的SIX锁，该应用程序可以获得某一行上的X锁，用于更改操作，同时其他应用程序只能对表中其他行进行只读操作。

S、U、X和Z方式用于表一级，但并不需要行锁配合，是比较严格的表加锁策略。

如果一个应用程序得到某表的S锁。该应用程序可以读表中的任何数据。同时它允许其他应用程序获得该表上的只读请求锁。如果有应用程序需要更改读该表上的数据，必须等S锁被释放。

如果一个应用程序得到某表的U锁，该应用程序可以读表中的任何数据，并最终可以通过获得表上的X锁来得到对表中任何数据的修改权。其他应用程序只能读取该表中的数据。U锁与S锁的区别主要在于更改的意图上。U锁的设计主要是为了避免两个应用程序在拥有S锁的情况下同时申请X锁而造成死锁的。

如果一个应用程序得到某表上的X锁，该应用程序可以读或修改表中的任何数据。其他应用程序不能对该表进行读或者更改操作。

如果一个应用程序得到某表上的Z锁，该应用程序可以读或修改表中的任何数据。其他应用程序，包括未提交读程序都不能对该表进行读或者更改操作。

IN锁用于表上以允许未提交读这一概念。

2.2.2 DB2行锁的模式

除了表锁之外，DB2还支持以下几种方式的行锁。

表二：DB2数据库行锁的模式

名称缩写	全名	需要表锁的最低级别	描述
S	共享锁（Share）	IS	该行正在被某个应用程序读取，其他应用程序只能对该行进行读操作。
U	更改锁（Update）	IX	某个应用程序正在读该行并有可能更改该行，其他应用程序只能读该行。
X	排它锁（eXclusive）	IX	该行正在被某个应用程序更改，其他应用程序不能访问该行。
W	弱排它锁（Weak eXclusive）	IX	当一行数据被插入表中的时候，该行上会被加上 W 锁。锁的拥有者能够更改该行，该锁基本与 X 锁相同，除了它与 NW 锁兼容。
NS	下一键共享锁（Next Key Share）	IS	锁的拥有者和其他程序都可以读该行，但不能对该行进行更改。当应用程序处于 RS 或 CS 隔离级下，该锁用来替代 S 锁。
NX	下一键排它锁（Next Key eXclusive）	IX	当一行数据被插入到索引中或从索引中被删除的时候，该行的下一行上会被加上该锁。锁的拥有者可以读，但不能更改锁定行。该锁与 X 锁类似，只是与 NS 锁兼容。
NW	下一键弱排它锁（Next Key Weak eXclusive）	IX	当一行被插入到索引中的时候，该行的下一行会被加上该锁。锁的拥有者可以读但不能更改锁定行。该锁与 X 和 NX 锁类似，只是与 W 和 NS 锁兼容。

2.2.3 DB2锁的兼容性

表三：**DB2**数据库表锁的相容矩阵

锁 A 的模式	**锁 B 的模式**							
	IN	IS	S	IX	SIX	U	X	Z
IN	Y	Y	Y	Y	Y	Y	Y	N
IS	Y	Y	Y	Y	Y	Y	N	N
S	Y	Y	Y	N	N	Y	N	N
IX	Y	Y	N	Y	N	N	N	N
SIX	Y	Y	N	N	N	N	N	N
U	Y	Y	Y	N	N	N	N	N
X	Y	N	N	N	N	N	N	N
Z	N	N	N	N	N	N	N	N

表四：**DB2**数据库行锁的相容矩阵

锁A的模式	**锁 B 的模式**						
	S	U	X	W	NS	NX	NW
S	Y	Y	N	N	Y	N	N
U	Y	N	N	N	Y	N	N
X	N	N	N	N	N	N	N
W	N	N	N	N	N	N	Y
NS	Y	Y	N	N	Y	Y	Y
NX	N	N	N	N	Y	N	N
NW	N	N	N	Y	Y	N	N

