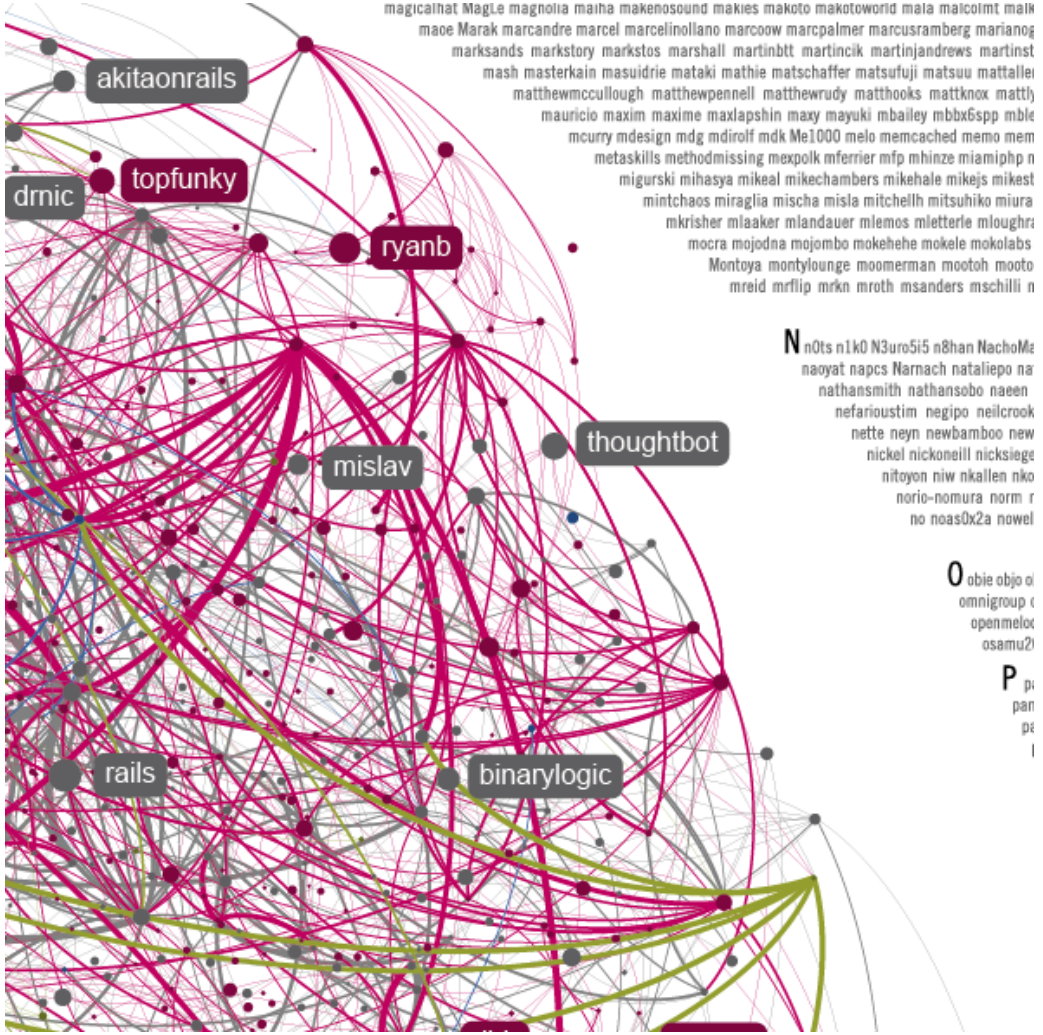
DB2设计与性能优化

第7周



法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

课程内容

- 并发
  - 基本概念
  - 并发问题
- 隔离级
  - 四种隔离级
  - 选择隔离级
  - Currently committed（V9.7 新引入）
- 锁
  - 锁概念
  - 常见锁现象（锁升级、锁等待、锁超时、死锁）
  - 锁监控
- DB2和Oracle锁机制对比
- 设计高并发应用

DB2并发

- DB2支持多个终端用户

- 必须控制对同一个数据集的访问以保证：
  - 数据的完整性(Data Integrity)
  - 数据的一致性(Data Consistency)

- 如果没有某种形式的并发控制，可能会遇到以下问题:
  - 丢失更新（Lost update）
  - 未提交的读（Uncommitted read）
  - 不可重复读(Non-repeatable read)
  - 幻象读（Phantom read）

- DB2使用基于锁的并发控制机制(locking base concurrency control)

DB2并发: 基本概念

什么是事务(transaction)?

- 事务又称为交易,是并发控制的基本单位。它是由一个或多个读/写数据库的SQL语句组成的一个操作序列。
- 这些操作要么都执行，要么都不执行。
- 事务是一个不可分割的工作单位(Unit of Work, UOW)。
- 事务用来保持数据的完整性:
  - 在并发访问数据库时
  - 在恢复数据库时

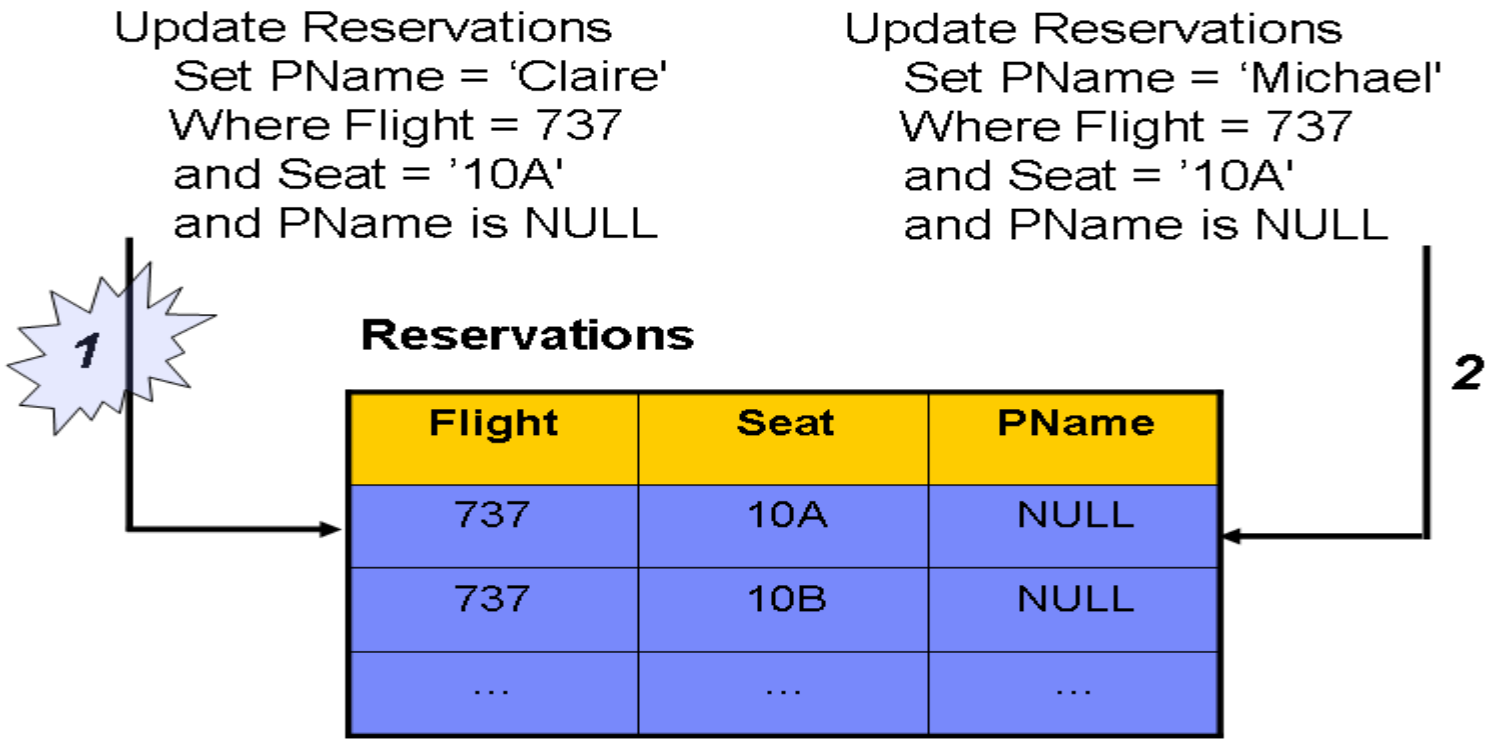
提交/回滚(Commit/Rollback)

- 确保事务是原子的(atomic)
- 提交使对数据库做出的更改成为永久的
- DB2保证已经提交的更新在数据库失败时能够持久
- 回滚使对数据库做出的更改被撤销(undo)

并发问题: 丢失更新

什么是丢失更新(Lost Update)?

