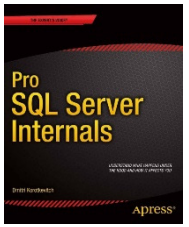
Into into Locking and Blocking

About me

- 20 years of experience in IT
- 14+ years of experience working with Microsoft SQL Server
- Microsoft SQL Server MVP
- Microsoft Certified Master
- Author of “Pro SQL Server Internals”

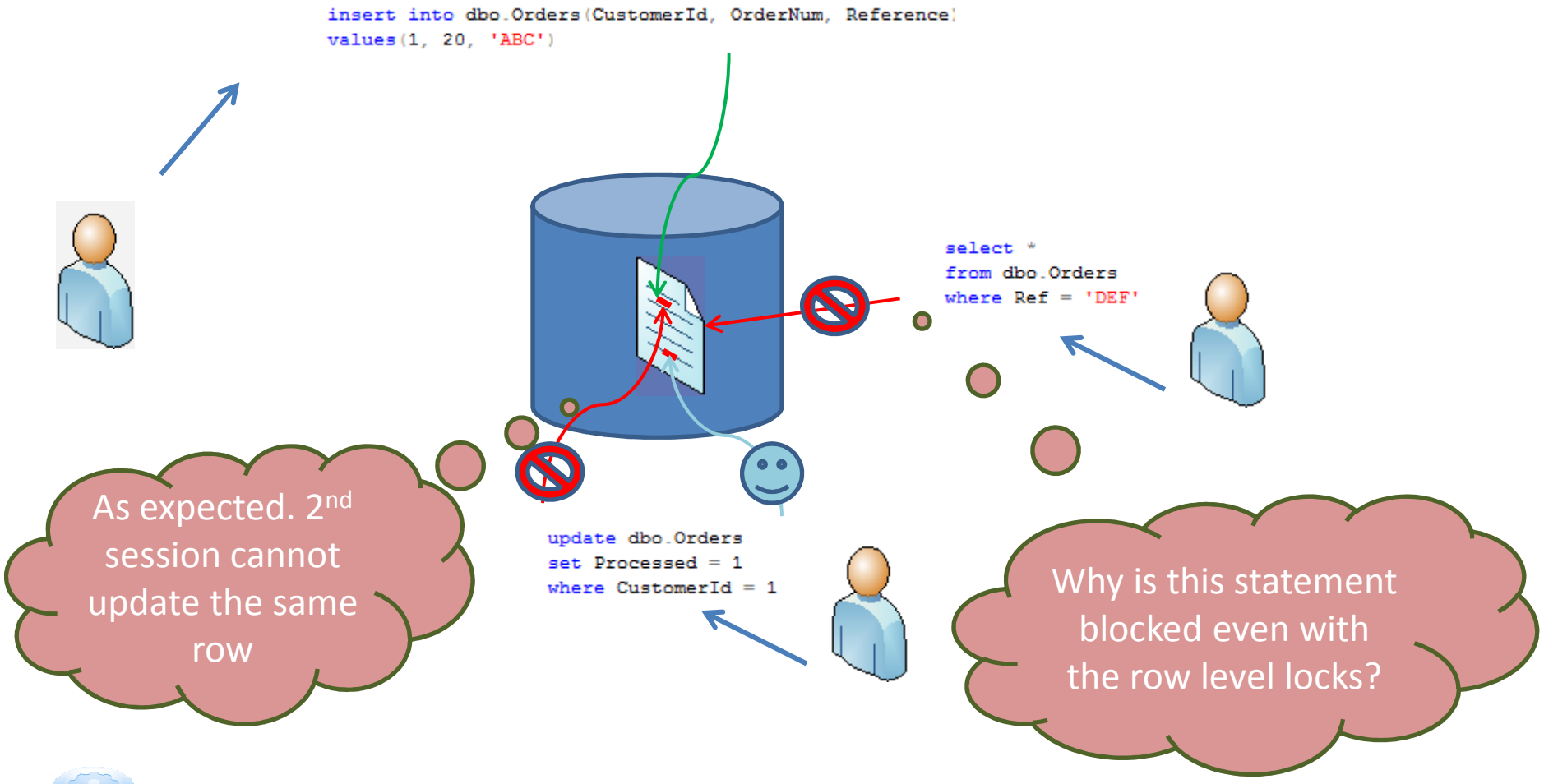
- Blog: http://aboutsqlserver.com
- Email: dmitri@aboutsqlserver.com



Agenda

- Concurrency Model in SQL Server
- Blocking Issues and Deadlocks
- Optimistic Isolation Levels
- Designing Transaction Strategy

Mystery of Row Level Locking



Exclusive Locks (X)

- Acquired by writers when row **is going to be** modified
- Always held till end of transaction regardless of transaction isolation level

```
set transaction isolation level read committed
begin tran
    update dbo.SmallRow set IntField = 0
    where Id = 125

    select * from sys.dm_tran_locks
    where request_session_id = 143
```

	resource_type	resource_subtype	resource_database_id	resource_description	resource_associated_entity_id	resource_lock_partition	request_mode	request_t
1	PAGE		2	1:1438	7566282316938805248	0	IX	LOCK
2	OBJECT		2		218295668	0	IX	LOCK
3	KEY		2	(7d00a258ee47)	7566282316938805248	0	X	LOCK

Update Locks (U)

- Acquired when writers **<u>are looking</u>** for the rows that need to be modified

```
set transaction isolation level read committed
begin tran
    update dbo.SmallRow set IntField = 0
    where IntField = 125

    select * from sys.dm_tran_locks
    where request_session_id = 143
```

EventClass	IntegerData2	Mode	SPID	Type
Lock:Acquired	0 - LOCK	4 - U	143	7 - KEY
Lock:Acquired	0 - LOCK	4 - U	143	7 - KEY
Lock:Acquired	0 - LOCK	4 - U	143	7 - KEY
Lock:Acquired	0 - LOCK	4 - U	143	7 - KEY
Lock:Acquired	0 - LOCK	4 - U	143	7 - KEY
Lock:Acquired	0 - LOCK	4 - U	143	7 - KEY
Lock:Acquired	0 - LOCK	4 - U	143	7 - KEY

	resource_type	resource_subtype	resource_database_id	resource_description	resource_associated_entity_id	resource_lock_partition	request_mode	request_t
1	PAGE		2	1:1438	72057594038845440	0	IX	LOCK
2	OBJECT		2		218295668	0	IX	LOCK
3	KEY		2	(7d00a258ee47)	72057594038845440	0	X	LOCK

Intent Locks (I*)

- Performance would be bad if only key (row) locks exist
- Intent locks indicate locks on child objects

```
set transaction isolation level repeatable read
begin tran
    select * from dbo.SmallRow where Id = 125

    select * from sys.dm_tran_locks
    where request_session_id = 143
```