下表是笔者总结了DB2中各SQL语句产生表锁的情况（假设缺省的隔离级别为CS）：

SQL 语句	行锁模式	表锁模式	允许的锁模式
Select * from table_name for read only with rr..	无	S	IN,IS,S,U
Select * from table_name for read only with rs..	NS	IS	IN,IS,S,SIX,U
Select * from table_name for read only with cs..	无	无	
Select * from table_name for read only with ur..	无	无	
Select * from table_name for update with rr	无	U	IN,IS,S
Select * from table_name for update with rs	U	IX	IN,IS,IX
Select * from table_name for update with cs	U	IX	IN,IS,IX
Select * from table_name for update with ur	U	IX	IN,IS,IX
Insert into table_name……	W	IX	IN,IS,IX
Update table_name……	X	IX	IN,IS,IX
Delete from table_name……	X	IX	IN,IS,IX
lock table table_name in share mode	无	S	IN,IS,S,U
lock table table_name in exclusive mode	无	X	IN
Alter table t1 add column id int		Z	
Drop table t1	X	Z	
Create table t1(id int)		Z	
注：alter，create，drop 会在 syscolumns,systables,systablespaces,sysuserath 等数据字典系统表中加行级锁。			

2.3 DB2锁的升级

每个锁在内存中都需要一定的内存空间，为了减少锁需要的内存开销，DB2提供了锁升级的功能。锁升级是通过对表加上非意图性的表锁，同时释放行锁来减少锁的数目，从而达到减少锁需要的内存开销的目的。锁升级是由数据库管理器自动完成的，有两个数据库的配置参数直接影响锁升级的处理：

locklist--在一个数据库全局内存中用于锁存储的内存。单位为页（4K）。

maxlocks--一个应用程序允许得到的锁占用的内存所占locklist大小的百分比。

锁升级会在这两种情况下被触发：

某个应用程序请求的锁所占用的内存空间超出了maxlocks与locklist的乘积大小。这时，数据库管理器将试图通过为提出锁请求的应用程序申请表锁，并释放行锁来节省空间。

在一个数据库中已被加上的全部锁所占的内存空间超出了locklist定义的大小。这时，数据库管理器也将试图通过为提出锁请求的应用程序申请表锁，并释放行锁来节省空间。

锁升级虽然会降低OLTP应用程序的并发性能，但是锁升级后会释放锁占有内存并增大可用的锁的内存空间。

锁升级是有可能会失败的，比如，现在一个应用程序已经在一个表上加有IX锁，表中的某些行上加有X锁，另一个应用程序又来请求表上的IS锁，以及很多行上的S锁，由于申请的锁数目过多引起锁的升级。数据库管理器试图为该应用程序申请表上的S锁来减少所需要的锁的数目，但S锁与表上原有的IX锁冲突，锁升级不能成功。

如果锁升级失败，引起锁升级的应用程序将接到一个-912的SQLCODE。在锁升级失败后，DBA应该考虑增加locklist的大小或者增大maxlocks的百分比。同时对编程人员来说可以在程序里对发生锁升级后程序回滚后重新提交事务（例如：if sqlca.sqlcode=-912 then rollback and retry等）。

3 Oracle 多粒度锁机制介绍

根据保护对象的不同，Oracle数据库锁可以分为以下几大类：

(1) DML lock（data locks，数据锁）：用于保护数据的完整性；

(2) DDL lock（dictionary locks，字典锁）：用于保护数据库对象的结构（例如表、视图、索引的结构定义）；

(3) Internal locks 和latches（内部锁与闩）：保护内部数据库结构；

(4) Distributed locks（分布式锁）：用于OPS（并行服务器）中；

(5) PCM locks（并行高速缓存管理锁）：用于OPS（并行服务器）中。

在Oracle中最主要的锁是DML（也可称为data locks，数据锁）锁。从封锁粒度（封锁对象的大小）的角度看，Oracle DML锁共有两个层次，即行级锁和表级锁。

3.1 Oracle的TX锁（行级锁、事务锁）

许多对Oracle不太了解的技术人员可能会以为每一个TX锁代表一条被封锁的数据行，其实不然。TX的本义是Transaction（事务），当一个事务第一次执行数据更改（Insert、Update、Delete）或使用SELECT… FOR UPDATE语句进行查询时，它即获得一个TX（事务）锁，直至该事务结束（执行COMMIT或ROLLBACK操作）时，该锁才被释放。所以，一个TX锁，可以对应多个被该事务锁定的数据行（在我们用的时候多是启动一个事务，然后SELECT… FOR UPDATE NOWAIT）。

在Oracle的每行数据上，都有一个标志位来表示该行数据是否被锁定。Oracle不像DB2那样，建立一个链表来维护每一行被加锁的数据，这样就大大减小了行级锁的维护开销，也在很大程度上避免了类似DB2使用行级锁时经常发生的锁数量不够而进行锁升级的情况。数据行上的锁标志一旦被置位，就表明该行数据被加X锁，Oracle在数据行上没有S锁。