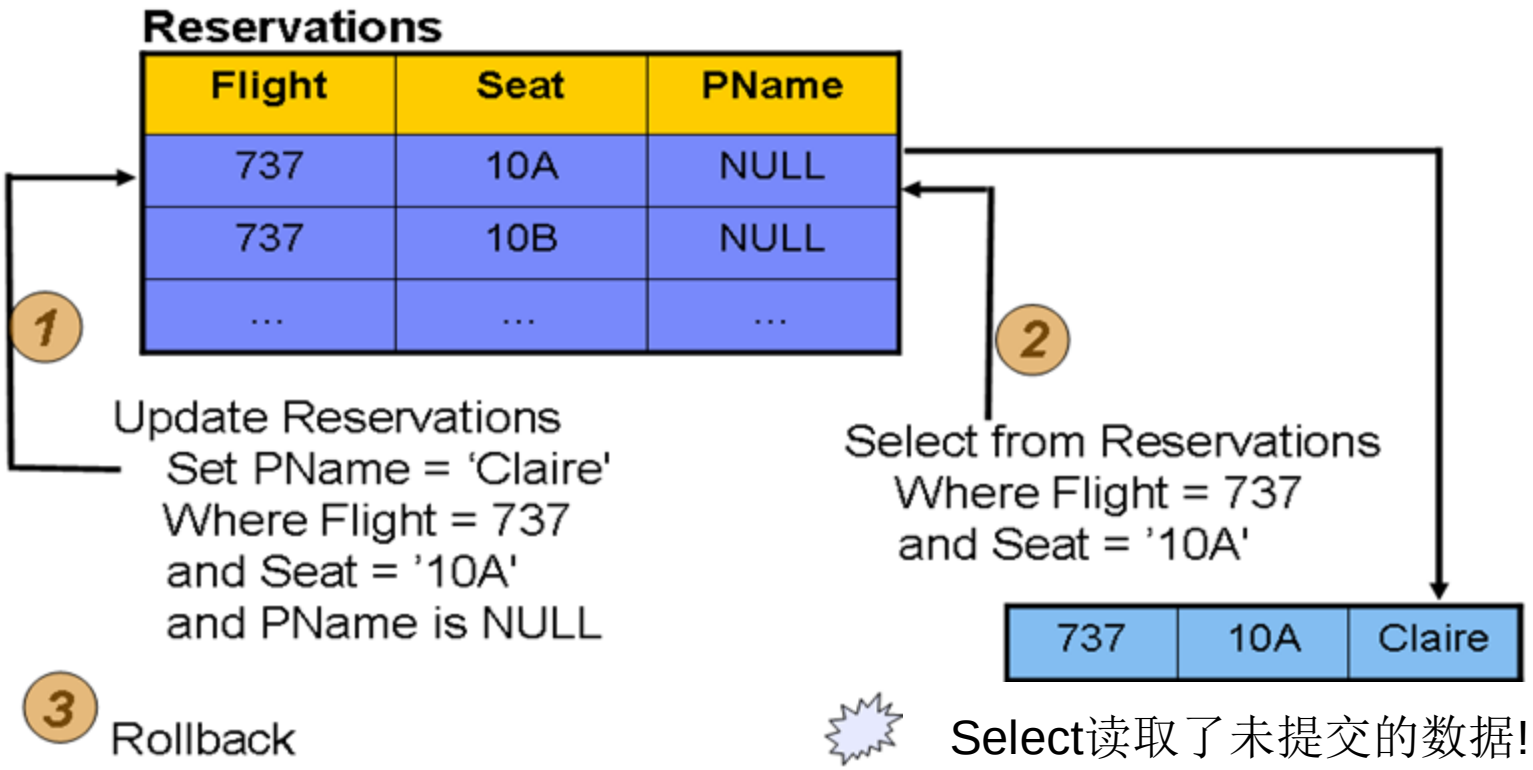
事务T1读取了数据，并进行了一些操作，然后更新数据。而事务T2也执行了相同的操作，导致事物双方可能会覆盖对方的更新。



- 允许两个以上的用户做读取，然后在没有并发控制机制的情况下更新了同样的数据
- 第一次的更新可能会丢失

并发问题: 未提交的读

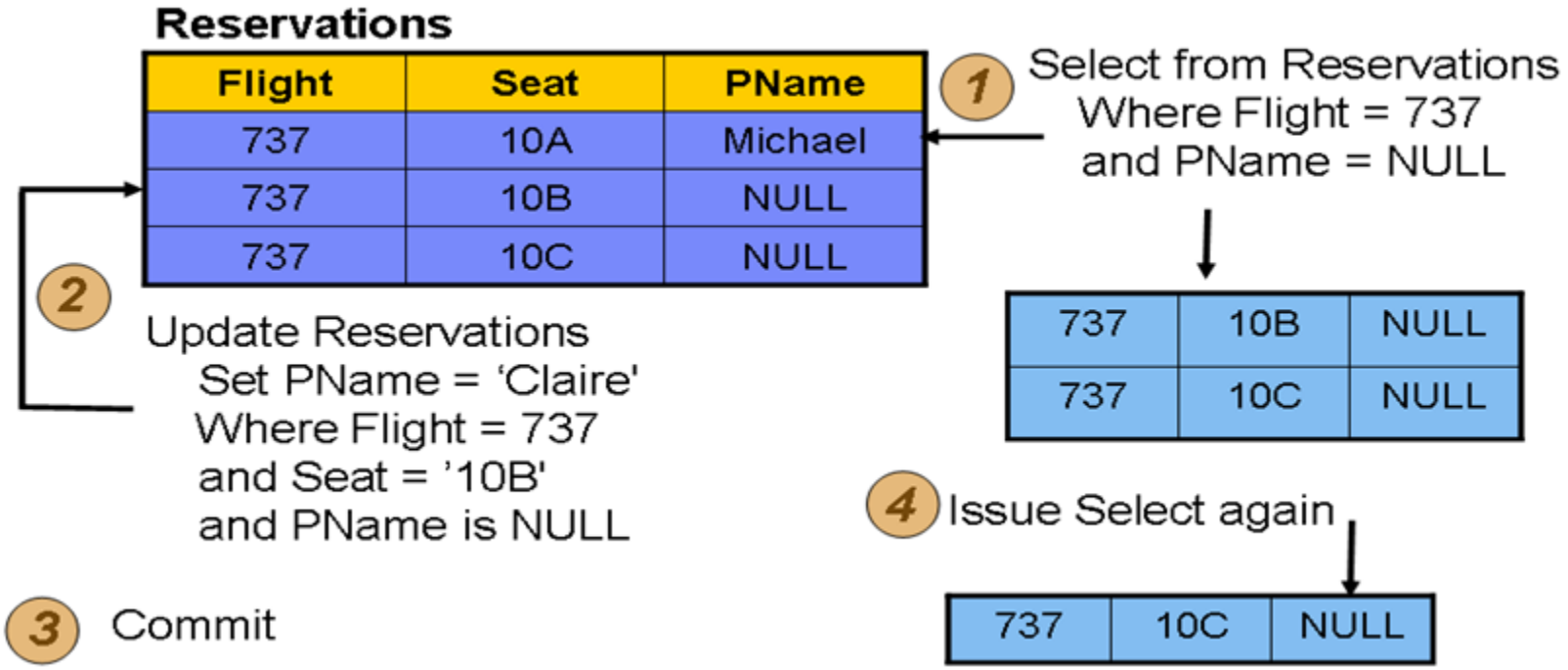
什么是未提交的读(Uncommitted Read, 也叫脏读)?
事务T1更新了数据还没有提交，但是事务T2读取了相同的数据，则T2读取的其实是错误的数据。



