Oracle Database Locking Mechanism Demystified

Pini Dibask, Product Manager for Database Solutions
November 17th 2016
About Me

• Pini Dibask, Product Manager, Database Solutions (Quest)
• Oracle DBA since 2006
• Oracle Certified Professional DBA (OCP)
• My Blog: OracleDBPro.BlogSpot.com

Pini.Dibask@Quest.com
http://Linkedin.com/in/pinidibask
@pini_dibask
About Quest

• Quest is now an independent company again!

• **Simplifies** IT management

• #1 independent software company for Database Tools

• Driven by innovation

  “*Spend less time on what you need to do, and more time on what you want to do!*”

• Committed to providing great products and superior support
Agenda

- Overview of Locks in Database Management Systems
- Oracle Database Locking Mechanism Concepts
- Advanced Locking Scenarios
- Monitoring Locks using Dictionary Views and Tools
Overview of **Locks** in Database Management Systems
Overview of Database Locks

• Why locks? Because Databases need to support multiple user applications
• Used to ensure Database Consistency and Integrity
• Affect the interaction of readers and writers
• Every DBMS has its own implementation of locking mechanism
Oracle Database
Locking Mechanism Concepts
High Level Overview

- Reader **never** blocks reader
  (except: SELECT .. FOR UPDATE)
- Reader **never** blocks writer
- Writer **never** blocks reader
  (except rare scenario of distributed transaction)
- Writer **might** block writer
  (depends on the operation)
Writer Never Blocks Reader - How?

• Oracle blocks can be modified during execution of DMLs
• Undo Tablespace holds “before” image of Database Blocks
• During SELECT query Oracle reads undo images if needed
• This provides 2 important features:
  • Non-Blocking Queries
  • Read Consistency
Lock Modes

• 2 Lock Modes in general:
  • **Share Lock** – Many can be acquired on a resource
  • **Exclusive Lock** – Only one can be acquired on a resource

• Example - User updates a row in table EMPLOYEES
  • **Row** will be locked in **exclusive** mode
  • **Table** will be locked in **share** mode
DML Locks

• **When?** During DML statements:
  
  *Insert, update, delete, merge, select ... for update*

• DML operations acquire 2 lock types:
  
  • **Row Locks** (AKA “TX Lock”)
  
  • **Table Locks** (AKA “TM Lock”)

Confidential
DML Row Locks ("TX Locks")

- Oracle uses row-level locking during DML operations
- Modified rows will be locked in exclusive lock mode
- Oracle stores lock information in the containing data block header
- **No overhead** with Oracle row-level locking mechanism
# DML Row Locks (“TX Locks”) - Demo

## Session #1

- SQL> CREATE TABLE employees (id NUMBER, name VARCHAR2 (20));
  
  **Table created.**

- SQL> INSERT INTO employees VALUES (1, 'David');
  
  **1 row created.**

- SQL> INSERT INTO employees VALUES (2, 'Jason');
  
  **1 row created.**

- SQL> commit;
  
  **Commit complete.**

- SQL> UPDATE employees SET name = 'Peter' WHERE id = 2;
  
  **1 row updated.**

## Session #2

- SQL> UPDATE employees SET name = 'Mark' WHERE id = 1;
  
  **1 row updated.**

- SQL> UPDATE employees SET name = 'John' WHERE id = 2;
  
  (waiting – session is blocked)

---

Row already locked by session #1
Table Locks ("TM Locks") Cont’d

- Oracle automatically locks tables (share mode) involved in DML operations
- Prevent DDL operations which may conflict with running transactions

Table EMPLOYEES

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>EMAIL</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
<th>MANAGER_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>King</td>
<td>SKING</td>
<td>17-JUN-87</td>
<td>AD_PRES</td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>NKOCCHAR</td>
<td>21-SEP-89</td>
<td>AD_VP</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>102</td>
<td>De Hann</td>
<td>LDEHANN</td>
<td>13-JAN-93</td>
<td>AD_VP</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>103</td>
<td>Hunold</td>
<td>AHUNOLD</td>
<td>03-JAN-90</td>
<td>IT_PROG</td>
<td>102</td>
<td>60</td>
</tr>
</tbody>
</table>

Table lock acquired  Exclusive row lock (TX) acquired  Row being updated
**Table Locks (“TM Locks”)**

- There are 6 lock modes (LMODE and REQUEST columns in V$LOCK)
  - [0, 1] – No lock
  - 2 – (RS) Row Share
  - 3 – (RX) Row Exclusive **(DML Operations)**
  - 4 – (S) Share
  - 5 – (SSX) Share Row Exclusive
  - 6 – (X) Exclusive **(DDL operations)**