3.2 TM锁（表级锁）

3.2.1 意向锁的引出

表是由行组成的，当我们向某个表加锁时，一方面需要检查该锁的申请是否与原有的表级锁相容；另一方面，还要检查该锁是否与表中的每一行上的锁相容。比如一个事务要在一个表上加S锁，如果表中的一行已被另外的事务加了X锁，那么该锁的申请也应被阻塞。如果表中的数据很多，逐行检查锁标志的开销将很大，系统的性能将会受到影响。为了解决这个问题，可以在表级引入新的锁类型来表示其所属行的加锁情况，这就引出了"意向锁"的概念。

意向锁的含义是如果对一个结点加意向锁，则说明该结点的下层结点正在被加锁；对任一结点加锁时，必须先对它的上层结点加意向锁。如：对表中的任一行加锁时，必须先对它所在的表加意向锁，然后再对该行加锁。这样一来，事务对表加锁时，就不再需要检查表中每行记录的锁标志位了，系统效率得以大大提高。

3.2.2 意向锁的类型

由两种基本的锁类型（S锁、X锁），可以自然地派生出两种意向锁：

意向共享锁（Intent Share Lock，简称IS锁）：如果要对一个数据库对象加S锁，首先要对其上级结点加IS锁，表示它的后裔结点拟（意向）加S锁；

意向排它锁（Intent Exclusive Lock，简称IX锁）：如果要对一个数据库对象加X锁，首先要对其上级结点加IX锁，表示它的后裔结点拟（意向）加X锁。

另外，基本的锁类型（S、X）与意向锁类型（IS、IX）之间还可以组合出新的锁类型，理论上可以组合出4种，即：S+IS，S+IX，X+IS，X+IX，但稍加分析不难看出，实际上只有S+IX有新的意义，其它三种组合都没有使锁的强度得到提高（即：S+IS=S，X+IS=X，X+IX=X，这里的"="指锁的强度相同）。所谓锁的强度是指对其它锁的排斥程度。

这样我们又可以引入一种新的锁的类型：

共享意向排它锁（Shared Intent Exclusive Lock,简称SIX锁）：如果对一个数据库对象加SIX锁，表示对它加S锁，再加IX锁，即SIX=S+IX。例如：事务对某个表加SIX锁，则表示该事务要读整个表（所以要对该表加S锁），同时会更新个别行（所以要对该表加IX锁）。

这样数据库对象上所加的锁类型就可能有5种：即S、X、IS、IX、SIX。

具有意向锁的多粒度封锁方法中任意事务T要对一个数据库对象加锁，必须先对它的上层结点加意向锁。申请封锁时应按自上而下的次序进行；释放封锁时则应按自下而上的次序进行；具有意向锁的多粒度封锁方法提高了系统的并发度，减少了加锁和解锁的开销。

3.3 Oracle的TM锁（表级锁）

Oracle的DML锁（数据锁）正是采用了上面提到的多粒度封锁方法，其行级锁虽然只有一种（即X锁），但其TM锁（表级锁）类型共有5种，分别称为共享锁（S锁）、排它锁（X锁）、行级共享锁（RS锁）、行级排它锁（RX锁）、共享行级排它锁（SRX锁），与上面提到的S、X、IS、IX、SIX相对应。需要注意的是，由于Oracle在行级只提供X锁，所以与RS锁（通过SELECT … FOR UPDATE语句获得）对应的行级锁也是X锁（但是该行数据实际上还没有被修改），这与理论上的IS锁是有区别的。 锁的兼容性是指当一个应用程序在表（行）上加上某种锁后，其他应用程序是否能够在表（行）上加上相应的锁，如果能够加上，说明这两种锁是兼容的，否则说明这两种锁不兼容，不能对同一数据对象并发存取。

