Chapter 9 & 10  Part2
Oracle 12c Database Data Concurrency : Transactions and Locking
B + tree

By David Itshak
shaked19@gmail.com

http://www.ildba.co.il/author/cimid/
http://www.sqlserver.co.il/?cat=940

Global Hebrew Virtual PASS Chapter :
https://www.youtube.com/watch?v=x4hGjYGbfkc
https://www.youtube.com/watch?v=eJO8G9if3EY

Sqlsatuday Israel 2016 :
Reference and Credits

**Oracle® Database** Concepts
12c Release 1 (12.1)
E41396-13
https://docs.oracle.com/database/121/CNCPT/toc.htm

**Oracle® Database** Performance Tuning Guide
12c Release 1 (12.1)
E49058-06
https://docs.oracle.com/database/121/TGDBA/toc.htm

**Oracle® Database SQL Language Reference**
12c Release 1 (12.1)
E41329-20

Oracle Essentials(Oracle Database 12c), 5th; O'Reilly, 2013

Oracle OCA Oracle Database 12c Administrator Certified Associate Study Guide Exam

Pro Oracle Database 12c Administration, 2 edition ISBN 1430257288 2013

Apress Oracle Database Transactions and Locking Revealed (2014)

Oracle Learning Library

Pro SQL Server Internals 2014  Apress
SQL Server Concurrency   Locking, Blocking and Row Versioning   By Kalen Delaney
SQL 2016 Book Online
Part II: Concurrency Control

- 3 Concurrency Control: Notions of Correctness for the Page Model
- 4 Concurrency Control Algorithms
- 5 Multiversion Concurrency Control
- 6 Concurrency Control on Objects: Notions of Correctness
- 7 Concurrency Control Algorithms on Objects
- 8 Concurrency Control on Relational Databases
- 9 Concurrency Control on Search Structures
- 10 Implementation and Pragmatic Issues
Latches

• However, latches do not show up in the sys.dm_tran_locks view.

• Latches are used to protect an internal structure for brief periods while it is being read or modified, not to ensure correct transaction behavior.

• Both the data page itself and the buffer that the data is occupying are protected by latches.
• Latches protect the physical integrity of the data; locks protect its logical integrity.
Controlling Locking

• Changing the transaction isolation level (the most common method).
• Changing the lock timeout period so that a transaction either skips past the locked rows, or rolls back.
• Using lock hints in SQL statements to control lock granularity, or specify custom behavior on encountering locked rows.
• Using bound connections to allow multiple connections to share the same locks.
• Using application locks to extend the resources that can be locked.
Controlling Concurrency and Locking Via the Isolation Level

- **READ UNCOMMITTED** – A transaction operating in READ UNCOMMITTED isolation level takes no locks while performing SELECT operations so it cannot block on locks held by other transactions.

- **READ COMMITTED** – The default isolation level, in which SQL Server holds shared locks only until the data has been read, and holds exclusive locks until the end of the transaction.

- **REPEATABLE READ** – A transaction operating in REPEATABLE READ isolation level keeps shared locks and exclusive locks until the end of the transaction.
Controlling Concurrency and Locking Via the Isolation Level

- **SERIALIZABLE** – The most restrictive isolation level, SERIALIZABLE adopts a special locking mechanism, using key-range locks, and holds all locks until the end of the transaction, so that users can't insert new rows into those ranges.

- **SNAPSHOT** – Has the outward appearance of SERIALIZABLE, but operates under a completely different concurrency model, optimistic concurrency,
Controlling Locking: SQL Server Lock Escalation

Since SQL Server 2008 you can also control how SQL Server performs the Lock Escalation – through the ALTER TABLE statement and the property LOCK_ESCALATION.

3 different options:
- **TABLE**: Always performs the Lock Escalation to the table level
- **AUTO**: Lock Escalation is performed to the partition level, if the table is partitioned, and otherwise to the table level.
- **DISABLE**: Disable Lock Escalation for that specific table. Lock Manager of SQL Server can then consume a huge amount of memory. **Not Recommended !!!!**

```sql
-- Controlling Lock Escalation
ALTER TABLE Person.Person
SET (
    LOCK_ESCALATION = AUTO -- or TABLE or DISABLE
)
GO
```
Lock Escalation

- System is decreasing the granularity of your locks
- Ex: DB turning your 100 row-level locks against a table into a single table-level lock.