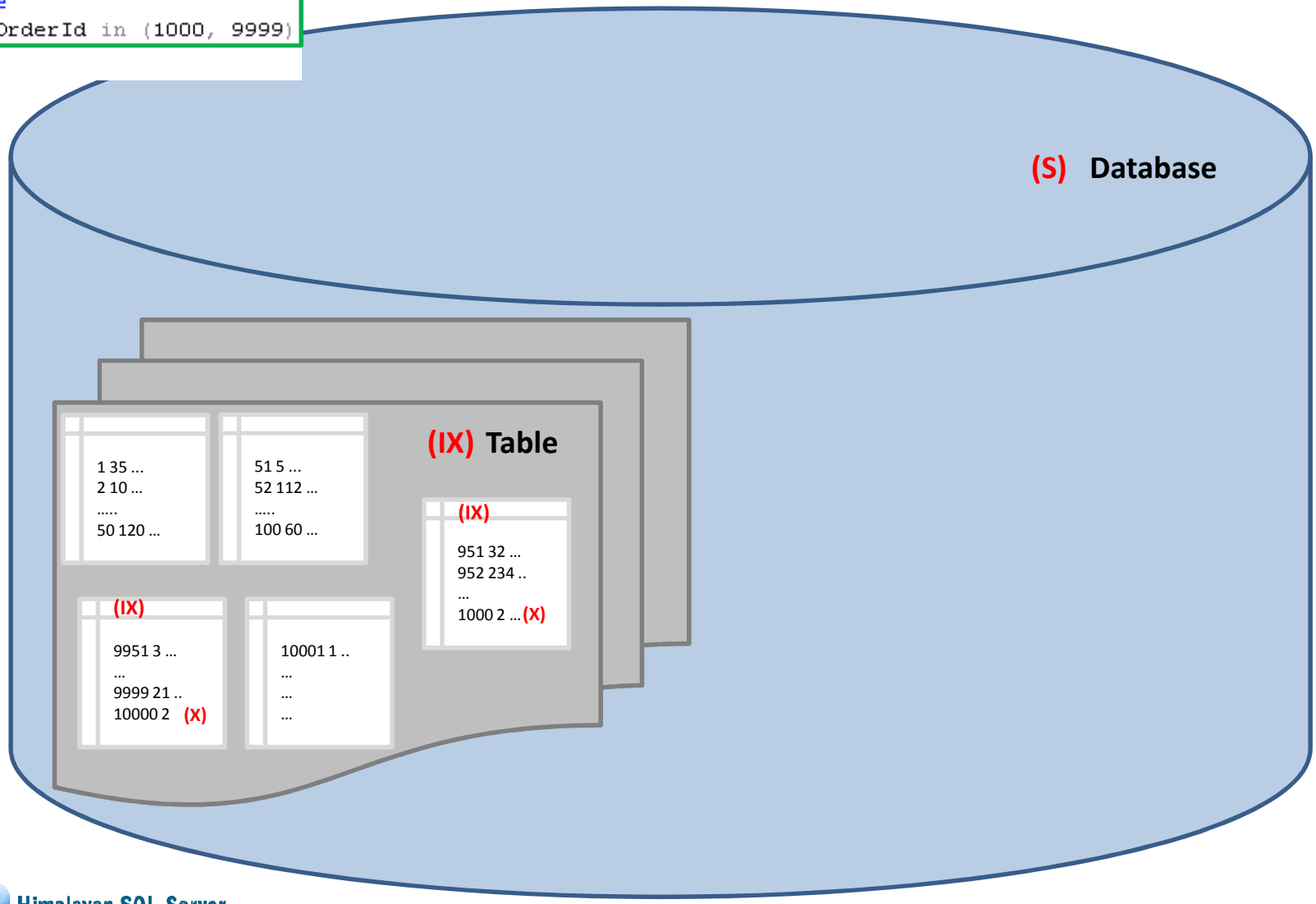
	resource_type	resource_subtype	resource_database_id	resource_description	resource_associated_entity_id	resource_lock_partition	request_mode	request_typ
1	PAGE		2	1:1438	7566282316938805248	0	IS	LOCK
2	OBJECT		2		218295668	0	IS	LOCK
3	KEY		2	(7d00a258ee47)	7566282316938805248	0	S	LOCK

How it works..

```
create table dbo.Orders
(
    OrderId int not null,
    CustomerId int not null,
    OrderNum varchar(32) not null,
    OrderDate smalldatetime not null,
    OrderTotal money not null,
    Processed bit not null
        constraint DEF_Orders_Processed
        default 0,

    constraint PK_Orders
    primary key clustered(OrderId)
)
```

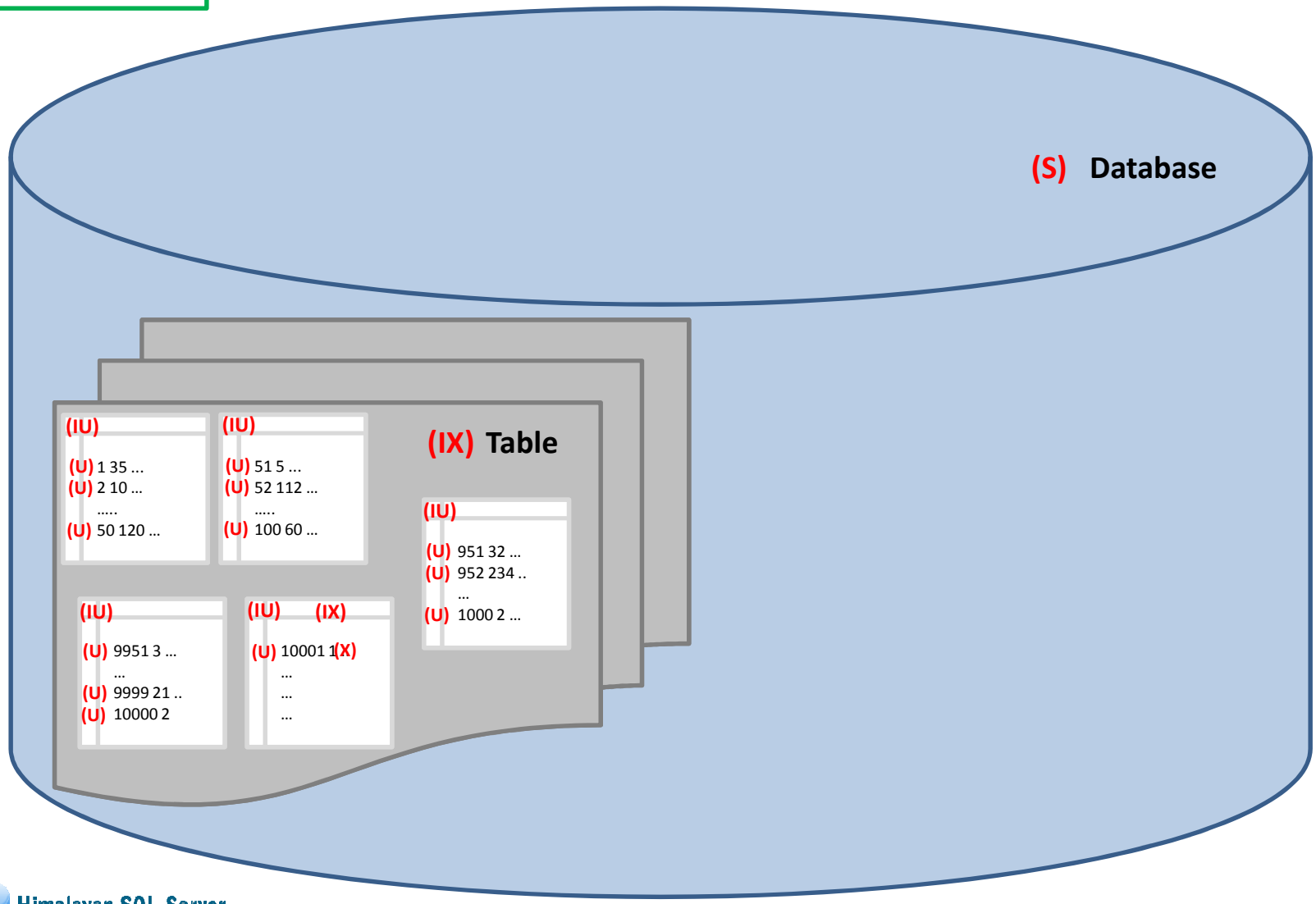
How it works..

```
begin tran
    update dbo.Orders
    set
        Processed = 1
    where
        OrderId in (1000, 9999)
commit
```



```
begin tran
    update dbo.Orders
    set
        Processed = 1
    where
        CustomerId = 1
commit
```

How it works..