<table>
<thead>
<tr>
<th>TM Blocking Matrix</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>Block</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Block</td>
<td>Block</td>
<td>Block</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Block</td>
<td></td>
<td>Block</td>
<td>Block</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Block</td>
<td>Block</td>
<td>Block</td>
<td>Block</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Block</td>
<td>Block</td>
<td>Block</td>
<td>Block</td>
<td>Block</td>
</tr>
</tbody>
</table>
DML Locks - Demo

(Session ID #383)

Row of employee_id #139 is locked in LMODE 6 - Exclusive (x)

Table EMPLOYEE is locked in LMODE 3 - Row Exclusive Table Lock (RX)

SQL> UPDATE employee
2     SET last_name = 'Jones'
3   WHERE employee_id = 139;
1 row updated.

SQL> SELECT type, lmode
2   FROM v$lock
3   WHERE sid = 383;

<table>
<thead>
<tr>
<th>TYPE</th>
<th>LMODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX</td>
<td>6</td>
</tr>
<tr>
<td>TM</td>
<td>3</td>
</tr>
</tbody>
</table>

SQL> SELECT object_name, session_id, oracle_username, locked_mode
2   FROM v$locked_object JOIN dba_objects USING (object_id);

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>SESSION_ID</th>
<th>ORACLE_USERNAME</th>
<th>LOCKED_MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>383</td>
<td>SALES</td>
<td>3</td>
</tr>
</tbody>
</table>
DDL Locks

• Protect definition of schema objects during DDL statements

• **Exclusive Locks (LMODE = 6)**
  • Most DDL Operations (e.g. ALTER TABLE, DROP TABLE)
  • Other sessions cannot execute DML or DDL on the object

• **Share Locks**
  • Allow data concurrency for similar operations

• Only modified objects are locked - Oracle never locks entire Data Dictionary
**DDL Locks Cont’d**

**Problem**
- DDL commands (e.g. ALTER TABLE, DROP TABLE) require exclusive locks (LMODE = 6)
- It’s hard to acquire exclusive lock on frequently accessed object

**Solution**
- DDL_LOCK_TIMEOUT parameter (available from Oracle 11g)
  - Specifies time limit for how long DDL statements will wait in DML lock queue
  - Default value is 0
  - Can be set at session level (ALTER SESSION) or instance level (ALTER SYSTEM)
# DDL Locks - Demo

<table>
<thead>
<tr>
<th>Session #1</th>
<th>Session #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL&gt; UPDATE employee</td>
<td>SQL&gt; drop table employee; drop table employee</td>
</tr>
<tr>
<td>SET last_name = 'Jones'</td>
<td>*</td>
</tr>
<tr>
<td>WHERE employee_id = 139;</td>
<td>ERROR at line 1: ORA-00054: resource busy and acquire with NOWAIT</td>
</tr>
<tr>
<td>1 row updated.</td>
<td>specified or timeout expired</td>
</tr>
<tr>
<td></td>
<td><strong>Table already locked in share mode</strong></td>
</tr>
<tr>
<td>SQL&gt; alter system set ddl_lock_timeout=10;</td>
<td>SQL&gt; drop table employee; drop table employee</td>
</tr>
<tr>
<td><strong>System altered.</strong></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>ERROR at line 1: ORA-00054: resource busy and acquire with NOWAIT</td>
</tr>
<tr>
<td></td>
<td>specified or timeout expired</td>
</tr>
</tbody>
</table>

- Available from version 11g
- Oracle waits 10 seconds before raising this error
How Oracle Manages Locks?

• Oracle **automatically** manages locks
• Locks will be **released** once transaction is over
• **Lowest** lock level will be used for maximum concurrency
• Oracle Database **never** escalates locks
• Ordering of locks is based on **FIFO** (First-In-First-Out)
### FIFO Lock Ordering - Demo

<table>
<thead>
<tr>
<th>Session #1</th>
<th>Session #2</th>
<th>Session #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL&gt; SELECT * FROM employees;</td>
<td>SQL&gt; LOCK TABLE employees IN EXCLUSIVE MODE;</td>
<td>SQL&gt; UPDATE employees</td>
</tr>
<tr>
<td>EMP_ID</td>
<td>NAME</td>
<td>DEPT_ID</td>
</tr>
<tr>
<td>1</td>
<td>David</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>4</td>
</tr>
</tbody>
</table>