- 一个终端用户正在更新数据，另一个终端用户在没有并发控制机制的情况下读取同样数据
- 未提交的数据可能会被读取

并发问题: 不可重复读

什么是不可重复读(Non Repeatable Read)?
一个事务的两次读取中，读到的数据是不同的, 即在两次事务之间发生了数据的修改。

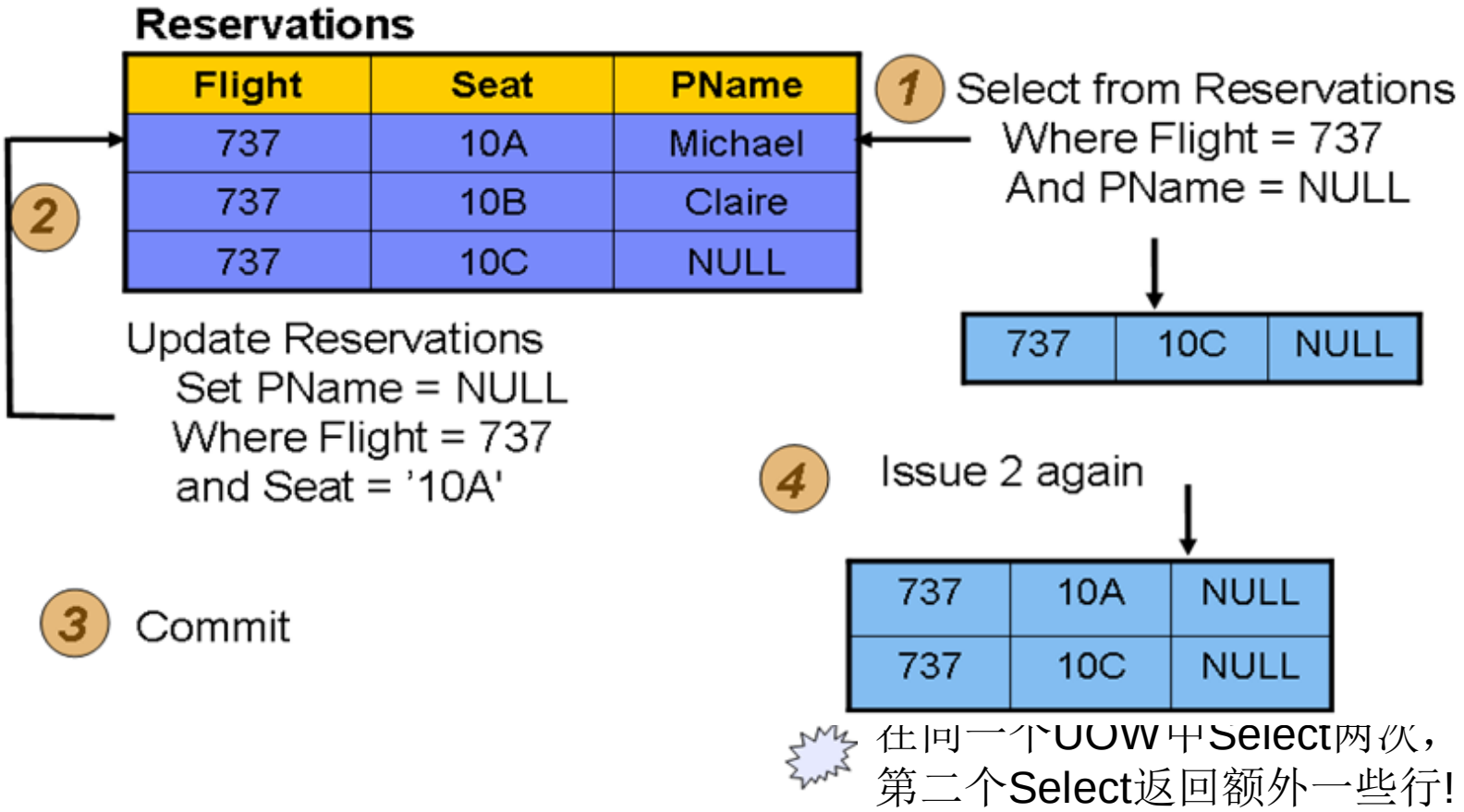


在同一个UOW中Select两次，
每次返回不同的结果!

- 一个终端用户在同一个事务中读取同一个数据集
- 每次返回不同的结果

并发问题: 幻像读

什么是幻像读(Phantom Read)?
幻象的问题与不可重复读相似，不同之处在于随后的行读取上。您可能会获取额外一些行。



- 一个终端用户在同一个事务中读取同一个数据集
- 第三次会返回额外一些行

DB2的隔离级别

DB2使用四种类型的隔离级别(isolation levels)

- 未提交读（UR）：不加锁，可以读到未提交的数据
- 游标稳定性（CS）：读到哪一行就在这一行加S锁
- 读稳定性（RS）：查询的结果集加S锁
- 可重复读（RR）：凡是读过的行都加S锁

DB2的隔离级别

四种隔离级别所能解决的问题对照表：

隔离级别	丢失更新	未提交的读	不可重复的读	幻像读
UR	**Yes**	No	No	No
CS	**Yes**	**Yes**	No	No
RS	**Yes**	**Yes**	**Yes**	No
RR	**Yes**	**Yes**	**Yes**	**Yes**

为您的应用程序选择合适的隔离级别：

- 一般来说，隔离级别越高，它能提供的并发越少，性能越低
- 读比较高的环境，使用UR或CS
- 读和写都比较高的环境，若需要大的数据稳定性，使用RR或RS

DB2隔离级别的粒度

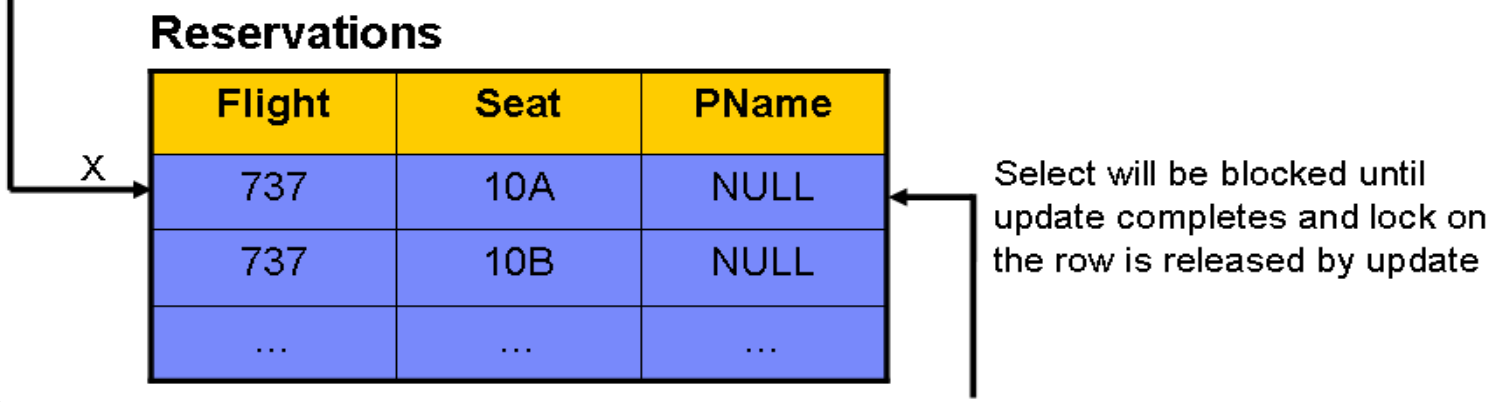
- 隔离级别的粒度
  - 语句级
    - SELECT … WITH {UR|CS|RS|RR}
  - 会话级Session (应用)
    - 命令行：CHANGE ISOLATION TO {UR|CS|RS|RR}
    - JDBC：Connection con = DriverManager.getConnection(this.aURL ,this.USER, this.PWD);
      con.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED)
  - 对动态SQL, 在运行时设置隔离级别
    - SET CURRENT ISOLATION {UR|CS|RS|RR|RESET}
  - 对嵌入式SQL, 在绑定或预编译时设置隔离级别
    - BIND <filename> … ISOLATION {UR|CS|RS|RR}…
    - PRECOMPILE <filename> … ISOLATION {UR|CS|RS|RR}…