Demo

EXCLUSIVE AND UPDATE LOCKS

Lock Compatibility Matrix

	NL	SCH-S	SCH-M	S	U	X	IS	IU	IX	SIU	SIX	UIX	BU	RS-S	RS-U	RI-N	RI-S	RI-U	RI-X	RX-S	RX-U	RX-X
NL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
SCH-S	N	N	C	N	N	N	N	N	N	N	N	N	N	I	I	I	I	I	I	I	I	I
SCH-M	N	C	C	C	C	C	C	C	C	C	C	C	C	I	I	I	I	I	I	I	I	I
S	N	N	C	N	N	C	N	N	C	N	C	C	C	N	N	N	N	N	C	N	N	C
U	N	N	C	N	C	C	N	C	C	C	C	C	C	N	C	N	N	C	C	N	C	C
X	N	N	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C
IS	N	N	C	N	N	C	N	N	N	N	N	N	C	I	I	I	I	I	I	I	I	I
IU	N	N	C	N	C	C	N	N	N	N	N	C	C	I	I	I	I	I	I	I	I	I
IX	N	N	C	C	C	C	N	N	N	C	C	C	C	I	I	I	I	I	I	I	I	I
SIU	N	N	C	N	C	C	N	N	C	N	C	C	C	I	I	I	I	I	I	I	I	I
SIX	N	N	C	C	C	C	N	N	C	C	C	C	C	I	I	I	I	I	I	I	I	I
UIX	N	N	C	C	C	C	N	C	C	C	C	C	C	I	I	I	I	I	I	I	I	I
BU	N	N	C	C	C	C	C	C	C	C	C	C	N	I	I	I	I	I	I	I	I	I
RS-S	N	I	I	N	N	C	I	I	I	I	I	I	I	N	N	C	C	C	C	C	C	C
RS-U	N	I	I	N	C	C	I	I	I	I	I	I	I	N	C	C	C	C	C	C	C	C
RI-N	N	I	I	N	N	N	I	I	I	I	I	I	I	C	C	N	N	N	N	C	C	C
RI-S	N	I	I	N	N	C	I	I	I	I	I	I	I	C	C	N	N	N	C	C	C	C
RI-U	N	I	I	N	C	C	I	I	I	I	I	I	I	C	C	N	N	C	C	C	C	C
RI-X	N	I	I	C	C	C	I	I	I	I	I	I	I	C	C	N	C	C	C	C	C	C
RX-S	N	I	I	N	N	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C
RX-U	N	I	I	N	C	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C
RX-X	N	I	I	C	C	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C

Key

N	No Conflict	SIU	Share with Intent Update
I	Illegal	SIX	Shared with Intent Exclusive
C	Conflict	UIX	Update with Intent Exclusive
		BU	Bulk Update
NL	No Lock	RS-S	Shared Range-Shared
SCH-S	Schema Stability Locks	RS-U	Shared Range-Update
SCH-M	Schema Modification Locks	RI-N	Insert Range-Null
S	Shared	RI-S	Insert Range-Shared
U	Update	RI-U	Insert Range-Update
X	Exclusive	RI-X	Insert Range-Exclusive
IS	Intent Shared	RX-S	Exclusive Range-Shared
IU	Intent Update	RX-U	Exclusive Range-Update
IX	Intent Exclusive	RX-X	Exclusive Range-Exclusive

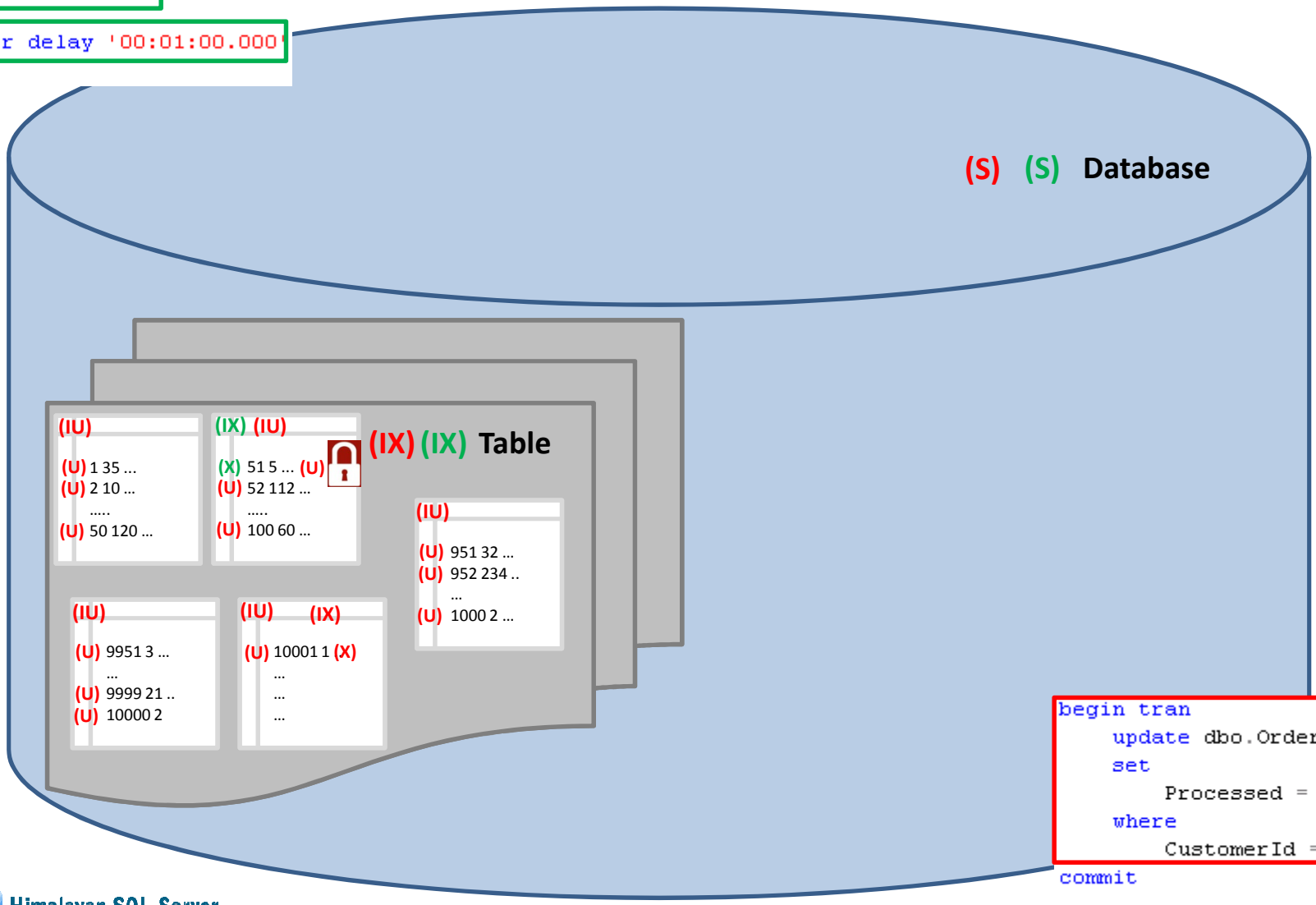
Locks Compatibility (Exclusive and Update)

	U	IX / IU	X
U	☹	☹	☹
IX / IU	☹		☹
X	☹	☹	☹

- ✓ Intent locks are compatible with each other
- ✓ Exclusive (X) locks are incompatible with (X) nor (U)
- ✓ Update (U) locks are incompatible with (U)
- ✓ (X) locks always held till end of transaction