1 row updated.
Manual Data Locks

• It is possible to override Oracle’s default locking mechanisms

• Should be avoided unless there is a justified application requirement

  Example: Transaction needs exclusive access to resource and must not wait for other transactions

• Manual row-level locks: **SELECT ... FOR UPDATE** statement

• Manual table-level locks: **LOCK TABLE** statement
## Manual Row Locks - Demo

<table>
<thead>
<tr>
<th>Session #1</th>
<th>Session #2</th>
</tr>
</thead>
</table>
| SQL> SELECT id, name  
FROM employees  
WHERE id = 2 FOR UPDATE; | SQL> UPDATE employees SET name = 'Mark' WHERE id = 1;  
1 row updated. |
| **ID** | **NAME** | **ID** | **NAME** |
| 2 Jason | | | |
| **ID** | **NAME** |
| 2 Jason | | | (waiting – session is blocked) |
# Manual Table Locks - Demo

- **LOCK TABLE IN** [ ROW SHARE | ROW EXCLUSIVE | SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE ] **MODE**

<table>
<thead>
<tr>
<th>Session #1 (SID 385)</th>
<th>Session #2 (SID 195)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL&gt; SELECT * FROM employees FOR UPDATE;</td>
<td>SQL&gt; LOCK TABLE employees IN ROW SHARE MODE;</td>
</tr>
<tr>
<td>ID NAME</td>
<td>Table(s) Locked.</td>
</tr>
<tr>
<td>2 Jason</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SQL&gt; LOCK TABLE employees IN EXCLUSIVE MODE;</td>
</tr>
<tr>
<td></td>
<td>(waiting - session is blocked)</td>
</tr>
</tbody>
</table>

SQL> SELECT sid, lmode acquired, request, blocking_session, SQL_TEXT FROM v$lock l JOIN v$session s USING (sid) LEFT JOIN v$sqlarea USING (sql_id) WHERE block = 1 OR request > 0 ;

<table>
<thead>
<tr>
<th>SID</th>
<th>ACQUIRED</th>
<th>REQUEST</th>
<th>BLOCKING_SESSION</th>
<th>SQL_TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>385</td>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>195</td>
<td>2</td>
<td>6</td>
<td>385</td>
<td>LOCK TABLE employees in exclusive mode</td>
</tr>
</tbody>
</table>
Advanced Locking Scenarios
Deadlocks

- 2 sessions, each one locking resource that other session wants
- At this stage both could become blocked forever
- Oracle automatically detects deadlock scenarios
- One of 2 sessions will be “Deadlock Victim”
- Oracle performs statement-level rollback
# Deadlocks - Demo

<table>
<thead>
<tr>
<th>Session #1</th>
<th>Session #2</th>
</tr>
</thead>
</table>
| SQL> UPDATE employee  SET first_name = 'Mark'  WHERE employee_id = 39;  
1 row updated. | SQL> UPDATE employee  SET first_name = 'David'  WHERE employee_id = 151;  
1 row updated. |
| SQL> UPDATE employee  SET first_name = 'Mark'  WHERE employee_id = 39;  
ORA-00060: deadlock detected while waiting for resource | SQL> UPDATE employee  SET first_name = 'Greg'  WHERE employee_id = 39;  
1 row updated. |
| Statement has been rolled-back | (Waiting - session is blocked)  
Row already locked by session #1 |
Deadlocks Cont’d

• Deadlock error (ORA-00060) is also audited in **Alert Log**

ORA-00060: Deadlock detected. More info in file C:\APP\ORACLE\diag\rdbms\orcl11\orcl11\trace\orcl11_ora_3600.trc.

**orcl11_ora_3600.trc**

```plaintext
*** 2016-11-06 13:37:39.317
DEADLOCK DETECTED (ORA-00060 )

[Transaction Deadlock]

The following deadlock is not an ORACLE error. It is a
deadlock due to user error in the design of an application
or from issuing incorrect ad-hoc SQL. The following
information may aid in determining the deadlock:

**Deadlock graph:**

| Resource Name | --------|--------|------------|--------|--------|
| TX-0006001f-0001eC1d | 28     | 767    | X          | 51     | 584    | X        |
| TX-00050012-0001e26c | 51     | 584    | X          | 28     | 767    | X        |

session 767: DID 0001-001C-0000494f session 584: DID 0001-0033-000006d9
session 584: DID 0001-0033-000006d9 session 767: DID 0001-001C-0000494f
```
Deadlocks Cont’d

orcl11_oracle_3600.trc (cont’d)