Currently Committed

V97New

- 例子: 未激活Currently Committed

1. Update Reservations Set PName = 'Claire'
   Where Flight = 737 and Seat = '10A' and PName is NULL



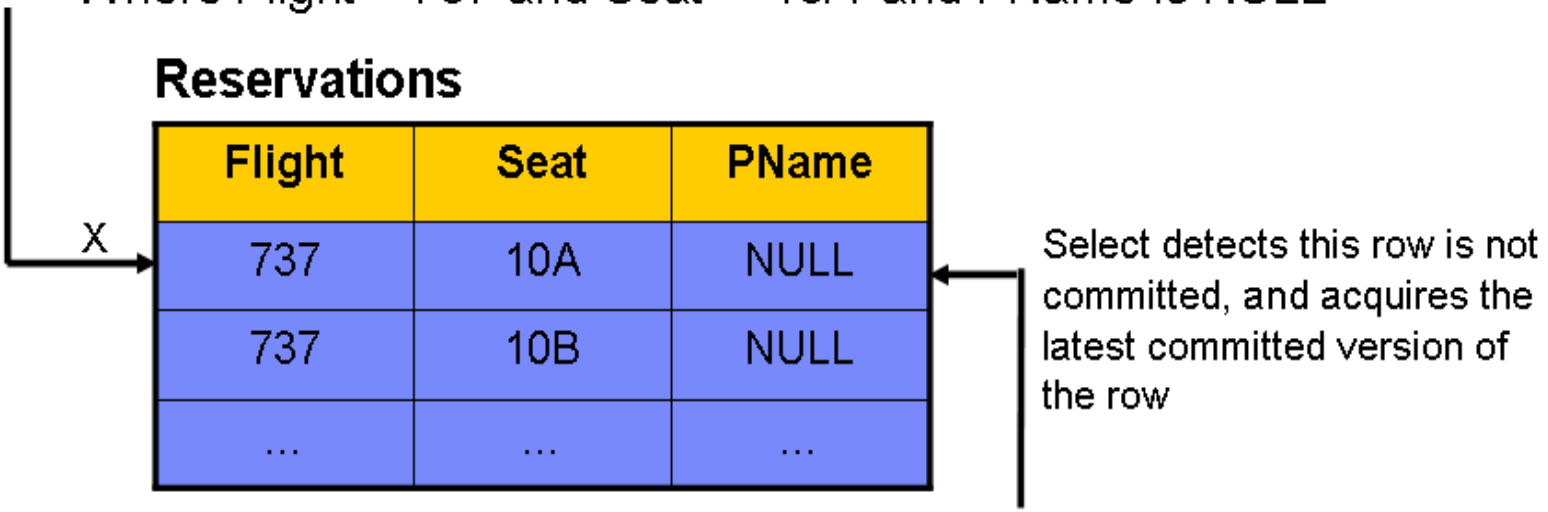
2. Select from Reservations Where Flight = 737 and PName is NULL
3. Update commits or rollback, release X lock
4. Select proceeds

If update does not commit or rollback for a long time, select will be on lock wait for a long time

Currently Committed

- 例子: 激活Currently Committed

1 Update Reservations Set PName = 'Claire'
Where Flight = 737 and Seat = '10A' and PName is NULL



2 Select from Reservations Where Flight = 737 and PName is NULL

3 Select proceeds and returns the result set

4 Update commits or rollback, release X lock

Currently Committed

- 什么是Currently Committed?
  - CS隔离级下的一种新的语义
  - 返回最近已落实的数据
  - 读事务不再被写事务阻塞
  - V9.7新引入，默认激活
- 使用Currently Committed有什么好处?
  - 适合于这样的应用：可以接受最近已落实的数据，不希望被其他写事务阻塞
  - 提高事务吞吐量，从而系统的并发性提高

锁的特性

- DB2使用锁机制来保证数据完整性，有下面的属性：
  - 加锁对象：表空间、表、分区、块和行
  - 使用方式：共享（Share）和排它（Exclusive）
  - 持有时间：瞬时、短时和长时
- 在需要时，锁被DB2自动获取
  - 唯一的例外：用户使用LOCK TABLE命令显示锁表
- 事务提交或回滚时，锁被释放
- DB2的锁策略基于下面的因素：
  - 隔离级
  - 语句类型 (DDL 还是 DML，读还是写)
  - 访问方式（表扫描还是索引扫描）
  - 参数设置（LOCKLIST，MAXLOCKS）
  - 命令（LOCK TABLE或者ALTER TABLE … LOCKSIZE row/table）

为什么会有意向（Intent）锁？

意向锁是提高并发性能的重要机制：

- 意向锁要弱于对应的非意向锁

  所谓强弱，指的是对其他锁排斥的程度。比如，对一张表加锁，IX是比X稍弱的锁，这是因为，IX代表了表中的一部分行将要或已经被锁住，而X是将整个表的所有行锁住。IX和IX是兼容的，因为它们可以代表对同一个表不同的行进行了锁定。

- 意向锁缩短了执行时间

  DB2在对表加其他锁时，通过检查表上是否加了意向锁，就能知道表中是否有一部分行被其他事务锁住，而不需要逐行查看是否有事务对这张表加了锁，这自然节省了时间和系统资源。

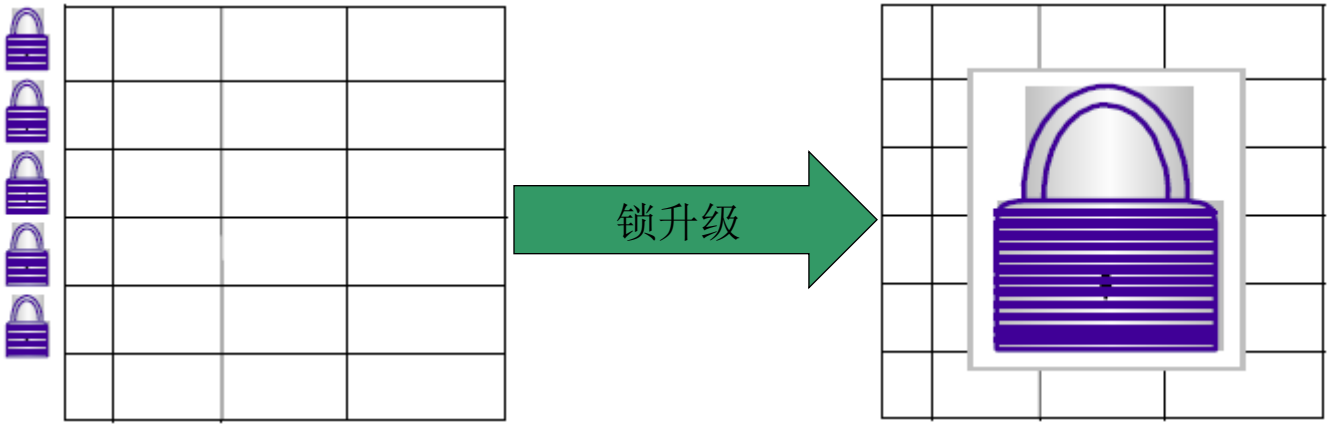
如何选择锁粒度？

锁粒度越小，并发性越好，资源占用越多

- 行锁具有最好的并发性，但是会占用过多资源
- 表锁、分区锁、块锁并发性最差，但是资源使用少
  - 使用ALTER TABLE…LOCKSIZE {ROW|BLOCKINSERT|TABLE}来指定锁粒度，影响对该表上的所有读写事务
    - 默认为ROW，使用行锁
    - BLOCKINSERT指定在插入时使用块锁 （适用MDC表）
    - TABLE 指定使用表锁
  - 使用LOCK TABLE语句
    - 提供应用级别的控制
    - 最小化资源占用
    - 没有UNLOCK语句，事务完成或者回滚后，自动释放