**Oracle will never escalate a lock. Never.**

- The terms lock conversion and lock promotion are synonymous.
Application-level “Optimistic Locking”

Idea: strive for short transactions or short lock duration

Approach:
- aim at two-phase structure of transactions: read phase + short write phase
- run queries under relaxed isolation level (typically read committed)
- rewrite program to test for concurrent writes during write phase

Example:

```sql
Select Balance, Counter Into :b, :c
From Accounts Where AccountNo = :x
...
compute interests and fees, set b, ...
...
Update Accounts
Set Balance = :b, Counter = Counter + 1
Where AccountNo = :x And Counter = :c
```

avoids lost updates, but cannot guarantee consistency
Optimistic Locking

- Defers all locking up to the point right before the update is performed.
- One popular implementation of optimistic locking is to keep the old and new values in the application, and upon updating the data, use an update like

```
Update table
Set column1 = :new_column1, column2 = :new_column2, ....
Where primary_key = :primary_key
And decode( column1, :old_column1, 1 ) = 1
And decode( column2, :old_column2, 1 ) = 1
```

Other Options:
- Optimistic Locking Using a Version Column (systimestamp column)
- Optimistic Locking Using a Checksum
Optimistic Locking Using a Version Column

**EODA@ORA12CR1> create table dept**

2 ( deptno number(2),
3 dname varchar2(14),
4 loc varchar2(13),
5 last_mod timestamp with time zone
6 default systimestamp
7 not null,
8 constraint dept_pk primary key(deptno)
9 )
10 /
11 Table created.

• Then we INSERT a copy of the DEPT data into this table:

**EODA@ORA12CR1> insert into dept( deptno, dname, loc )**

2 select deptno, dname, loc
3 from scott.dept;
4 rows created.

**EODA@ORA12CR1> commit**
That code re-creates the DEPT table, but with an additional LAST_MOD column that uses the TIMESTAMP WITH TIME ZONE data type.

We have defined this column to be NOT NULL so that it must be populated, and its default value is the current system time.

**TIMESTAMP** data type has the highest precision available in Oracle, typically going down to the microsecond.

For an application that involves user think time, this level of precision on the TIMESTAMP is more than sufficient.

The odds of two people reading and modifying the same row in the same fraction of a second are very small indeed.
Optimistic Locking Using a Version Column

- Maintain this value.

1. Application can maintain the LAST_MOD column by setting its value to SYSTIMESTAMP when it updates a record.

2. A trigger/stored procedure
   - Trigger will add additional processing on top of that already done by Oracle.
   - Each application is responsible for maintaining this field
     - It needs to consistently verify that the LAST_MOD column was not changed and set the LAST_MOD column to the current SYSTIMESTAMP.
   - The best way: encapsulating the update logic in a stored procedure and not allowing the application to update the table directly at all.
Optimistic Locking Using a Version Column

- Example, if an application queries the row where DEPTNO=10:

  ```sql
  variable deptno number
  variable dname varchar2(14)
  variable loc varchar2(13)
  variable last_mod varchar2(50)
  begin
  deptno := 10;
  select dname, loc, to_char(last_mod, 'DD-MON-YYYY HH.MI.SSXFF AM TZR')
  into :dname,:loc,:last_mod
  from dept
  where deptno = :deptno;
  end;
  /
  PL/SQL procedure successfully completed.
  ```

- which we can see is currently

  ```sql
  select :deptno dno, :dname dname, :loc loc, :last_mod lm
  from dual;
  DNO DNAME LOC LM
  -------- ------------ ---------- ----------------------------------------
  10 ACCOUNTING NEW YORK 15-APR-2014 07.04.01.147094 PM -06:00
  ```
Optimistic Locking Using a Version Column