----- Information for the OTHER waiting sessions -----  
Session 584:
  sid: 584  ser: 56305  cwsid: 507983025  user: 100/SALES  
  flags: (0x10004)  USR/-  flags_idl: (0x1)  BSY/-/-/-/-/-/-  
  flags2: (0x40009)  -/-/INC  
pid: 51  O/S info: user: SYSTEM, term: ISRVMPALW06, ospid: 17368  
  image: ORACLE.EXE (SHAD)  
  client details:  
    O/S info: user: PROD\pdibask, term: DSG5P2DG72, ospid: 1740:4440  
    machine: PROD\DSG5P2DG72 program: sqlplus.exe  
    application name: SQL*Plus, hash valuc=3669545024  
current SQL:  
  UPDATE employee  
  SET first_name = 'John'  
  WHERE employee_id = 151

----- End of information for the OTHER waiting sessions -----  

Information for THIS session:  

----- Current SQL Statement for this session (sql_id-a2vzskvuhvtb) -----  
UPDATE employee  
  SET first_name = 'Mark'  
  WHERE employee_id = 39
Blocked Inserts

• Most common scenario of blocked inserts:
  • 2 sessions insert same value for column that has unique or primary key

• Another scenario that involves tables with foreign keys
  • Row inserted/deleted on the parent table
  • Row inserted to the child table - may be blocked
## Blocked Inserts - Demo

<table>
<thead>
<tr>
<th>Session #1</th>
<th>Session #2</th>
</tr>
</thead>
</table>
| SQL> CREATE TABLE employees  
  (  
    id   NUMBER,  
    name VARCHAR2 (20),  
    CONSTRAINT pk_id PRIMARY KEY (id)  
  );  

Table created.  
SQL> insert into employees values (1, 'John');  
1 row created.  
SQL> commit;  
Commit complete.  
| SQL> insert into employees values (1, 'David');  
(waiting – row already locked by session #1)  
|  
| Star  
| ERROR at line 1:  
ORA-00001:  
unique constraint (SALES.PK_ID) violated |
### Blocked Inserts - Demo Cont’d

<table>
<thead>
<tr>
<th>Session #1</th>
<th>Session #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL&gt; CREATE TABLE employees</td>
<td>SQL&gt; insert into employees values (1, 'David');</td>
</tr>
<tr>
<td>(</td>
<td>(waiting – row already locked by session #1)</td>
</tr>
<tr>
<td>id     NUMBER,</td>
<td></td>
</tr>
<tr>
<td>name   VARCHAR2 (20),</td>
<td></td>
</tr>
<tr>
<td>CONSTRAINT pk_id PRIMARY KEY (id)</td>
<td></td>
</tr>
<tr>
<td>);</td>
<td></td>
</tr>
<tr>
<td><strong>Table created.</strong></td>
<td></td>
</tr>
<tr>
<td>SQL&gt; insert into employees values (1, 'John');</td>
<td></td>
</tr>
<tr>
<td><strong>1 row created.</strong></td>
<td></td>
</tr>
<tr>
<td>SQL&gt; rollback;</td>
<td></td>
</tr>
<tr>
<td><strong>Rollback complete.</strong></td>
<td></td>
</tr>
</tbody>
</table>
Unindexed Foreign Keys

• Oracle Database places **full table lock** (LMODE = 4) on child table when:
  • Unindexed foreign key column on child table
  • Session updates parent table’s primary key
  • Session deletes row from parent table

• Increases probability for deadlocks

• **Best practice** - foreign key columns should be indexed
  • Exception - Matching primary key or unique key never updated or deleted
### Unindexed Foreign Keys Cont’d

**Parent Key** Primary key of referenced table

Table **DEPARTMENTS** (Referenced or Parent Table)

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>DEPARTMENT_NAME</th>
<th>MANAGER_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>IT</td>
<td>103</td>
<td>1400</td>
</tr>
<tr>
<td>90</td>
<td>Executive</td>
<td>100</td>
<td>1700</td>
</tr>
</tbody>
</table>

Session updates value of primary key

EMPLOYEES table is locked

Full table lock acquired

Table **EMPLOYEES** (Dependent Child Table)