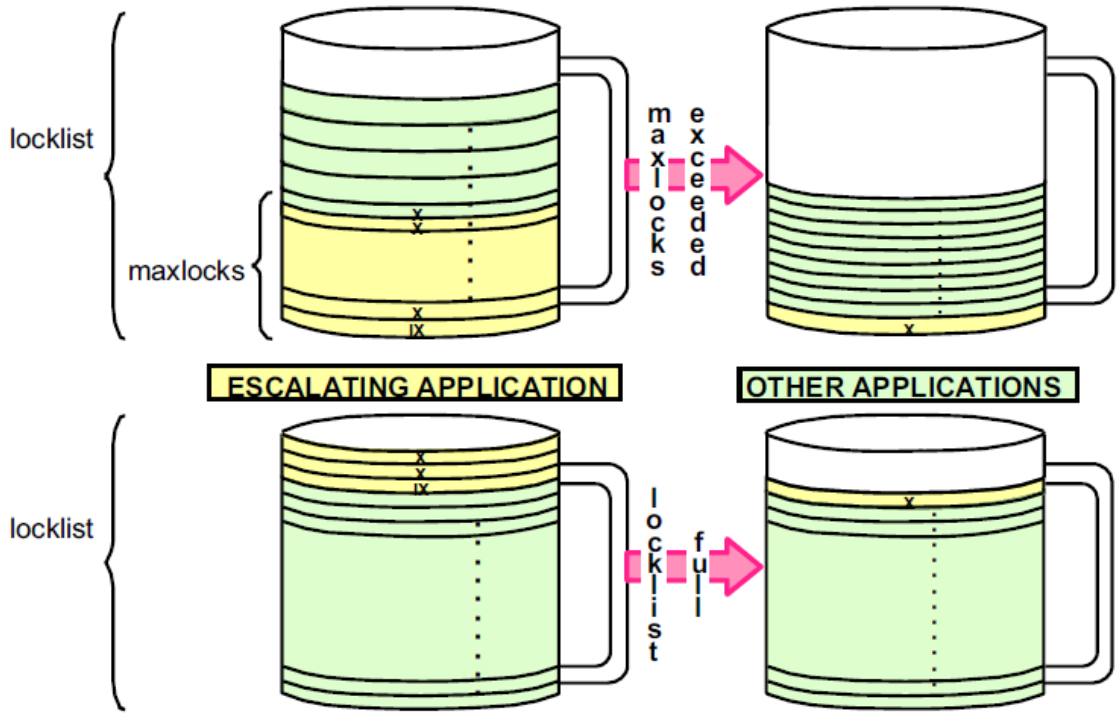
锁升级

- 存储锁的内存不够时，用一个表锁代替多个行锁，以减少锁的数量



锁升级

- 两种情况
  - LOCKLIST满
  - 一个应用使用的锁内存达到MAXLOCKS



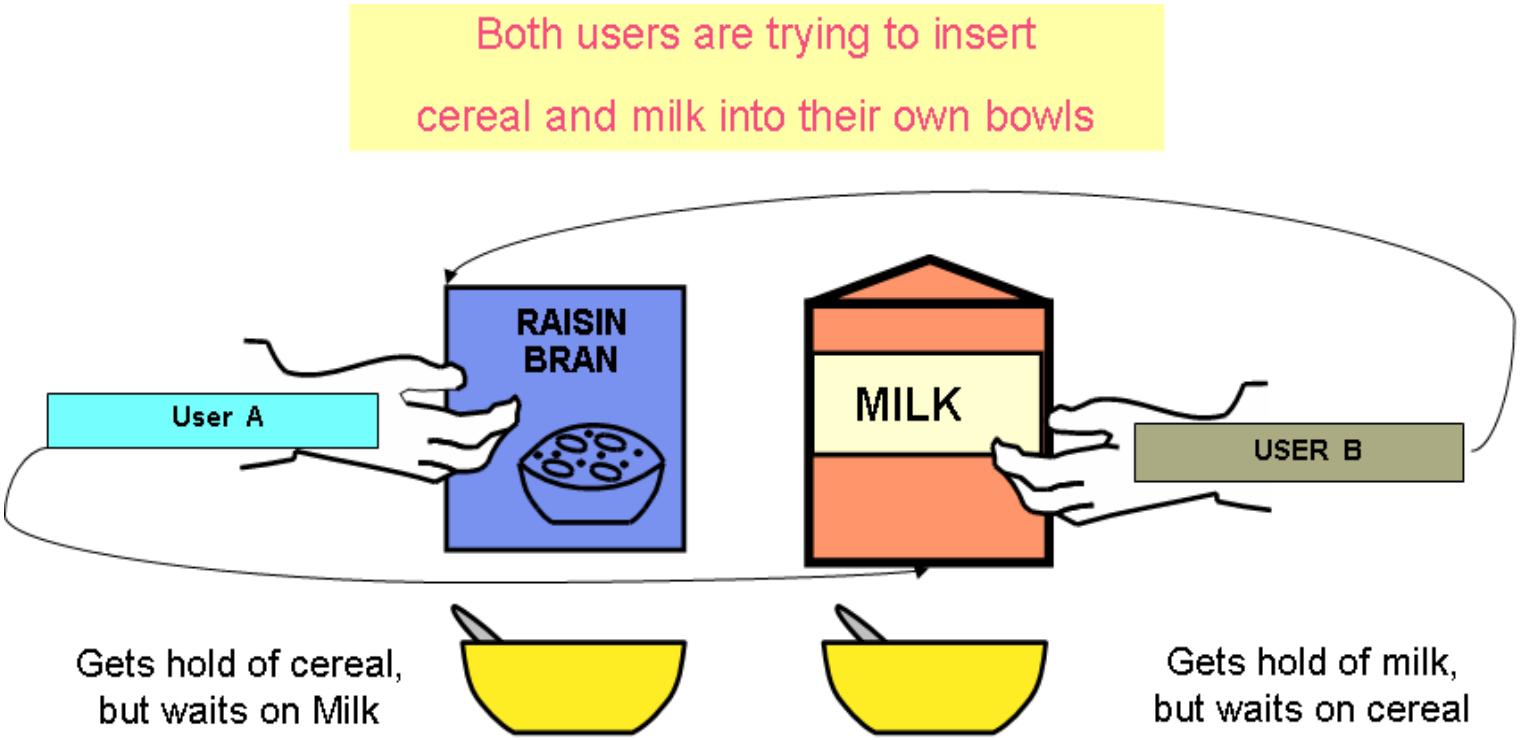
解决锁升级问题

考虑如下方法:

- 调整LOCKLIST和MAXLOCKS数据库配置参数
- 增加提交(commit)的频率
- 重新审视您的隔离级别
  - 使用应用程序能容许的最低的隔离级别
- 如果表对并发访问不关键，可考虑使用LOCK TABLE语句
  - 必须小心选择这样的表
  - 当负载高时,不可能调整LOCKLIST或MAXLOCKS，则减少非关键表的并发访问，从而减少关键表上的锁升级