How it works..

```
begin tran
    update dbo.Orders
    set
        Processed = 1
    where
        OrderId = 51
    waitfor delay '00:01:00.000'
commit
```




```
begin tran
    update dbo.Orders
    set
        Processed = 1
    where
        CustomerId = 1
commit
```

Shared Locks (S)

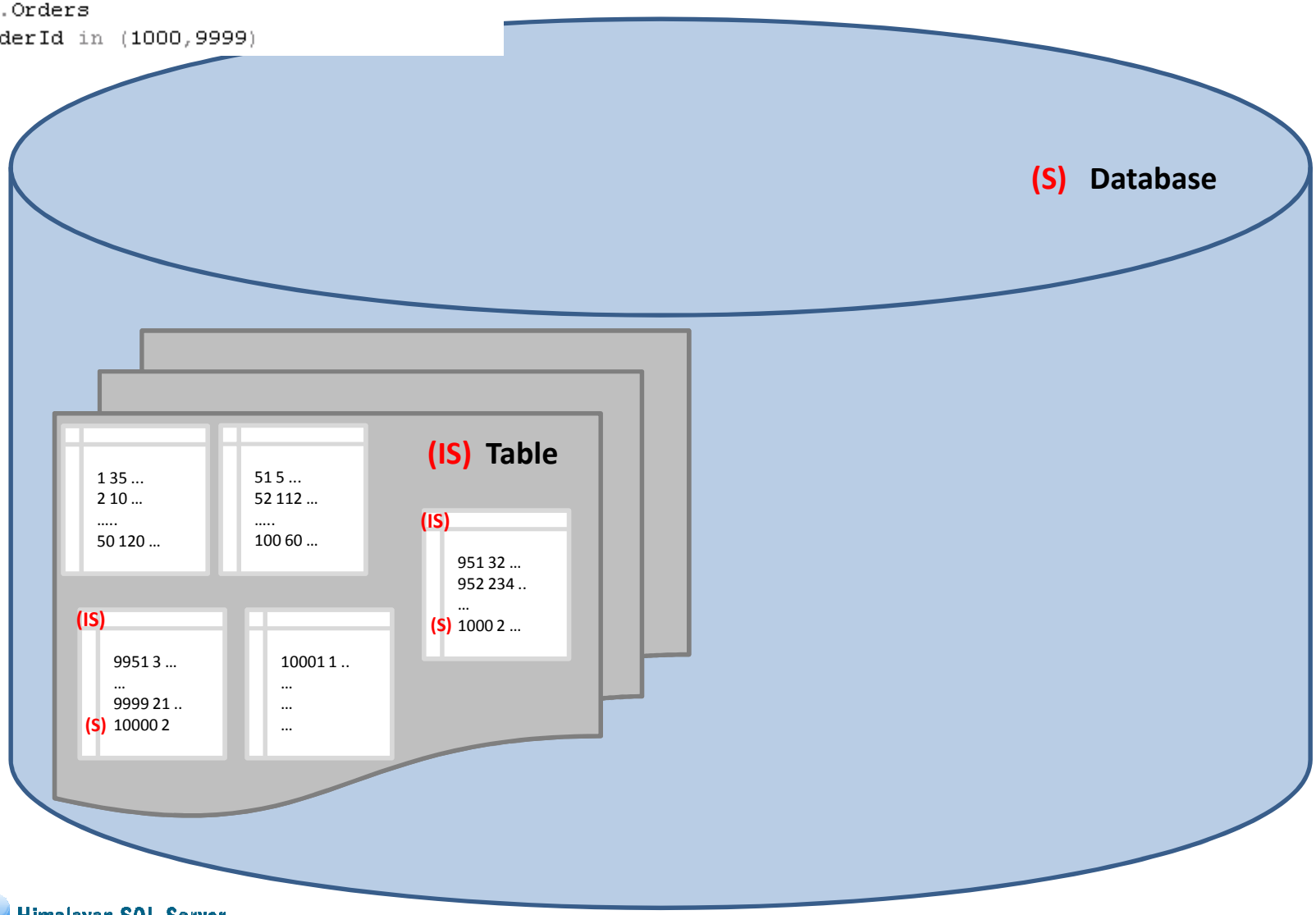
- Acquired by Readers

```
set transaction isolation level repeatable read
begin tran
    select * from dbo.SmallRow where Id = 125

    select * from sys.dm_tran_locks
    where request_session_id = 143
```

	resource_type	resource_subtype	resource_database_id	resource_description	resource_associated_entity_id	resource_lock_partition	request_mode	request_ty
1	PAGE		2	1:1438	7566282316938805248	0	IS	LOCK
2	OBJECT		2		218295668	0	IS	LOCK
3	KEY		2	(7d00a258ee47)	7566282316938805248	0	S	LOCK

How it works..

```
set transaction isolation level read committed

select *
from dbo.Orders
where OrderId in (1000,9999)
```



Locks Compatibility (Slightly bigger)

	S	U	IX / IU / IS	X
S			☹	☹
U		☹	☹	☹
IX / IU / IS	☹	☹		☹
X	☹	☹	☹	☹

- ✓ Shared (S) locks are compatible with (S) and (U)
- ✓ Shared (S) locks are incompatible with (X)
- ✓ (X) locks always held till end of transaction

Pessimistic locking, Readers and Concurrency

Isolation Level	(S) Locks behavior	Table Hint
Read Uncommitted	(S) locks not acquired	NOLOCK
Read Committed	Acquired and released immediately	READCOMMITTED
Repeatable Read	Held till end of transaction	REPEATABLEREAD
Serializable	**Range** locks held till end of transaction	HOLDLOCK

- Concurrency Issues:
  - Dirty Reads
    - Process reads uncommitted (dirty) data from another process transaction
  - Non-Repeatable Reads
    - Data has been modified between reads
  - Phantom Reads & Ghost Rows
    - Rows have been inserted (or deleted) between reads
  - Lost Updates
    - Same row modified by multiple processes. All updates but last one are lost
- In some cases SQL Server can “optimize” the locking
  - Example – Read Committed - (S) lock on PAGE level rather than on ROW level

Demo

TRANSACTIONAL ISOLATION LEVEL AND LOCKING

Troubleshooting

- Main source of concurrency issues in the system – not optimized queries
- How to troubleshoot locking issues
  - Real-time Approach: mainly DMVs (sys.dm_tran_locks)
  - Data Collection: Collecting “Blocked Process Report”
    - SQL Trace
    - Extended Events
    - Event Notifications (Script is included)