下表为Oracle数据库TM锁的兼容矩阵（Y=Yes，表示兼容的请求； N=No，表示不兼容的请求；-表示没有加锁请求）：

表五：Oracle数据库TM锁的相容矩阵

T2 T1	S	X	RS	RX	SRX	-
S	Y	N	Y	N	N	Y
X	N	N	N	N	N	Y
RS	Y	N	Y	Y	Y	Y
RX	N	N	Y	Y	N	Y
SRX	N	N	Y	N	N	Y
-	Y	Y	Y	Y	Y	Y

一方面，当Oracle执行SELECT…FOR UPDATE、INSERT、UPDATE、DELETE等DML语句时，系统自动在所要操作的表上申请表级RS锁（SELECT…FOR UPDATE）或RX锁（INSERT、UPDATE、DELETE），当表级锁获得后，系统再自动申请TX锁，并将实际锁定的数据行的锁标志位置位（指向该TX锁）；另一方面，程序或操作人员也可以通过LOCK TABLE语句来指定获得某种类型的TM锁。下表是笔者总结了Oracle中各SQL语句产生TM锁的情况：

表六：Oracle数据库TM锁小结

SQL 语句	表 锁 模式	允许的锁模式
Select * from table_name……	无	RS、RX、S、SRX、X
Insert into table_name……	RX	RS、RX
Update table_name……	RX	RS、RX
Delete from table_name……	RX	RS、RX
Select * from table_name for update	RS	RS、RX、S、SRX
lock table table_name in row share mode	RS	RS、RX、S、SRX
lock table table_name in row exclusive mode	RX	RS、RX
lock table table_name in share mode	S	RS、S
lock table table_name in share row exclusive mode	SRX	RS
lock table table_name in exclusive mode	X	无

我们可以看到，通常的DML操作（SELECT…FOR UPDATE、INSERT、UPDATE、DELETE），在表级获得的只是意向锁（RS或RX），其真正的封锁粒度还是在行级；另外，Oracle数据库的一个显著特点是，在缺省情况下，单纯地读数据（SELECT）并不加锁，Oracle通过回滚段（Rollback segment）来保证用户不读"脏"数据。这些都提高了系统的并发程度。

由于意向锁及数据行上锁标志位的引入，减小了Oracle维护行级锁的开销，这些技术的应用使Oracle能够高效地处理高度并发的事务请求。

4 DB2多粒度封锁机制的监控

在DB2中对锁进行监控主要有两种方式，第一种方式是快照监控，第二种是事件监控方式。

4.1 快照监控方式

当使用快照方式进行锁的监控前，必须把监控锁的开关打开，可以从实例级别和会话级别打开，具体命令如下：

db2 update dbm cfg using dft_mon_lock on(实例级别)

db2 update monitor switches using lock on(会话级别，推荐使用)

当开关打开后，可以执行下列命令来进行锁的监控

db2 get snapshot for locks on ebankdb（可以得到当前数据库中具体锁的详细信息）

db2 get snapshot for locks on ebankdb

Fri Aug 15 15:26:00 JiNan 2004（红色为锁的关键信息）

```
            Database Lock Snapshot
Database name                               = DEV
Database path                               = /db2/DEV/db2dev/NODE0000/SQL00001/
Input database alias                        = DEV
Locks held                                  = 49
Applications currently connected            = 38
Agents currently waiting on locks           = 6
Snapshot timestamp                          = 08-15-2003 15:26:00.951134
Application handle                          = 6
Application ID                              = *LOCAL.db2dev.030815021007
Sequence number                             = 0001
Application name                            = disp+work
Authorization ID                            = SAPR3
Application status                          = UOW Waiting
Status change time                          =
Application code page                       = 819
Locks held                                  = 0
Total wait time (ms)                        = 0
Application handle                          = 97
Application ID                              = *LOCAL.db2dev.030815060819
Sequence number                             = 0001
Application name                            = tp
Authorization ID                            = SAPR3
Application status            = Lock-wait
Status change time                          = 08-15-2003 15:08:20.302352
Application code page                       = 819
Locks held                                  = 6
Total wait time (ms)                        = 1060648
  Subsection waiting for lock               = 0
  ID of agent holding lock                  = 100
  Application ID holding lock               = *LOCAL.db2dev.030815061638
  Node lock wait occurred on                = 0
  Lock object type                          = Row
  Lock mode                                 = Exclusive Lock (X)
  Lock mode requested                       = Exclusive Lock (X)
  Name of tablespace holding lock           = PSAPBTABD
  Schema of table holding lock              = SAPR3
  Name of table holding lock                = TPLOGNAMES
  Lock wait start timestamp                 = 08-15-2003 15:08:20.302356
  Lock is a result of escalation            = NO
List Of Locks
  Lock Object Name            = 29204
 Node number lock is held at = 0
 Object Type         = Table
 Tablespace Name              = PSAPBTABD
 Table Schema                 = SAPR3
 Table Name                   = TPLOGNAMES
 Mode            = IX
 Status                       = Granted
 Lock Escalation       = NO
```