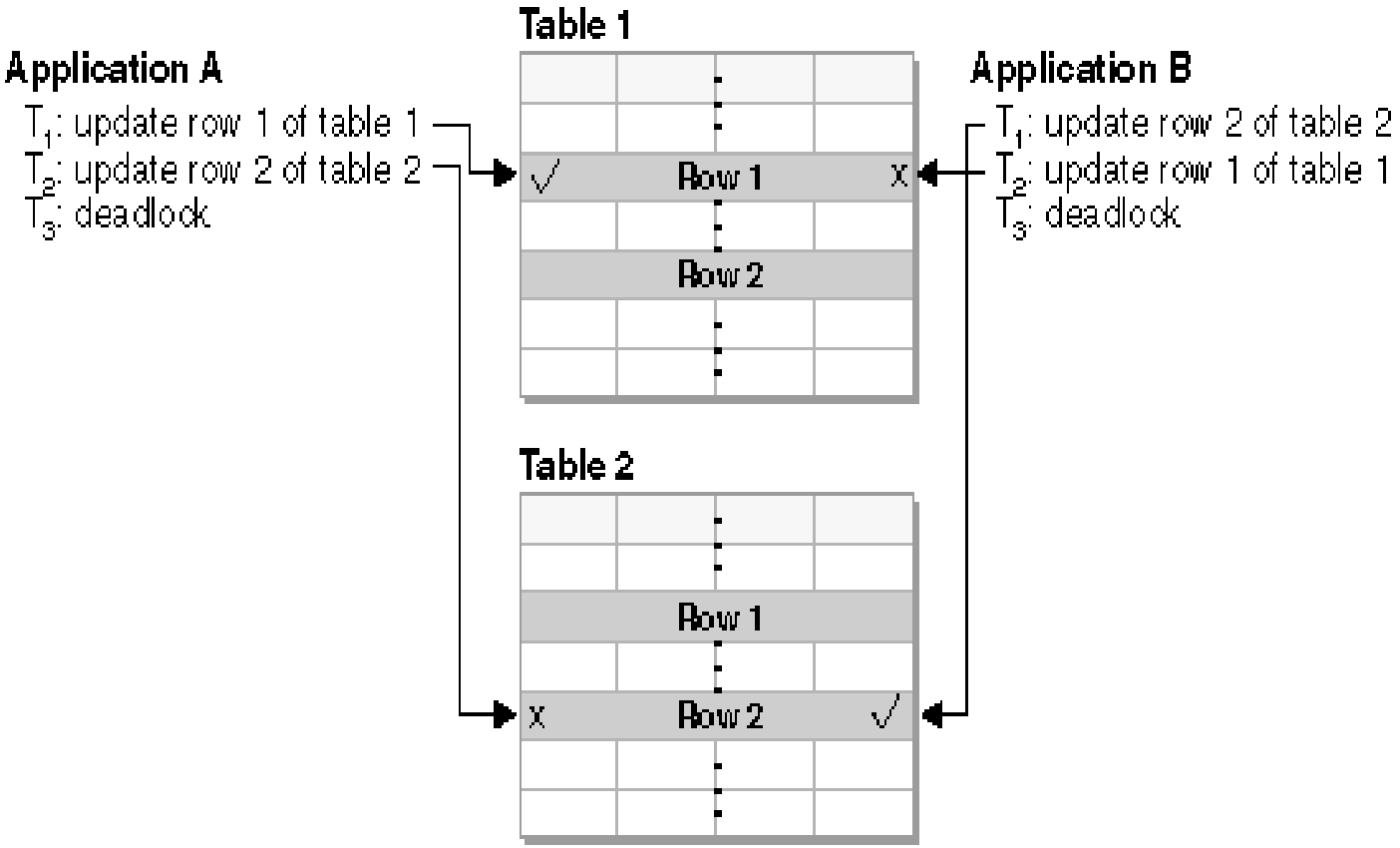
死锁

- 当连接到同一个数据库的两个以上的进程无限期地等待某一资源时，便可能发生死锁。
- 因为每个进程都保持着另一个进程需要的资源，所以这种等待永远都无法解除。
- 死锁是大多数程序设计中都要解决的问题。



DB2的死锁

一个真实的死锁例子：



解决死锁问题

- 用同样的顺序访问数据
- 读取随后更新的数据
  - 解决方案：在SELECT上使用FOR UPDATE子句
  - 这样读取数据会使用U锁而不是NS锁
- 优化访问计划(使用了表扫描而不是索引扫描)
- 使用适合的隔离级别
- 避免并发的DDL
  - DDL会更新和锁定大量的编目行
- 频繁提交
  - 尽可能早地提交DDL语句和DML语句
- 使用LOCKTIMEOUT来帮助避免死锁
  - 锁超时也好不了多少，但有时宁愿选择锁超时而不是死锁

锁超时

锁等待

- 应用程序在请求加锁时可能会等待一个锁:
  - 当应用程序试图访问由另一个应用程序不兼容的锁已经锁定的数据, 并且加锁请求是无条件时会出现锁等待
  - 锁等待可以是无限期的(如果持有锁的应用程序忘记提交或回滚)

锁超时

- DB2提供了锁超时特性来防止应用程序无限期的锁等待
  - 使用LOCKTIMEOUT数据库配置参数启用锁超时检测
    - LOCKTIMEOUT缺省为-1(锁超时检测为关闭状态)
  - 如果锁超时了, 会返回一个sqlcode -911, 原因代码68的错误, 事务将回滚
  - 避免死锁的发生
  - 使用SET CURRENT LOCK TIMEOUT语句在会话级别覆盖数据库级的配置参数LOCKTIMEOUT

解决锁超时问题

- 把数据库配置参数LOCKTIMEOUT调整到合适的大小(若干秒)
- 如果可能的话，避免非常长的事务和WITH HOLD游标
  - 持有锁的时间越长，和其他应用程序争夺资源的机会越大
- 尽早提交如下动作:
  - 写的动作，比如删除，删除和修改
  - 数据定义语言(DDL)语句，比如ALTER, CREATE和DROP
  - BIND和REBIND命令
- 避免查询过大的结果集，特别是在RR隔离级别下
- 尽可能使用较低的隔离级别
- 适当时使用注册表环境变量如DB2_EVALUNCOMMITTED、DB2_SKIPDELETE和DB2_SKIPINSERTED

监控锁的问题

- 可以使用各种不同的监控工具监控锁的问题:
  - db2pd工具
  - 快照监控(Snapshot monitor)
  - 事件监控(Event monitor)
  - 管理日志（文件名是<instance name>.nfy，仅适用于LINUX和UNIX平台）
  - 诊断日志（文件名是db2diag.log）
- 收集下面关于锁的信息:
  - 死锁和锁升级
    - 对一个特定的数据库，死锁或锁升级是否会出现，何时出现
    - 涉及到哪一个应用程序
    - 涉及到什么锁
  - 锁等待和锁超时
    - 应用程序是否处在锁等待或锁超时
    - 哪一个应用程序在持有锁
    - 锁等待或锁超时的数目，锁等待花费的时间
  - 锁的信息
    - 应用程序持有锁的数目
    - 应用程序持有锁的列表
    - 锁的名称，类型，模式，对象以及所有的详细信息

db2pd工具

特点

- ✓ 不浪费引擎的处理能力
- ✓ 直接读取内存，速度快

不同选项

- ✓ Db2pd –locks：捕获锁的详细信息
- ✓ Db2pd -wlocks ：捕获锁等待信息
- ✓ Db2pd -apinfo <applhandle> ：捕获锁拥有者和等待着详细运行时信息

db2pd –locks 举例

打开一个会话

– 设置隔离级别为RS，随后发起SELECT语句（不提交）

```
hotel16:/home/hotel16/qchong> db2 set isolation level RS
DB20000I  The SQL command completed successfully.
hotel16:/home/hotel16/qchong> db2 +c "select * from t1 where i1 > 4"

I1
-----------
          5
          6
          7
          8
          9

  5 record(s) selected.
```

db2pd –locks 举例

打开另外一个会话，使用db2pd -db wsdb -locks -file lock1.out ， 锁的详细信息如下：

Database Partition 0 -- Database WSDB -- Active -- Up 0 days 00:02:46

Locks:

Address	TranHdl	Lockname	Type	Mode	Sts	Owner	Dur	HoldCount	Att	ReleaseFlg
0x0000002AC1899750	2	020004000D0080000000000052	Row	.NS	G	2	1	0	0x00	0x00000001
0x0000002AC18997E0	2	020004000C0080000000000052	Row	.NS	G	2	1	0	0x00	0x00000001
0x0000002AC1899360	2	020004000B0080000000000052	Row	.NS	G	2	1	0	0x00	0x00000001
0x0000002AC1899480	2	020004000A0080000000000052	Row	.NS	G	2	1	0	0x00	0x00000001
0x0000002AC1899630	2	020004000900800000000000052	Row	.NS	G	2	1	0	0x00	0x00000001
0x0000002AC18995A0	2	53514C43324731350655EBAA41	Internal P	..S	G	2	1	0	0x00	0x40000000
0x0000002AC1899870	2	020004000000000000000000054	Table	IS	G	2	1	0	0x00	0x000000018

PageNum 128
SlotNum 13

PoolID 2
ObjectID 4

Table Lock
Lock Mode IS

Lock
Granted

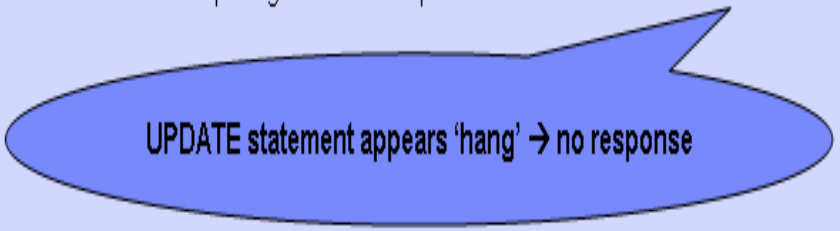
锁快照-举例

Session 1

```
hotel16:/home/hotel16/qchong> db2 set isolation level RS
DB20000I  The SQL command completed successfully.
hotel16:/home/hotel16/qchong> db2 +c "select * from t1 where i1 > 4"

I1
----------
         5
         6
         7
         9
        10

 5 record(s) selected.
```