• Update statement to modify the information.
• Last line very important
  – Make sure timestamp has not changed and uses the built-in function TO_TIMESTAMP_TZ (tz is short for time zone) to convert the string we saved in from the SELECT statement back into the proper data type.
• line 3 of the UPDATE statement updates the LAST_MOD column to be the current time if the row is found to be updated:

```sql
EODA@ORA12CR1> update dept
2 set dname = initcap(:dname),
3 last_mod = systimestamp
4 where deptno = :deptno
5 and last_mod = to_timestamp_tz(:last_mod, 'DD-MON-YYYY HH.MI.SSXFF AM TZR');
1 row updated.
```
Optimistic Locking Using a Version Column

- One row was updated, the row of interest.
- We updated the row by primary key (DEPTNO) and verified that the LAST_MOD column had not been modified by any other session between the time we read it first and the time we did the update.
- If we were to try to update that same record again, using the same logic but without retrieving the new LAST_MOD value:

```
EODA@ORA12CR1> update dept
2 set dname = upper(:dname),
3 last_mod = systimestamp
4 where deptno = :deptno
5 and last_mod = to_timestamp_tz(:last_mod, 'DD-MON-YYYY HH.MI.SSXFF AM TZR');
0 rows updated.
```
- 0 rows updated is reported this time because the predicate on LAST_MOD was not satisfied.
Unrestricted **multiprogramming level (MPL)** can lead to performance disaster known as **data-contention thrashing**:

- additional transactions cause superlinear increase of lock waits
- throughput drops sharply
- response time approaches infinity
Benefit of MPL Limitation

system admin sets **MPL limit**: during load bursts excessive transactions wait in **transaction admission queue**

avoids thrashing, but poses a tricky tuning problem:
- overly low MPL limit causes long waits in admission queue
- overly high MPL limit opens up the danger of thrashing
problem is even more difficult for highly heterogeneous workloads
Chapter 10: Implementation and Pragmatic Issues

• 10.2 Data Structures of a Lock Manager
• 10.3 Multi-Granularity Locking and Lock Escalation
• 10.4 Transient Versioning
• 10.5 Nested Transactions for Intra-transaction parallelism
• 10.6 Tuning Options
• **10.7 Overload Control**
• 10.8 Lessons Learned
conflict ratio = \[ \frac{\text{# locks held by all trans.}}{\text{# locks held by running trans.}} \]

critical conflict ratio \( \approx 1.3 \)
Conflict-ratio-driven Overload Control Algorithm

upon begin request of transaction t:
    if conflict ratio < critical conflict ratio
    then admit t else put t in admission queue fi

upon lock wait of transaction t:
    update conflict ratio
    while not (conflict ratio < critical conflict ratio)
        among trans. that are blocked and block other trans.
        choose trans. v with smallest product
            #locks held * #previous restarts
        abort v and put v in admission queue od

upon termination of transaction t:
    if conflict ratio < critical conflict ratio then
        for each transaction q in admission queue do
            if (q will be started the first time) or
                (q has been a rollback/cancellation victim and
                    all trans. that q was waiting for are terminated)
                then admit q fi od fi
Doğma:

HPC ve HYBRID Transactional Analytical Processing (HTAP) çevrelerinde transaksiyon yönetimi.
HPC Background

• HPC gives engineers the computation resources they need to speed research and development.

• Examples:
  – Test simulations
  – Modeling solutions
  – Highly complex problems

• Computes nodes
  – In future: Several thousands on Win 7 works stations
  – .Net APP use Oracle Data Access Components (ODAC)
HPC Background-1

- No of Processes running VS SQL Server
  - Job can assigned per CPU Core.
  - 12 cores per compute Nodes.
  - Assume we have N Compute Nodes
  - In Full utilization
    - Job per core
    - $N \times 12 = 1200$ processes of .NET APP vs DB.
What Runs ON HPC

• MATLAB
• Mechanical CAD:
  – ABAQUS
  – CST (Computer Simulation Technology)
• Monte Carlo Simulation
Basic Architecture of an HPC Cluster

The Heat Map view gives instant feedback on the health of the cluster—upwards of 1,000 nodes, without scrolling.
Real Time BI- Windows HPC Server 2008
HYBRID Transactional Analytical Processing – HTAP