Demo

BLOCKING IN THE SYSTEM

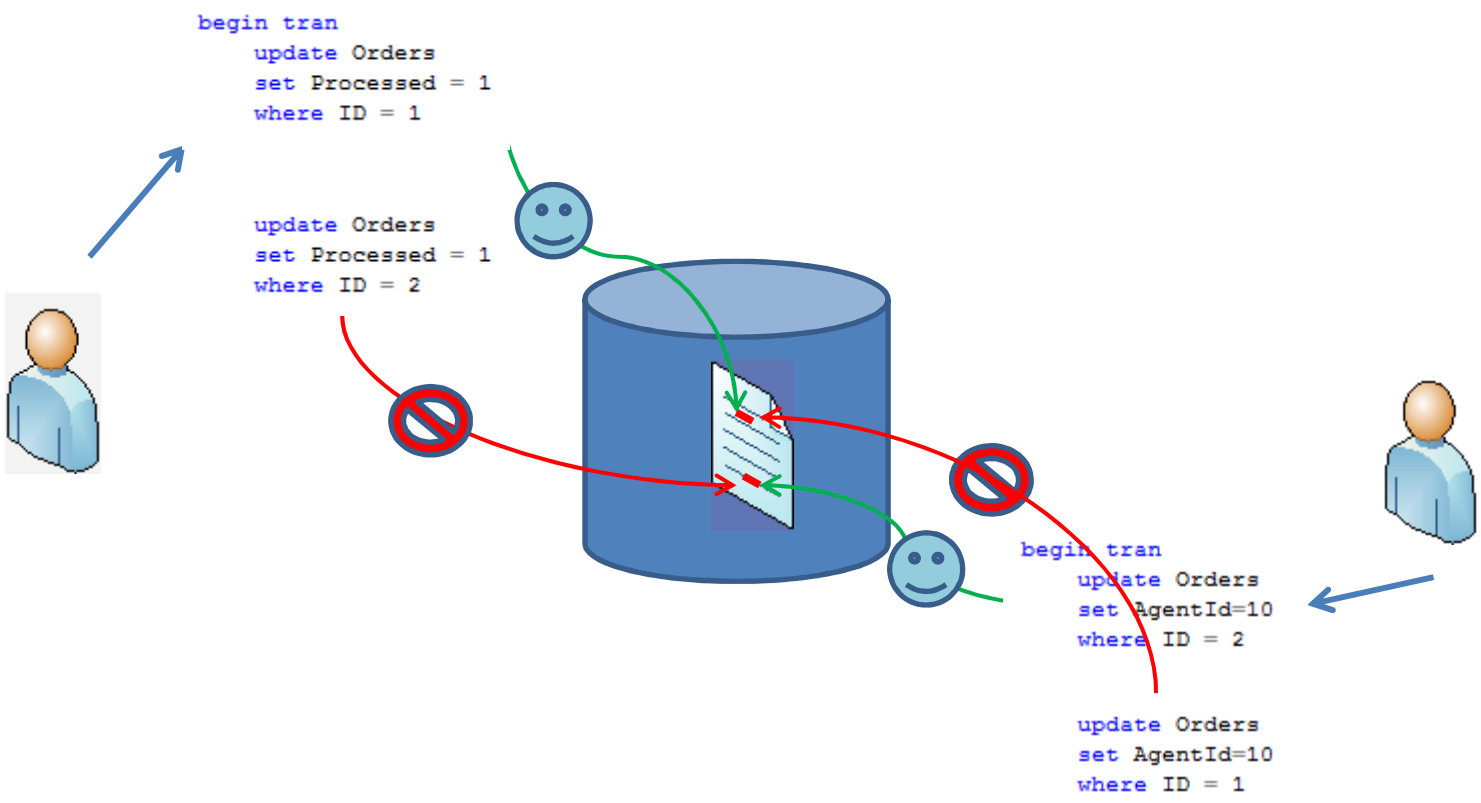
If you have handle..

```
declare
    @H varbinary(max) = 0x03000C00949D7F11ADA7E900659E0000010000000000000000

SELECT
    qp.query_plan,
    SUBSTRING(qt.TEXT, (qs.statement_start_offset/2)+1,
            ((
                CASE qs.statement_end_offset
                    WHEN -1 THEN DATALENGTH(qt.TEXT)
                    ELSE qs.statement_end_offset
                END - qs.statement_start_offset)/2)+1),
    qs.sql_handle,
    qs.execution_count,
    (qs.total_logical_reads + qs.total_logical_writes) / qs.execution_count as [Avg IO],
    qs.total_logical_reads, qs.last_logical_reads,
    qs.total_logical_writes, qs.last_logical_writes,
    qs.total_worker_time,
    qs.last_worker_time,
    qs.total_elapsed_time/1000 total_elapsed_time_in_ms,
    qs.last_elapsed_time/1000 last_elapsed_time_in_ms,
    qs.last_execution_time,
    qp.query_plan
FROM
    sys.dm_exec_query_stats qs
        CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
        OUTER APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
where
    qs.sql_handle = @H
go
```

Results | Messages

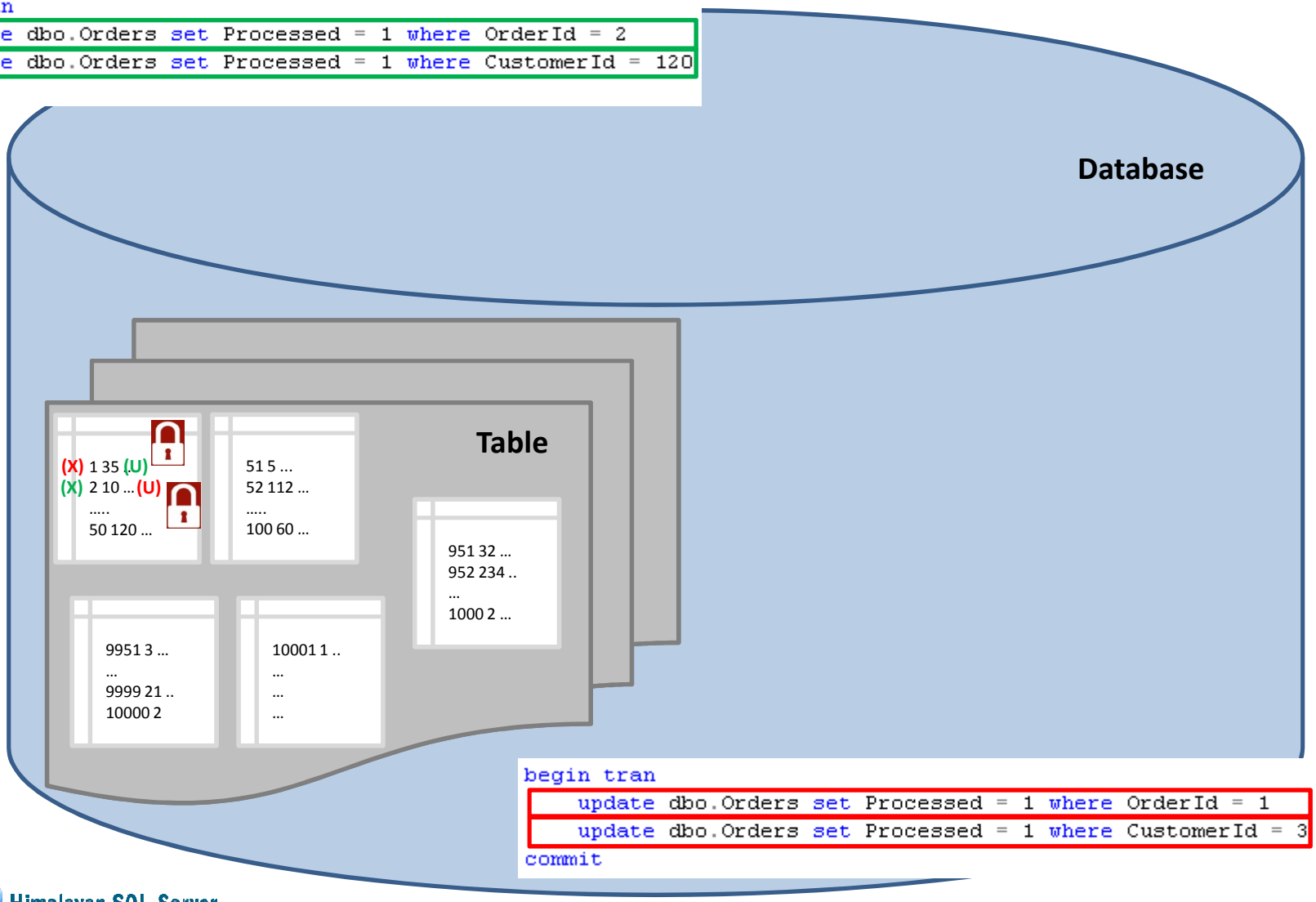
query_plan	(No column name)	sql_handle
<ShowPlanXML xmlns="http://schemas.microsoft.com...	insert into dbo.DataRecords(ID,Field1,Field2,Field3...	0x03000C00949D7F1