db2 get snapshot for database on dbname |grep -i locks（**UNIX,LINUX**平台）

```
Locks held currently                        = 7
Lock waits                                  = 75
Time database waited on locks (ms)          = 82302438
Lock list memory in use (Bytes)             = 20016
Deadlocks detected                          = 0
Lock escalations                            = 8
Exclusive lock escalations                  = 8
Agents currently waiting on locks           = 0
Lock Timeouts                               = 20
```

db2 get snapshot for database on dbname |find /i "locks"(NT平台)

db2 get snapshot for locks for applications agentid 45(注：45为应用程序句柄)

```
 Application handle                         = 45
Application ID                             = *LOCAL.db2dev.030815021827
Sequence number                            = 0001
Application name                           = tp
Authorization ID                           = SAPR3
Application status                         = UOW Waiting
Status change time                         =
Application code page                      = 819
Locks held                                 = 7
Total wait time (ms)                       = 0
List Of Locks
 Lock Object Name            = 1130185838
 Node number lock is held at = 0
 Object Type                 = Key Value
 Tablespace Name             = PSAPBTABD
 Table Schema                = SAPR3
 Table Name                  = TPLOGNAMES
 Mode                        = X
 Status                      = Granted
 Lock Escalation             = NO
 Lock Object Name            = 14053937
 Node number lock is held at = 0
 Object Type                 = Row
 Tablespace Name             = PSAPBTABD
 Table Schema                = SAPR3
 Table Name                  = TPLOGNAMES
 Mode                        = X
 Status                      = Granted
 Lock Escalation             = NO
```

也可以执行下列表函数（注：在**DB2 V8**之前只能通过命令，**DB2 V8**后可以通过表函数，推荐使用表函数来进行锁的监控）

db2 select * from table(snapshot_lock('DBNAME',-1)) as locktable监控锁信息
db2 select * from table(snapshot_lockwait('DBNAME',-1) as lock_wait_table监控应用程序锁等待的信息

4.2 事件监控方式：

当使用事件监控器进行锁的监控时候，只能监控死锁(死锁的产生是因为由于锁请求冲突而不能结束事务，并且该请求冲突不能够在本事务内解决。通常是两个应用程序互相持有对方所需要的锁，在得不到自己所需要的锁的情况下，也不会释放现有的锁)的情况，具体步骤如下：

db2 create event monitor dlock for deadlocks with details write to file '$HOME/dir'
db2 set event monitor dlock state 1
db2evmon -db dbname -evm dlock看具体的死锁输出（如下图）

```
Deadlocked Connection ...
Deadlock ID:   4
Participant no.: 1
Participant no. holding the lock: 2
Appl Id: G9B58B1E.D4EA.08D387230817
Appl Seq number: 0336
Appl Id of connection holding the
lock: G9B58B1E.D573.079237231003
Seq. no. of connection holding the lock: 0126
Lock wait start time: 06/08/2005 08:10:34.219490
Lock Name       : 0x000201350000030E0000000052
Lock Attributes : 0x00000000
Release Flags   : 0x40000000
Lock Count      : 1
Hold Count      : 0
Current Mode    : NS  - Share (and Next Key Share)
Deadlock detection time: 06/08/2005 08:10:39.828792
Table of lock waited on      : ORDERS
Schema of lock waited on     : DB2INST1
Tablespace of lock waited on : USERSPACE1
Type of lock: Row
Mode of lock: NS  - Share (and Next Key Share)
Mode application requested on lock: X   - Exclusive
Node lock occured on: 0
Lock object name: 782
Application Handle: 298
Deadlocked Statement:
Type     : Dynamic
Operation: Execute
Section  : 34
Creator  : NULLID
Package  : SYSSN300
Cursor   : SQL_CURSN300C34
Cursor was blocking: FALSE
Text     : UPDATE ORDERS  SET TOTALTAX = ?,
TOTALSHIPPING = ?,
LOCKED = ?, TOTALTAXSHIPPING = ?, STATUS = ?,
 FIELD2 = ?, TIMEPLACED = ?,
FIELD3 = ?, CURRENCY = ?, SEQUENCE = ?,
TOTALADJUSTMENT = ?, ORMORDER = ?,
SHIPASCOMPLETE = ?, PROVIDERORDERNUM = ?,
TOTALPRODUCT = ?, DESCRIPTION = ?,
MEMBER_ID = ?, ORGENTITY_ID = ?, FIELD1 = ?,
STOREENT_ID = ?, ORDCHNLTYP_ID = ?,
ADDRESS_ID = ?, LASTUPDATE = ?, COMMENTS = ?,
```

```
NOTIFICATIONID = ? WHERE ORDERS_ID = ?
List of Locks:
Lock Name                   : 0x000201350000030E0000000052
Lock Attributes             : 0x00000000
Release Flags               : 0x40000000
Lock Count                  : 2
Hold Count                  : 0
Lock Object Name            : 782
Object Type                 : Row
Tablespace Name             : USERSPACE1
Table Schema                : DB2INST1
Table Name                  : ORDERS
Mode                        : X   - Exclusive
Lock Name                   : 0x00020040000029B30000000052
Lock Attributes             : 0x00000020
Release Flags               : 0x40000000
Lock Count                  : 1
Hold Count                  : 0
Lock Object Name            : 10675
Object Type                 : Row
Tablespace Name             : USERSPACE1
Table Schema                : DB2INST1
Table Name                  : BKORDITEM
Mode                        : X   - Exclusive（略去后面信息）
```