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>EMAIL</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
<th>MANAGER_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>IT</td>
<td>SKING</td>
<td>17-JUN-87</td>
<td>AD_PRES</td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>101</td>
<td>Executive</td>
<td>NKOCHHAR</td>
<td>21-SEP-89</td>
<td>AD_VP</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>102</td>
<td>De Hann</td>
<td>LDEHANN</td>
<td>13-JAN-93</td>
<td>AD_VP</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>103</td>
<td>Hunold</td>
<td>AHUNOLD</td>
<td>03-JAN-90</td>
<td>IT_PROG</td>
<td>102</td>
<td>60</td>
</tr>
</tbody>
</table>

Unindexed Foreign Key

Full table lock acquired

Exclusive row lock (TX) acquired

Primary key modified
## Indexed Foreign Keys

**Parent Key** Primary key of referenced table

### Table DEPARTMENTS (Referenced or Parent Table)

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>DEPARTMENT_NAME</th>
<th>MANAGER_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>IT</td>
<td>103</td>
<td>1400</td>
</tr>
<tr>
<td>90</td>
<td>Executive</td>
<td>100</td>
<td>1700</td>
</tr>
<tr>
<td>280</td>
<td>Event Planning</td>
<td></td>
<td>1700</td>
</tr>
</tbody>
</table>

Session deletes a row

### Table EMPLOYEES (Dependent Child Table)

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>EMAIL</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
<th>MANAGER_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>King</td>
<td>SKING</td>
<td>17-JUN-87</td>
<td>AD_PRES</td>
<td>103</td>
<td>90</td>
</tr>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>NKOCCHAR</td>
<td>21-SEP-89</td>
<td>AD_VP</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>102</td>
<td>De Hann</td>
<td>LDEHANN</td>
<td>13-JAN-93</td>
<td>AD_VP</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>103</td>
<td>Hunold</td>
<td>AHUNOLD</td>
<td>03-JAN-90</td>
<td>IT_PROG</td>
<td>102</td>
<td>60</td>
</tr>
</tbody>
</table>

Indexed Foreign Key

DMLs are allowed on EMPLOYEES

Full table lock acquired
Exclusive row lock (TX) acquired
Row being deleted
Monitoring
Monitoring locks via Oracle Dictionary Views

- **V$SESSION** - Lists session information for each current session
- **V$LOCK/DBA_LOCK** - Lists all locks currently held and all requests for a lock
- **V$LOCKED_OBJECT** - Lists sessions holding locks on what objects and in what mode
- **DBA_BLOCKERS** - Lists sessions holding a lock that blocks another session
- **DBA_WAITERS** - Lists sessions that are waiting for a lock
Monitoring locks via Oracle Dictionary Views Cont’d

SQL> SELECT DECODE (blocking_session, null, null, 'BLOCKED') status,
    4  sid,
    4  lmode,
    4  request,
    4  ctime duration,
    4  USER,
    4  program,
    4  blocking_session,
    4  DECODE (request, 0, NULL, SQL_TEXT) SQL_TEXT
FROM v$lock l
    JOIN v$session s USING (sid)
    LEFT JOIN v$sqlarea USING (sql_id)
WHERE block = 1 OR request > 0
ORDER BY status

<table>
<thead>
<tr>
<th>STATUS</th>
<th>SID</th>
<th>LMODE</th>
<th>REQUEST</th>
<th>DURATION</th>
<th>USER</th>
<th>PROGRAM</th>
<th>BLOCKING_SESSION</th>
<th>SQL_TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKED</td>
<td>195</td>
<td>4</td>
<td>5</td>
<td>16581</td>
<td>SALES</td>
<td>sqlplus.exe</td>
<td>385</td>
<td>lock table employees in share row exclusive mode</td>
</tr>
<tr>
<td>BLOCKED</td>
<td>13</td>
<td>0</td>
<td>3</td>
<td>10129</td>
<td>SALES</td>
<td>Toad.exe</td>
<td>385</td>
<td>insert into employees values (1, 'Jason')</td>
</tr>
<tr>
<td></td>
<td>385</td>
<td>4</td>
<td>0</td>
<td>16575</td>
<td>SALES</td>
<td>sqlplus.exe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Monitoring Blocked Sessions via Tools
Monitoring Blocked Sessions via Tools
Monitoring Blocked Sessions via Tools
Monitoring Blocked Sessions via Tools
Monitoring Blocked Sessions via Tools
Monitoring Blocked Sessions via Tools
Summary

• Is lock a bad thing? **No! Locks are essential!**

• Hold locks as long as you need, but not more than you need

• Avoid Manual Locking unless it is justified

• Foreign keys in most cases should be indexed

• Proactively monitor your Database to identify blocked sessions

• Modify application code if needed
Q&A