Classic Deadlock



Real Life Deadlock

```
begin tran
    update dbo.Orders set Processed = 1 where OrderId = 2
    update dbo.Orders set Processed = 1 where CustomerId = 120
commit
```




```
begin tran
    update dbo.Orders set Processed = 1 where OrderId = 1
    update dbo.Orders set Processed = 1 where CustomerId = 3
commit
```

Troubleshooting Deadlocks

- SQL Profiler – deadlock graph
- Trace flag 1222 – writing deadlock info to error log

  DBCC TRACEON (1222, -1)
- Extended Events
  - Included to System_Health session by default

Demo

DEADLOCKS

Deadlock Graph

```
2010-07-25 15:59:50.53 spid22s      deadlock-list
2010-07-25 15:59:50.53 spid22s          deadlock victim=process84d708
2010-07-25 15:59:50.53 spid22s          process-list
2010-07-25 15:59:50.53 spid22s                  process id=process84d708 <..> spid=54 <..> isolationlevel=read committed (2)<..>
2010-07-25 15:59:50.53 spid22s                      executionStack
2010-07-25 15:59:50.53 spid22s                          frame procname=adhoc line=1 stmtstart=50 <..>
2010-07-25 15:59:50.53 spid22s                              UPDATE [SmallRow] set [CharField] = @1  WHERE [IntField]=@2
2010-07-25 15:59:50.53 spid22s                          frame procname=adhoc line=1 stmtstart=2 <..>
2010-07-25 15:59:50.53 spid22s                              update SmallRow set CharField = 'cde' where IntField = 49998
2010-07-25 15:59:50.53 spid22s                          inputbuf
2010-07-25 15:59:50.53 spid22s                              update SmallRow set CharField = 'cde' where IntField = 49998
2010-07-25 15:59:50.53 spid22s                  process id=process599048 <..>  spid=53 <..> isolationlevel=read committed(2)<..>
2010-07-25 15:59:50.53 spid22s                      executionStack
2010-07-25 15:59:50.53 spid22s                          frame procname=adhoc line=1 stmtstart=58 <..>
2010-07-25 15:59:50.53 spid22s                              UPDATE [SmallRow] set [CharField] = @1  WHERE [IntField]=@2
2010-07-25 15:59:50.53 spid22s                          frame procname=adhoc line=1 stmtstart=2 <..>
2010-07-25 15:59:50.53 spid22s                              update SmallRow set CharField = 'cde' where IntField = 2
2010-07-25 15:59:50.53 spid22s                          inputbuf
2010-07-25 15:59:50.53 spid22s                              update SmallRow set CharField = 'cde' where IntField = 2
2010-07-25 15:59:50.53 spid22s              resource-list
2010-07-25 15:59:50.53 spid22s                  keylock <..> dbid=7  objectname=SqlSat40.dbo.SmallRow indexname=PK_SmallRow <..>
2010-07-25 15:59:50.53 spid22s                      owner-list
2010-07-25 15:59:50.53 spid22s                          owner id=process599048 mode=X
2010-07-25 15:59:50.53 spid22s                      waiter-list
2010-07-25 15:59:50.53 spid22s                          waiter id=process84d708 mode=U requestType=wait
2010-07-25 15:59:50.53 spid22s                  keylock <..> dbid=7 objectname=SqlSat40.dbo.SmallRow indexname=PK_SmallRow  <..>
2010-07-25 15:59:50.53 spid22s                      owner-list
2010-07-25 15:59:50.53 spid22s                          owner id=process84d708 mode=X
2010-07-25 15:59:50.53 spid22s                      waiter-list
2010-07-25 15:59:50.53 spid22s                          waiter id=process599048 mode=U requestType=wait
```

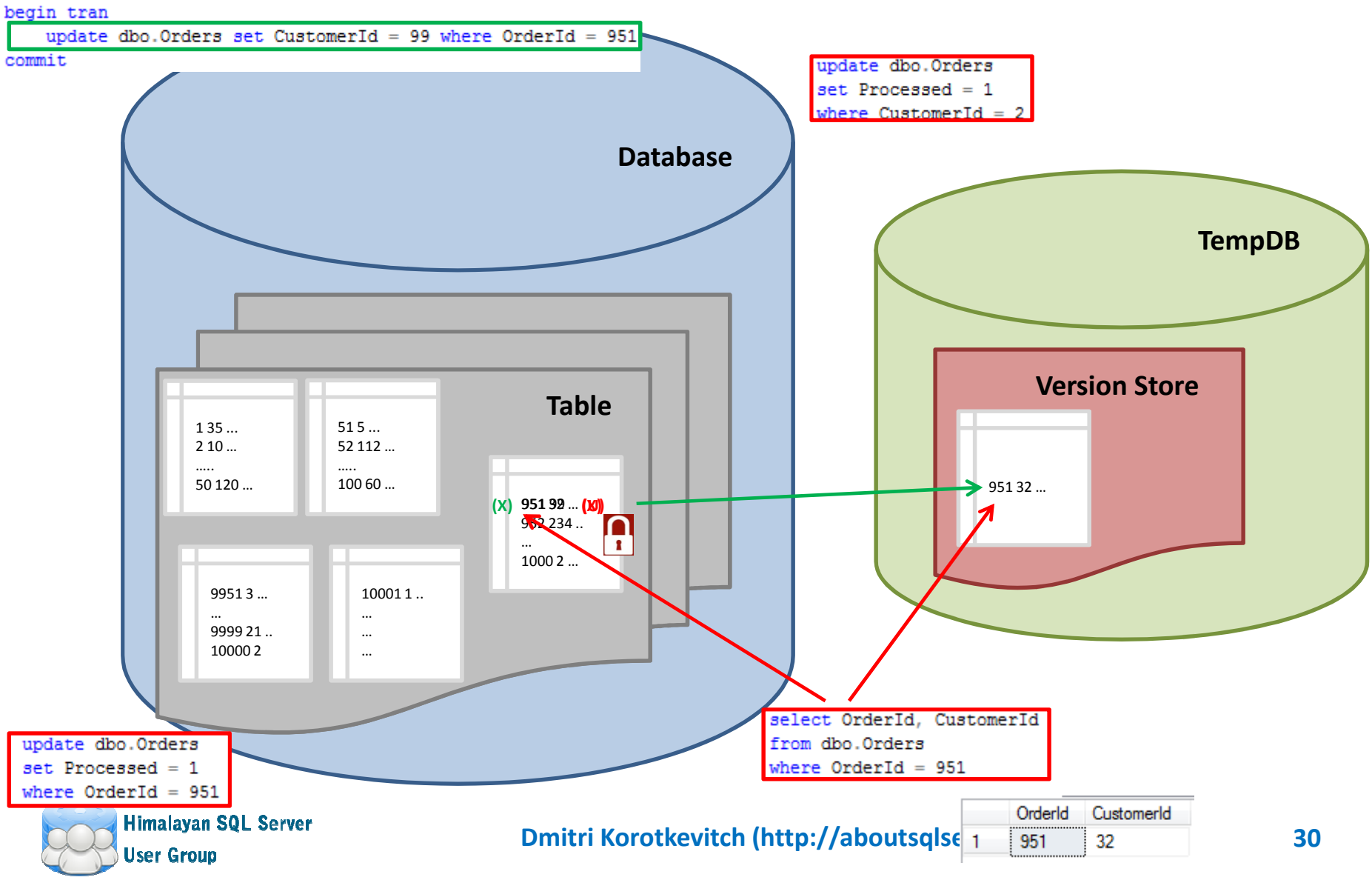
Lock Escalation

- SQL Server tries to escalate locks to the table/partitions level
  - Initial Threshold: ~5,000 locks on the object
  - If it fails, it tries again every ~1,250 locks
- It is completely normal unless it introduces the issues
  - Sign is high percent of intent locks
- Pattern: batch operation triggers lock escalation. All other sessions accessing the object are blocked
- Solution
  - Trace flag 1211 (instance level) – not recommended but sometimes required
  - SQL Server 2008+: *alter table .. set lock_escalation*
  - Optimistic transaction isolation levels
    - Row version model – writers don't block readers