Session 2

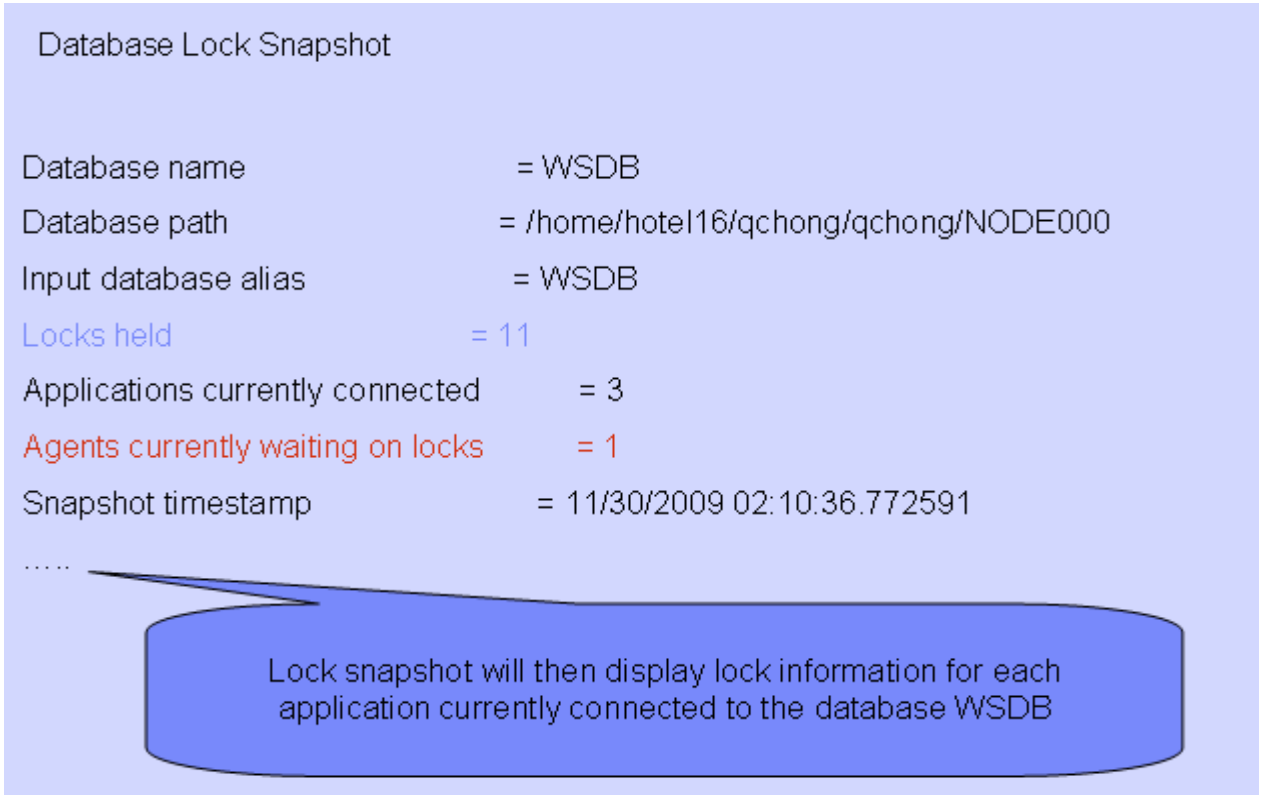
```
hotel16:/home/hotel16/qchong/work> db2 connect to wsdb

   Database Connection Information

 Database server        = DB2/LINUXX8664 9.5.6
 SQL authorization ID   = QCHONG
 Local database alias   = WSDB

hotel16:/home/hotel16/qchong/work> db2 "update t1 set i1=i1-1 where i1 > 7"
```



锁快照-分析（1）

1，UPDATE MONITOR SWITCHES USING LOCK ON

2，GET SNAPSHOT FOR LOCKS ON WSDB



锁快照-分析（2）

```
Application handle                    = 14
Application ID                        = *LOCAL.qchong.091130070851
Sequence number                       = 00002
Application name                      = db2bp
CONNECT Authorization ID              = QCHONG
Application status                    = Lock-wait
Status change time                    = Not Collected
Application code page                 = 1208
Locks held                            = 4
Total wait time (ms)                  = 25333
  ID of agent holding lock            = 7
  Application ID holding lock         = *LOCAL.qchong.091130070659
  Lock name                           = 0x020004000C0080000000000052
  Lock attributes                     = 0x00000000
  Release flags                       = 0x40000000
  Lock object type                    = Row
  Lock mode                           = Next Key Share (NS)
  Lock mode held                      = Update Lock (U)
  Lock mode requested                 = Exclusive Lock (X)
  Name of tablespace holding lock     = USERSPACE1
  Schema of table holding lock        = QCHONG
  Name of table holding lock          = T1
  Data Partition Id of table holding lock = 0
  Lock wait start timestamp           = 11/30/2009 02:10:11.438770
..... Cont'
```

```
(Cont'd from left)

List Of Locks
  Lock Name                 = 0x020004000C0080000000000052
  Lock Attributes           = 0x00000000
  Release Flags             = 0x40000000
  Lock Count                = 1
  Hold Count                = 0
  Lock Object Name          = 8388620
  Object Type               = Row
  Tablespace Name           = USERSPACE1
  Table Schema              = QCHONG
  Table Name                = T1
  Mode                      = X
  Status                    = Converting
  Current Mode              = U

  Lock Name                 = 0x010000000100000001004700 56
  Lock Attributes           = 0x00000000
.....
```

锁快照-分析（3）

```
Application handle                    = 7
Application ID                        = *LOCAL.qchong.091130070659
Sequence number                       = 00004
Application name                      = db2bp
CONNECT Authorization ID              = QCHONG
Application status                    = UOW Waiting
Status change time                    = Not Collected
Application code page                 = 1208
Locks held                            = 7
Total wait time (ms)                  = 0

List Of Locks
 Lock Name                  = 0x020004000D0080000000000052
 Lock Attributes            = 0x00000000
 Release Flags              = 0x00000001
 Lock Count                 = 1
 Hold Count                 = 0
 Lock Object Name           = 8388621
 Object Type                = Row
 Tablespace Name            = USERSPACE1
 Table Schema               = QCHONG
 Table Name                 = T1
 Mode                       = NS
....
```

The rest list of locks currently held by this application (Appl handle as 7)

事件监控器

创建事件监控器

- 存入表：CREATE EVENT MONITOR <lockevmon> FOR LOCKING write to table
- 存入文件：CREATE EVENT MONITOR <lockevmon> FOR LOCKING write to file /home/db2inst1/eventmonitor

激活事件监控器

- SET EVENT MONITOR <lockevmon> STATE 1

收集信息

- 如果存入表，从STMT_LOCKEVMON中查询相关信息
- 如果存入文件，使用命令db2evmon -path '/home/db2inst1/eventmonitor' 格式化文件，分析相关信息

关闭事件监控器

- SET EVENT MONITOR <lockevmon> STATE 0

事件监控器-举例（1）

Session 1

```
hotel16:/home/hotel16/qchong> db2 set isolation level RS
DB20000I  The SQL command completed successfully.
hotel16:/home/hotel16/qchong> db2 +c "select * from t1 where i1 = 15"

I1
-----------
         15

  1 record(s) selected.

hotel16:/home/hotel16/qchong> db2 +c "update t2 set i1 =i1+1 where i1 =15"
DB20000I  The SQL command completed successfully.
```

Session 2

```
hotel55:/home/hotel55/qchong> db2 set isolation level to RS
DB20000I  The SQL command completed successfully.
hotel16:/home/hotel16/qchong> db2 +c "select * from t2 where i1 = 15"

I1
-----------
         15

  1 record(s) selected.

hotel16:/home/hotel16/qchong> db2 +c "update t1 set i1 = i1+1 where i1 = 15"
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0911N  The current transaction has been rolled back because of a deadlock
or timeout.  Reason code "2".  SQLSTATE=40001
```