5 Oracle 多粒度封锁机制的监控

为了监控Oracle系统中锁的状况，我们需要对几个系统视图有所了解：

5.1 v$lock视图

v$lock视图列出当前系统持有的或正在申请的所有锁的情况，其主要字段说明如下：

表七：v$lock视图主要字段说明

字段名称	类型	说明
SID	NUMBER	会话（SESSION）标识；
TYPE	VARCHAR（2）	区分该锁保护对象的类型；
ID1	NUMBER	锁标识 1；（主要关注的内容）
ID2	NUMBER	锁标识 2；
LMODE	NUMBER	锁模式：0（None），1（null），2（row share），3（row exclusive），4 （share），5（share row exclusive），6（exclusive）
REQUEST	NUMBER	申请的锁模式：具体值同上面的 LMODE
CTIME	NUMBER	已持有或等待锁的时间；
BLOCK	NUMBER	是否阻塞其它锁申请；

其中在TYPE字段的取值中，本文只关心TM、TX两种DML锁类型；

5.2 v$locked_object视图

v$locked_object视图列出当前系统中哪些对象正被锁定，其主要字段说明如下：

表八：v$locked_object视图字段说明

字段名称	类型	说明
XIDUSN	NUMBER	回滚段号；
XIDSLOT	NUMBER	槽号；
XIDSQN	NUMBER	序列号；
OBJECT_ID	NUMBER	被锁对象标识；
SESSION_ID	NUMBER	持有锁的会话（SESSION）标识；
ORACLE_USERNAME	VARCHAR2（30）	持有该锁的用户的 Oracle 用户名；
OS_USER_NAME	VARCHAR2（15）	持有该锁的用户的操作系统用户名；
PROCESS	VARCHAR2（9）	操作系统的进程号；
LOCKED_MODE	NUMBER	锁模式，取值同表三中的 LMODE；

5.3 Oracle锁监控脚本

根据上述系统视图，可以编制脚本来监控数据库中锁的状况。

5.3.1 showlock.sql

第一个脚本showlock.sql，该脚本通过连接v$locked_object与all_objects两视图，显示哪些对象被哪些会话锁住：

```
/* showlock.sql */
column o_name format a10
column lock_type format a20
column object_name format a15
select rpad(oracle_username,10) o_name,session_id sid,
decode(locked_mode,0,'None',1,'Null',2,'Row share',
```

```
3,'Row Exclusive',4,'Share',5,'Share Row Exclusive',6,'Exclusive') lock_type,
object_name ,xidusn,xidslot,xidsqn
from v$locked_object,all_objects
where v$locked_object.object_id=all_objects.object_id;
```

5.3.2 showalllock.sql

第二个脚本showalllock.sql，该脚本主要显示当前所有TM、TX锁的信息；

```
/* showalllock.sql */
select sid,type,id1,id2,
decode(lmode,0,'None',1,'Null',2,'Row share',
3,'Row Exclusive',4,'Share',5,'Share Row Exclusive',6,'Exclusive')
lock_type,request,ctime,block
from v$lock
where TYPE IN('TX','TM');
```