Demo

LOCK ESCALATION

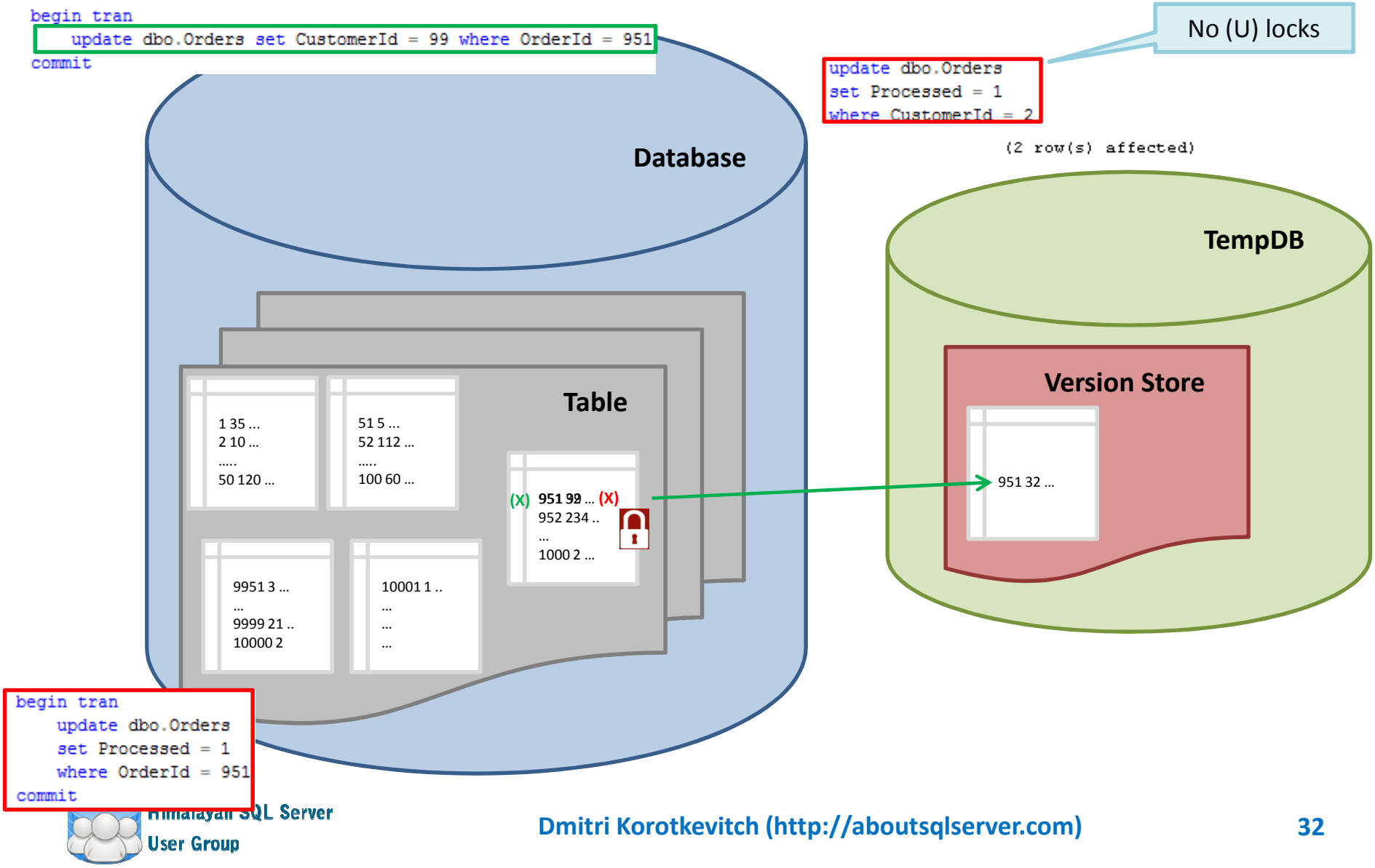
Read Committed Snapshot



Demo

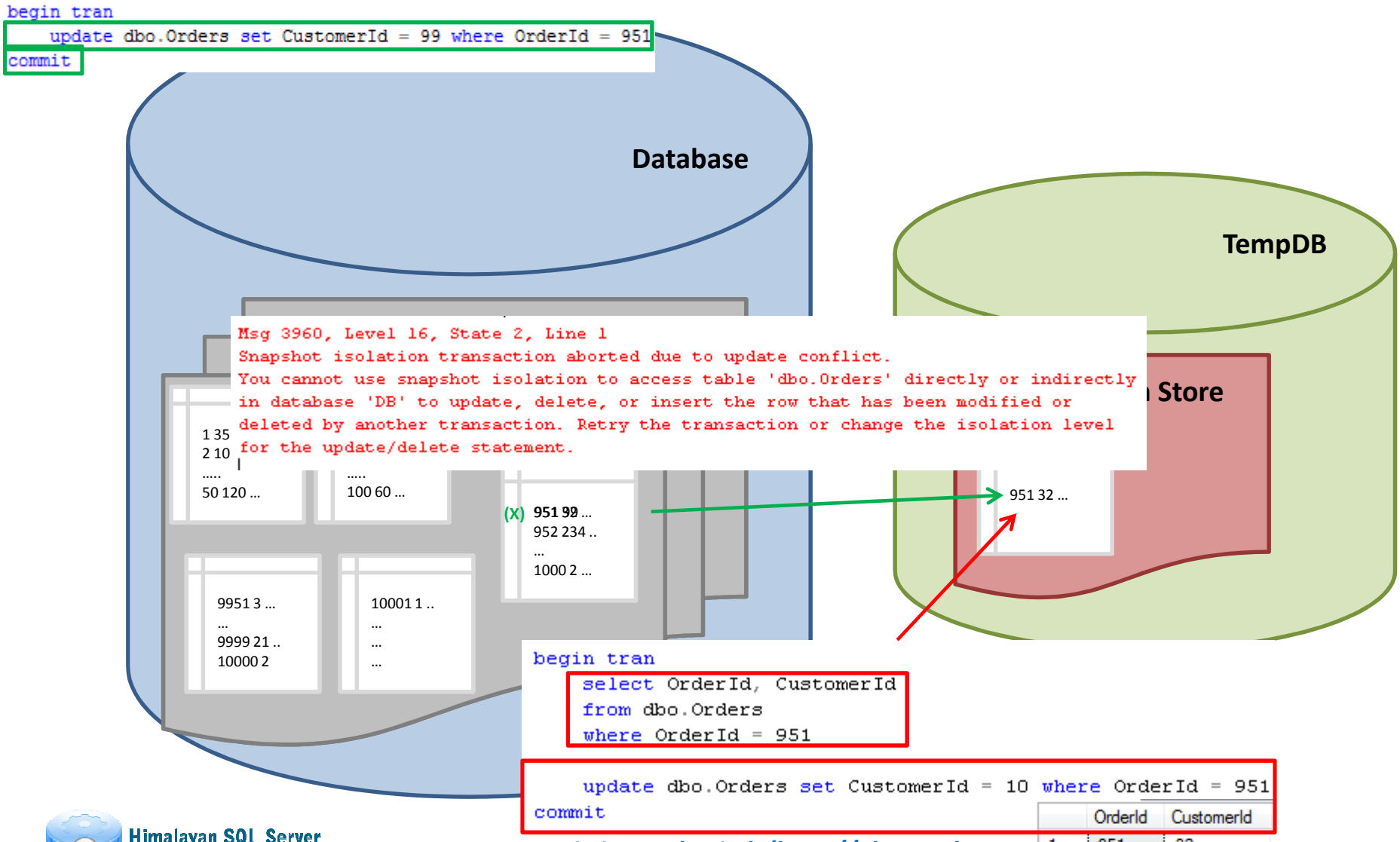
READ COMMITTED SNAPSHOT

Snapshot (U/X locks)