1. User submits job.
2. Assigns nodes for client job
3. Compute Nodes run vs SQL Server
   In memory OLTP
4. Job finished, response return to client

Oracle (~10T Data)
12CR1, 64 bit
SQL 2014

Dozens Compute nodes: .Net, Matlab for simulation data mining the results and real time analytics
Win 2008/12 R2

SAN
NAS
Oracle /SQL Data file on HDS storage
Log file on shared Drives using CIFS Protocol to SATA disks.
Real time BI + HTAP on HPC

1. Identify simulation objective
2. Understand simulation data
3. Prepare data on HPC
4. Develop models to explore/analyze and predict data
5. Test Models
6. Deploy Model
7. Monitor performance & re-calibrate

- Use Matlab, .Net App
- Good starting points
- MSSQL data mining
The problem

The OS has no ability to characterize Oracle session traffic

Mixed workloads represent real-world Oracle traffic

How can we prevent certain sessions from consuming too many resources on the server?
Introducing Database Resource Manager

- DBRM is a quality of service (QoS) tool where we can schedule and guarantee sessions a certain level of service.

- What is a “session”?
  - Oracle user, application, computer
  - Sessions are organized into consumer groups

- What resources can be scheduled?
  - CPU utilization
  - Parallel execution
  - Concurrent active sessions
  - Session time limits
  - Etc…
Database Resource Manager (DBRM)

DBRM allows us to guarantee service levels for sessions

DBRM also enables us to cap sessions such that they don’t overreach

DBRM consists of several related components
Resource Manager Components

- Resource Consumer Group
- Plan Directive
- Resource Plan
Configuring Database Resource Manager with EM
Configuring Database Resource Manager with EM
Configuring Database Resource Manager with EM

The Database Resource Manager helps you manage how resources such as CPU and Parallel Queuing are allocated among user sessions.

**TIP** Administration Privileges Specify which users or roles have the system privilege "Administer Resource Manager", which is required for configuring Resource Manager:

- **Consumer Groups** Define Consumer Groups, which are user sessions grouped together based on resource processing requirements.
- **Consumer Group Mappings** Define consumer group mapping rules, which are used to map user sessions to consumer groups.
- **Plans** Define Resource Plans for a single database, which contain directives that specify how resources are allocated to Consumer Groups.
- **Performance Statistics** Monitor the statistics for the currently enabled Resource Plan.
Configuring Database Resource Manager with EM

### Consumer Groups:
- **BATCH_GROUP**: NO, Consumer group for batch operations
- **DEFAULT_CONSUMER_GROUP**: YES, Consumer group for users not assigned to any consumer group
- **DSS_CRITICAL_GROUP**: NO, Consumer group for critical DSS queries
- **DSS_GROUP**: NO, Consumer group for DSS queries
- **ETL_GROUP**: NO, Consumer group for ETL
- **INTERACTIVE_GROUP**: NO, Consumer group for interactive, OLTP operations
- **LOW_GROUP**: NO, Consumer group for low-priority sessions
- **ORASAPPQOS_0**: YES, Consumer group for Application QOS
- **ORASAPPQOS_1**: YES, Consumer group for Application QOS
- **ORASAPPQOS_2**: YES, Consumer group for Application QOS
- **ORASAPPQOS_3**: YES, Consumer group for Application QOS
- **ORASAPPQOS_4**: YES, Consumer group for Application QOS
- **ORASAPPQOS_5**: YES, Consumer group for Application QOS
- **ORASAPPQOS_6**: YES, Consumer group for Application QOS
- **ORASAPPQOS_7**: YES, Consumer group for Application QOS
- **ORASAUTOTASK**: Consumer group for autotask operations
- **SYS_GROUP**: YES, Consumer group for system administrators
Configuring Database Resource Manager with EM
Configuring Database Resource Manager with EM

### General

Create rules to enable the resource manager to automatically assign sessions to consumer groups