6 DB2 多粒度封锁机制示例

以下示例均运行在DB2 UDB中，适用所有数据库版本。首先打开三个命令行窗口(DB2 CLP)，其中两个（以下用SESS#1、SESS#2表示）以db2admin用户连入数据库，以操作SAMPLE库中提供的示例表（employee）；另一个（以下用SESS#3表示）以db2admin用户连入数据库，对执行的每一种类型的SQL语句监控加锁的情况；希望读者通过这种方式对每一种类型的SQL语句监控加锁的情况。(因为示例篇幅很大，笔者在此就不做了，建议读者用类似方法验证加锁情况)

```
/home/db2inst1>db2 +c update employee set comm=9999(SESS#1)
/home/db2inst1>db2 +c select * from employee(SESS#2处于lock wait)
/home/db2inst1>db2 +c get snapshot for locks on sample(SESS#3监控加锁情况)
```

注：db2 +c为不自动提交（commit）SQL语句，也可以通过 db2 update command options using c off关闭自动提交（autocommit，缺省是自动提交）

7 总结

总的来说，DB2的锁和Oracle的锁主要有以下大的区别：

1．Oracle通过具有意向锁的多粒度封锁机制进行并发控制，保证数据的一致性。其DML锁（数据锁）分为两个层次（粒度）：即表级和行级。通常的DML操作在表级获得的只是意向锁（RS或RX），其真正的封锁粒度还是在行级；DB2也是通过具有意向锁的多粒度封锁机制进行并发控制，保证数据的一致性。其DML锁（数据锁）分为两个层次（粒度）：即表级和行级。通常的DML操作在表级获得的只是意向锁（IS，SIX或IX），其真正的封锁粒度也是在行级；另外，在Oracle数据库中，单纯地读数据（SELECT）并不加锁，这些都提高了系统的并发程度，Oracle强调的是能够"读"到数据，并且能够快速的进行数据读取。而DB2的锁强调的是"读一致性"，进行读数据(SELECT)时会根据不同的隔离级别（RR,RS,CS）而分别加S,IS,IS锁，只有在使用UR隔离级别时才不加锁。从而保证不同应用程序和用户读取的数据是一致的。

2. 在支持高并发度的同时，DB2和Oracle对锁的操纵机制有所不同：Oracle利用意向锁及数据行上加锁标志位等设计技巧，减小了Oracle维护行级锁的开销，使其在数据库并发控制方面有着一定的优势。而DB2中对每个锁会在锁的内存（locklist）中申请分配一定字节的内存空间，具体是X锁64字节内存，S锁32字节内存（注：DB2 V8之前是X锁72字节内存而S锁36字节内存）。

3. Oracle数据库中不存在锁升级，而DB2数据库中当数据库表中行级锁的使用超过locklist*maxlocks会发生锁升级。

4. 在Oracle中当一个session对表进行insert，update，delete时候，另外一个session仍然可以从Orace回滚段或者还原表空间中读取该表的前映象（before image）；而在DB2中当一个session对表进行insert，update，delete时候，另外一个session仍然在读取该表数据时候会处于lock wait状态，除非使用UR隔离级别可以读取第一个session的未提交的值；所以Oracle同一时刻不同的session有读不一致的现象，而DB2在同一时刻所有的session都是"读一致"的。

8 结束语

DB2中关于并发控制（锁）的建议

1．正确调整locklist，maxlocks，dlchktime和locktimeout等和锁有关的数据库配置参数（locktimeout最好不要等于-1）。如果锁内存不足会报SQL0912错误而影响并发。

2．写出高效而简洁的SQL语句（非常重要）。

3．在业务逻辑处理完后尽可能快速commit释放锁。

4．对引起锁等待（SQL0911返回码68）和死锁（SQL0911返回码2）的SQL语句创建最合理的索引（非常重要，尽量创建复合索引和包含索引）。

5．使用 altER TABLE 语句的 LOCKSIZE 参数控制如何在持久基础上对某个特定表进行锁定。检查syscat.tables中locksize字段，尽量在符合业务逻辑的情况下，每个表中该字段为"R"（行级锁）。

6．根据业务逻辑使用正确的隔离级别（RR，RS，CS和UR）。

7．当执行大量更新时，更新之前，在整个事务期间锁定整个表（使用 SQL LOCK TABLE 语句）。这只使用了一把锁从而防止其它事务进行这些更新，但是对于其他用户它的确减少了数据并发性。

免责声明和公开声明