Himalayan SQL Server User Group

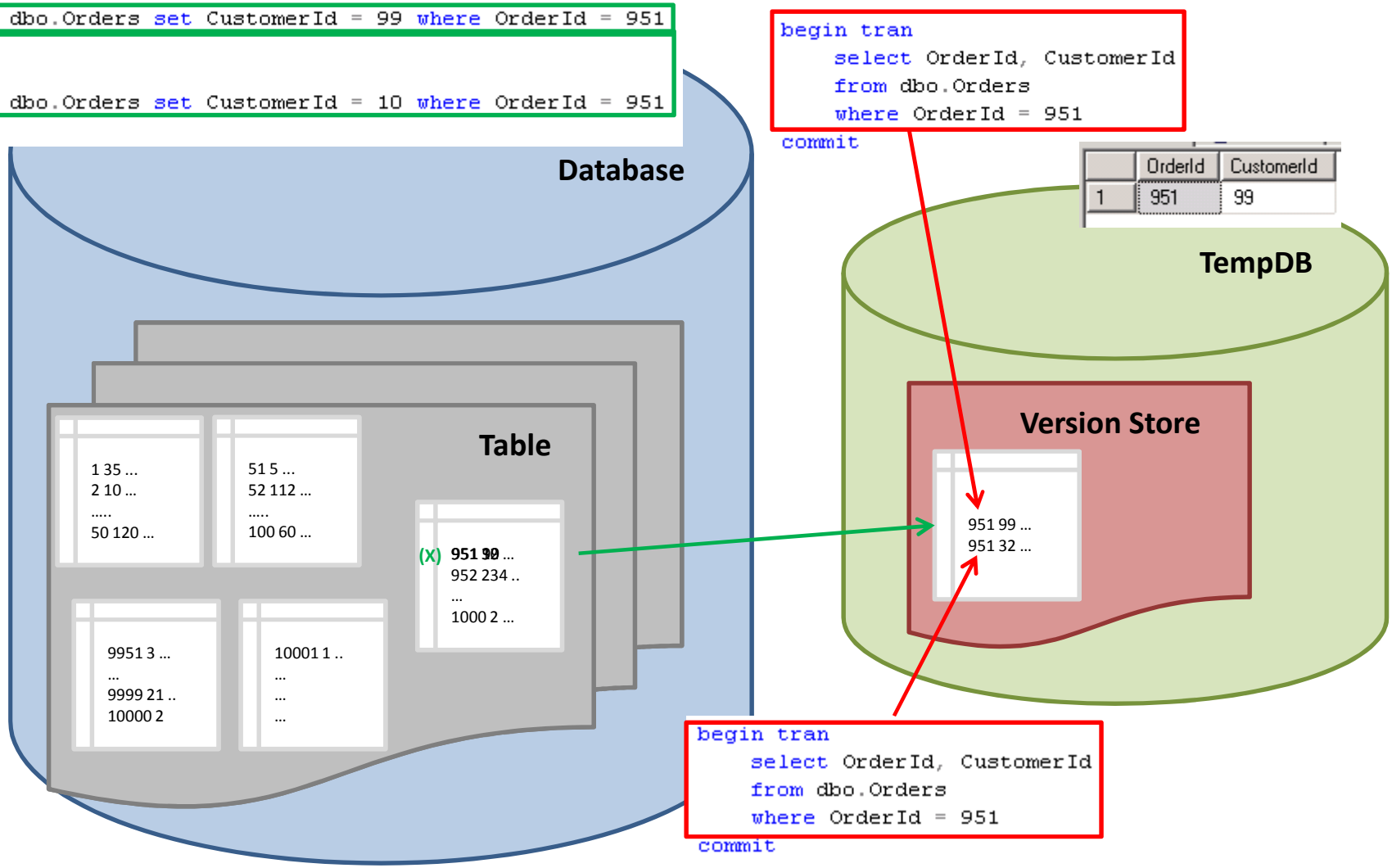
Snapshot (Updates – same row)



Snapshot (Selects)

```
begin tran
    update dbo.Orders set CustomerId = 99 where OrderId = 951
commit
begin tran
    update dbo.Orders set CustomerId = 10 where OrderId = 951
commit
```

```
begin tran
    select OrderId, CustomerId
    from dbo.Orders
    where OrderId = 951
commit
```



	OrderId	CustomerId
1	951	99

```
begin tran
    select OrderId, CustomerId
    from dbo.Orders
    where OrderId = 951
commit
```

	OrderId	CustomerId
1	951	32

Demo

SNAPSHOT ISOLATION LEVEL

Optimistic Isolation Levels

- Using Row Versioning
  - Writers don't block readers
- Read committed snapshot (Statement Level Consistency)
  - Only 1 "last committed" version stored
  - Can be switched as the database option without the code changes
    - **Good emergency technique if system suffers of concurrency issues**
    - **Helps only to resolve conflicts between readers and writers**
- Snapshot (Transaction Level Consistency)
  - Multiple "old" versions stored in the version store
  - Different behavior (Demo)
- TANSTAAFL! (There Ain't No Such Thing As A Free Lunch!)
  - Bigger row size (14 bytes pointer)
  - **Increases Index fragmentation during updates.**
    - Do not use FILLFACTOR = 100
  - Tempdb load (When it is enabled, not even used)
  - Development challenges
    - Referential integrity based on triggers
    - Different update behavior

Demo

OPTIMISTIC ISOLATION LEVEL AND FRAGMENTATION

Schema Locks

- Every statement acquires Schema Stability (SCH-S) lock that prevents underlying objects to be altered
- DDL operations acquire Schema Modification (SCH-M) locks incompatible with (SCH-S).
  - Access to the objects is blocked for the duration of the operation
- Typical problems:
  - Partition function alteration (especially when data movement is involved)
  - Online index rebuild in very busy OLTP systems
  - Full-text index rebuild

Demo

SCHEMA LOCKS

What isolation level should I choose?

- It depends ☺
- Do not use READ UNCOMMITTED unless you don't worry about data consistency
- Read Committed – *good enough* when queries are optimized
- Optimistic isolation levels
  - Good for Data Warehouse/Reporting systems
  - Questionable for OLTP
    - RCSI can be OK
      - Can you live with additional index fragmentation?
      - Can system handle extra tempdb load?

Key Points

- 1st rule to improve concurrency – optimize queries
- 2nd rule to improve concurrency – optimize queries
- 3rd rule to improve concurrency – optimize queries
- Choose appropriate isolation level
- Use short transactions
- Update entities in the same orders
- Do not update the row multiple times in 1 transaction
- Be careful with SQL Generators and ORM frameworks
- http://aboutsqlserver.com – a lot of stuff

Q & A

- Thank you very much for attending!

- Slides and scripts are available: http://aboutsqlserver.com/presentations
- Email: dmitri@aboutsqlserver.com