**Attribute:** All

### Add Rule for Selected Type

<table>
<thead>
<tr>
<th>Select</th>
<th>Priority</th>
<th>Attribute</th>
<th>Value</th>
<th>Consumer Group</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>1</td>
<td>Service Module and Action</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Service and Module</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Module and Action</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Module</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Service</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Oracle User</td>
<td>SYS, SYSTEM</td>
<td>SYS_GROUP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Client Program</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Client OS User</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Client Machine</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Client ID</td>
<td>No Mappings Specified</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add Rule for Selected Type
A Simple Resource Plan

A Simple Resource Plan

Resource Plan "DAYTIME"

Directive 1
75% of CPU

Consumer Group "OLTP"

Directive 2
15% of CPU

Consumer Group "REPORTING"

Directive 3
10% of CPU

Consumer Group "OTHER_GROUPS"
Resource Plan Key point

- Only one plan can be active on a server at a time
  - In RAC environments, keep plans and schedules the same
  - Use the Scheduler and maintenance windows to switch plans
  - You can use subplans

- A complex resource plan can extend up to 8 priority levels

- We create plans by using DBMS_RESOURCE_PLAN or (as always) Cloud Control 12c
## Built in Resource Plan

<table>
<thead>
<tr>
<th>Consumer Group Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_PLAN</td>
<td>A general purpose plan that prioritizes SYS to allow the DBA to debug hangs and unexpected heavy loads.</td>
</tr>
<tr>
<td>DEFAULT_MAINTENANCE_PLAN</td>
<td>Plan for the maintenance windows. Provides some CPU resources for automated maintenance tasks.</td>
</tr>
<tr>
<td>MIXED_WORKLOAD_PLAN</td>
<td>Plan for managing mixed workloads (interactive and batch).</td>
</tr>
<tr>
<td>DSS_PLAN</td>
<td>Plan for managing DSS workloads (DSS, ETL, and batch).</td>
</tr>
<tr>
<td>ETL_CRITICAL_PLAN</td>
<td>Similar to DSS_PLAN but prioritizes ETL over DSS.</td>
</tr>
</tbody>
</table>

## Built in Consumer Groups

<table>
<thead>
<tr>
<th>Consumer Group Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERACTIVE_GROUP</td>
<td>Consumer group for interactive, OLTP operations.</td>
</tr>
<tr>
<td>BATCH_GROUP</td>
<td>Consumer group for batch operations.</td>
</tr>
<tr>
<td>DSS_GROUP</td>
<td>Consumer group for decision support system (DSS) queries.</td>
</tr>
<tr>
<td>DSS_CRITICAL_GROUP</td>
<td>Consumer group for critical DSS queries.</td>
</tr>
<tr>
<td>ETL_GROUP</td>
<td>Consumer group for data load, or ETL jobs.</td>
</tr>
<tr>
<td>SYS_GROUP</td>
<td>Consumer group for system administrators.</td>
</tr>
<tr>
<td>LOW_GROUP</td>
<td>Consumer group for low-priority sessions.</td>
</tr>
<tr>
<td>OTHER_GROUPS</td>
<td>Default consumer group for all sessions.</td>
</tr>
</tbody>
</table>

# Consumer Groups Mapping Rules

<table>
<thead>
<tr>
<th>Session Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_module_action</td>
<td>A combination of service name, module name, and action name, in this format:</td>
</tr>
<tr>
<td></td>
<td>service_name.module_name.action_name</td>
</tr>
<tr>
<td>service_module</td>
<td>A combination of service and module names in this format:</td>
</tr>
<tr>
<td></td>
<td>service_name.module_name</td>
</tr>
<tr>
<td>module_name_action</td>
<td>A combination of module and action names in this format:</td>
</tr>
<tr>
<td></td>
<td>module_name.action_name</td>
</tr>
<tr>
<td>module_name</td>
<td>The module name in the currently running application.</td>
</tr>
<tr>
<td>service_name</td>
<td>The service name used by the client to establish a connection.</td>
</tr>
<tr>
<td>oracle_user</td>
<td>The Oracle database user name.</td>
</tr>
<tr>
<td>client_program</td>
<td>The name of the client program used to log onto the server.</td>
</tr>
<tr>
<td>client_os_user</td>
<td>The operating system user name of the client.</td>
</tr>
<tr>
<td>client_machine</td>
<td>The name of the computer from which the client is making the connection.</td>
</tr>
<tr>
<td>client_id</td>
<td>The client identifier.</td>
</tr>
</tbody>
</table>