事件监控器-举例（2）

```
Reading /home/hotel16/qchong/DLOCKS/00000000.evt ...
------------------------------------------------------------------
                    EVENT LOG HEADER
  Event Monitor name: MYEMON
  Server Product ID: SQL09056
  Version of event monitor data: 9
  Byte order: LITTLE ENDIAN
  Number of nodes in db2 instance: 1
  Codepage of database: 1208
  Territory code of database: 1
  Server instance name: qchong
------------------------------------------------------------------


------------------------------------------------------------------
  Database Name: WSDB
  Database Path: /home/hotel16/qchong/qchong/NODE0000/SQL00001/
  First connection timestamp: 12/01/2009 02:00:05.315640
  Event Monitor Start time:   12/01/2009 02:01:13.510925
------------------------------------------------------------------

3) Deadlock Event ...
  Deadlock ID:   1
 Number of applications deadlocked: 2
  Deadlock detection time: 12/01/2009 02:06:55.852349
  Rolled back Appl participant no: 2
  Rolled back Appl Id: *LOCAL.qchong.091201070440
  Rolled back Appl seq number: : 0003
```

```
....(Cont')

5) Deadlocked Connection ...
  Deadlock ID:   1
  Participant no.: 2
  Participant no. holding the lock: 1
  Appl Id: *LOCAL.qchong.091201070440
  Appl Seq number: 00003
  Appl Id of connection holding the lock: *LOCAL.qchong.091201070435
  Seq. no. of connection holding the lock: 00001
  Lock wait start time: 12/01/2009 02:06:27.555323
  Lock Name      : 0x05000400120080000000000052
  Lock Attributes : 0x00000000
  Release Flags   : 0x40000000
  Lock Count      : 1
  Hold Count      : 0
  Current Mode    : U   - Update
  Deadlock detection time: 12/01/2009 02:06:55.852597
  Table of lock waited on      : T1
  Schema of lock waited on     : QCHONG
  Data partition id for table  : 0
  Tablespace of lock waited on : TS1
  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: 8388626
  Application Handle: 15
```

事件监控器-举例（3）

```
Deadlocked Statement:
   Type     : Dynamic
   Operation: Execute Immediate
   Section  : 203
   Creator  : NULLID
   Package  : SQLC2G15
   Cursor   :
   Cursor was blocking: FALSE
   Text     : update t1 set i1 = i1+1 where i1 = 15
  List of Locks:
Lock Name                 : 0x05000400120080000000000052
     Lock Attributes      : 0x00000000
     Release Flags        : 0x40000000
.....
7) Deadlocked Connection ...
  Deadlock ID:   1
  Participant no.: 1
  Participant no. holding the lock: 2
  Appl Id: *LOCAL.qchong.091201070435
  Appl Seq number: 00002
  Appl Id of connection holding the lock: *LOCAL.qchong.091201070440
  Seq. no. of connection holding the lock: 00001
Lock wait start time: 12/01/2009 02:06:47.075589
Lock Name      : 0x050005001200C0000000000052
  Lock Attributes : 0x00000000
  Release Flags   : 0x40000000
  Lock Count      : 1
```

```
(Cont'd)
Hold Count      : 0
  Current Mode    : U   - Update
Deadlock detection time: 12/01/2009 02:06:55.853211
  Table of lock waited on      : T2
  Schema of lock waited on     : QCHONG
  Data partition id for table  : 0
  Tablespace of lock waited on : TS1
 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: 12582930
  Application Handle: 14
  Deadlocked Statement:
    Type     : Dynamic
    Operation: Execute Immediate
    Section  : 203
    Creator  : NULLID
    Package  : SQLC2G15
    Cursor   :
    Cursor was blocking: FALSE
    Text     : update t2 set i1 =i1+1 where i1 =15
  List of Locks:
     ...
```

ADMIN消息（管理日志和 db2diag.log）

设定MON_LCK_MSG_LVL配置（DB CFG）

- 0 – No notification logging
- 1 – Only log notification for lock escalation (default)
- 2 – Log notification for lock escalation and deadlock
- 3 – Log notification for lock escalation, deadlock and lock timeout

日志-锁现象举例

ADMIN消息

锁升级消息W – Warning message，I – Informational message， E – Error message

ADM5500W DB2 is performing lock escalation. The total number of locks currently held is "<locksHeld>", and the target number of locks to hold is "<targetNumber>".

ADM5501I DB2 is performing lock escalation. The total number of locks currently held is "<locksHeld>", and the target number of locks to hold is "<targetNumber>". The current statement being executed is "<currentStatement>".

ADM5502W The escalation of "<numLocks>" locks on table "<tableName>" to lock intent "<lockIntent>" was successful.

ADM5503E The escalation of "<numLocks>" locks on table "<tableName>" to lock intent "<lockIntent>" has failed. The SQLCODE is "<SQLCODE>".

ADM5504W The escalation of "<numLocks>" locks on DATAPARTITIONID "<datapartitionid>" of table "<tableName>" to lock intent "<lockIntent>" was successful.

ADM5505E The escalation of "<numLocks>" locks on DATAPARTITIONID "<datapartitionid>" of table <tableName>" to lock intent "<lockIntent>" has failed. The SQLCODE is "<SQLCODE>".

DB2和Oracle锁机制对比

1、Oracle对数据的读操作是不加锁的。DB2强调“读一致性”，在读数据时，会根据隔离级别的不同而加锁，只有在使用UR隔离级别时才不加锁，这保证了不同应用程序和用户读取的数据是一致的。Oracle同一时刻不同的应用程序有读不一致的现象，而DB2在同一时刻所有的应用程序读到的数据都是一致的。

2、Oracle不存在锁升级，DB2会发生锁升级。

3、在UR隔离级别， DB2读最新的没有提交的脏数据，Oracle是读曾经提交过的数据。在一些应用里，需要读到最新的数据，也就是脏数据。

4、Oracle利用意向锁及数据行上加锁标志位来实现锁机制，DB2中的每个锁都记录在内存中。当请求加锁时，DB2会检查锁列表，看锁对象上是否已加锁，请求锁和现有锁是否兼容。

5、DB2在做数据修改时，在日志中既记录了修改前数据（也就是UNDO日志），也记录了修改后的数据（即REDO日志）。而Oracle只在日志中记录修改后数据，而将修改前的数据记录到回滚段中。

6，在Oracle中，当一个应用程序对表以Insert、Update和Delete操作进行修改时，另外一个应用程序在读取该表时，会从回滚段中读取该表修改前的数据。而对于DB2 V9.7之前的版本，读取数据的应用程序，遇到正在被其他应用程序修改的数据时，将会进行锁等待（除非使用UR隔离级别）。对于DB2 V9.7及以后版本，由于引入了当前已落实，读取数据的应用程序将不需要等待锁释放，而是会从日志中读取数据修改前的版本（即已落实的版本，相当于Oracle回滚段中的数据）。

设计高并发应用（1）

1、使事务尽可能地短小。这可以通过在您的程序逻辑允许的情况下频繁地使用COMMIT语句（即使对只读的事务也一样）来实现。

2、有序访问对象以减少死锁的发生。

3、在SELECT中使用FOR FETCH ONLY从句，因为只会获得NS共享锁。

4、如果游标是可更新的，使用FOR UPDATE从句以最小化锁转换和死锁的发生。

5、在一个工作单元的结束执行INSERT, UPDATE, DELETE, 能缩小持有X锁的时间。

6、使用CLOSE CURSOR语句的WITH RELEASE选项，这减少了锁的持有时间。

7、对只读表使用ALTER TABLE ... LOCKSIZE TABLE，锁粒度改变为表。

8、以EXCLUSIVE模式使用LOCK TABLE语句务必谨慎。

9、调整参数LOCKLIST和MAXLOCKS，以最小化锁升级的发生

设计高并发应用（2）

10、只有必要时才记录事务信息。

11、快速清理数据：
ALTER TABLE ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE

12、利用DB2数据移动工具中的并行特性。

13、设置数据库级的LOCKTIMEOUT参数（推荐值为30~120秒之间）。不要保留为默认值-1。

14、不要检索不需要的数据。例如在SELECT语句中使用FETCH FIRST n ROWS ONLY 子句。

15、OLTP环境下因高吞吐率和高事务，使用CS隔离级，并频繁提交以减少锁的持续时间。
16、 BI环境下因吞吐率低和主要以只读为主，使用CS或UR隔离级。

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间