Managing Resource Consumers Groups

- Their initial group is used during initial connection – this happens dynamically depending upon your mapping rules

- You can then either manually or programmatically switch consumer group membership for sessions

```
BEGIN
  DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESS ('17', '12345', 'HIGH_PRIORITY');
END;
```

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (  
    PLAN => 'DAYTIME',  
    GROUP_OR_SUBPLAN => 'OLTP',  
    COMMENT => 'OLTP group',  
    MGMT_P1 => 75,  
    SWITCH_GROUP => 'LOW_GROUP',  
    SWITCH_TIME => 5);
END;
```
Resource Manager in Multitenant Environment

CDB resource plan in root

- Directive: share = 3
  - PDB salespdb

- Directive: share = 3
  - PDB servicespdb

- Directive: share = 1
  - PDB hrdpdb

SQL Server 2016 Resource Governor I/O Memory

• Resource pools. A resource pool represents the server’s physical resources

• Workload groups. A workload group serves as a container for session requests that have similar classification criteria

• Classification. The classification process assigns incoming sessions to a workload group based on the characteristics of the session
SQL Server 2016 Resource Governor I/O Example

-- NOTE: reporting service performance is not affected by resource governance
-- so we must show improvements here by a query
-- clear the buffers
CHECKPOINT
GO
DBCC DROPCLEANBUFFERS
DBCC FREEPROCCACHE
GO

USE [AdventureworksDW2016CTP3]
GO

-- make a temporary table to do IO-intensive write operations on
CREATE TABLE Temp (ID int);
GO

-- do the IO-intensive write operation
INSERT INTO Temp
    select top 1000000 ProductKey from FactResellerSalesXL_CCI;
GO

-- clean up
DROP TABLE Temp
SQL Server 2016 Resource Governor I/O Example

USE [AdventureworksDW2016CTP3]
GO
ALTER RESOURCE POOL dbapool
WITH (
    MAX_IOPS_PER_VOLUME = 50 -- This is a very low IOPS, so should have a noticeable effect
);
GO
ALTER RESOURCE GOVERNOR RECONFIGURE;
SQL Server 2016 Resource Governor I/O Example

USE [AdventureworksDW2016CTP3]
GO
ALTER RESOURCE POOL dbapool
WITH (MAX_IOPS_PER_VOLUME = 50 -- This is a very low IOPS, so should have a noticeable effect)
GO
ALTER RESOURCE GOVERNOR RECONFIGURE;
SQL Server 2016 Resource Governor I/O Example
Wait-depth Limitation (WDL)

Wait depth of transaction $t =$

\[
\begin{cases}
0 & \text{if } t \text{ is running} \\
i + 1 & \text{if } \max \{\text{wait depth of transactions that block } t\} = i
\end{cases}
\]

Policy: allow only wait depths $\leq 1$

Case 1:

Case 2:
Chapter 10: Implementation and Pragmatic Issues

• 10.2 Data Structures of a Lock Manager
• 10.3 Multi-Granularity Locking and Lock Escalation
• 10.4 Transient Versioning
• 10.5 Nested Transactions for Intra-transaction parallelism
• 10.6 Tuning Options
• 10.7 Overload Control

• 10.8 Lessons Learned
Lessons Learned

• Locking can be efficiently implemented, with flexible handling of memory overhead by means of multi-granularity locks
• Tuning options include
  • choice of isolation levels
  • application-level tricks
  • MPL limitation
• Tuning requires extreme caution to guarantee correctness: if in doubt, don‘t do it!
• Concurrency control is susceptible to data-contention thrashing and needs overload control
Lessons Learned

• we looked some more advanced locking topics, including lock mode conversion, when SQL Server acquires additional locks on data that is already locked.

• We covered the special lock mode called key-range locks that can be held on ranges of index keys when running queries under SERIALIZABLE isolation level.

• We looked at when, and how, SQL Server will escalate locks on smaller resources into table or partition locks.

• Finally, we explored latches and compile locks