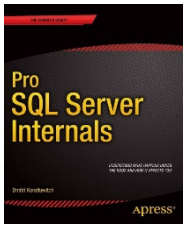
«Это Мое»

Блокировки, Дедлоки и Прочая Чепуха

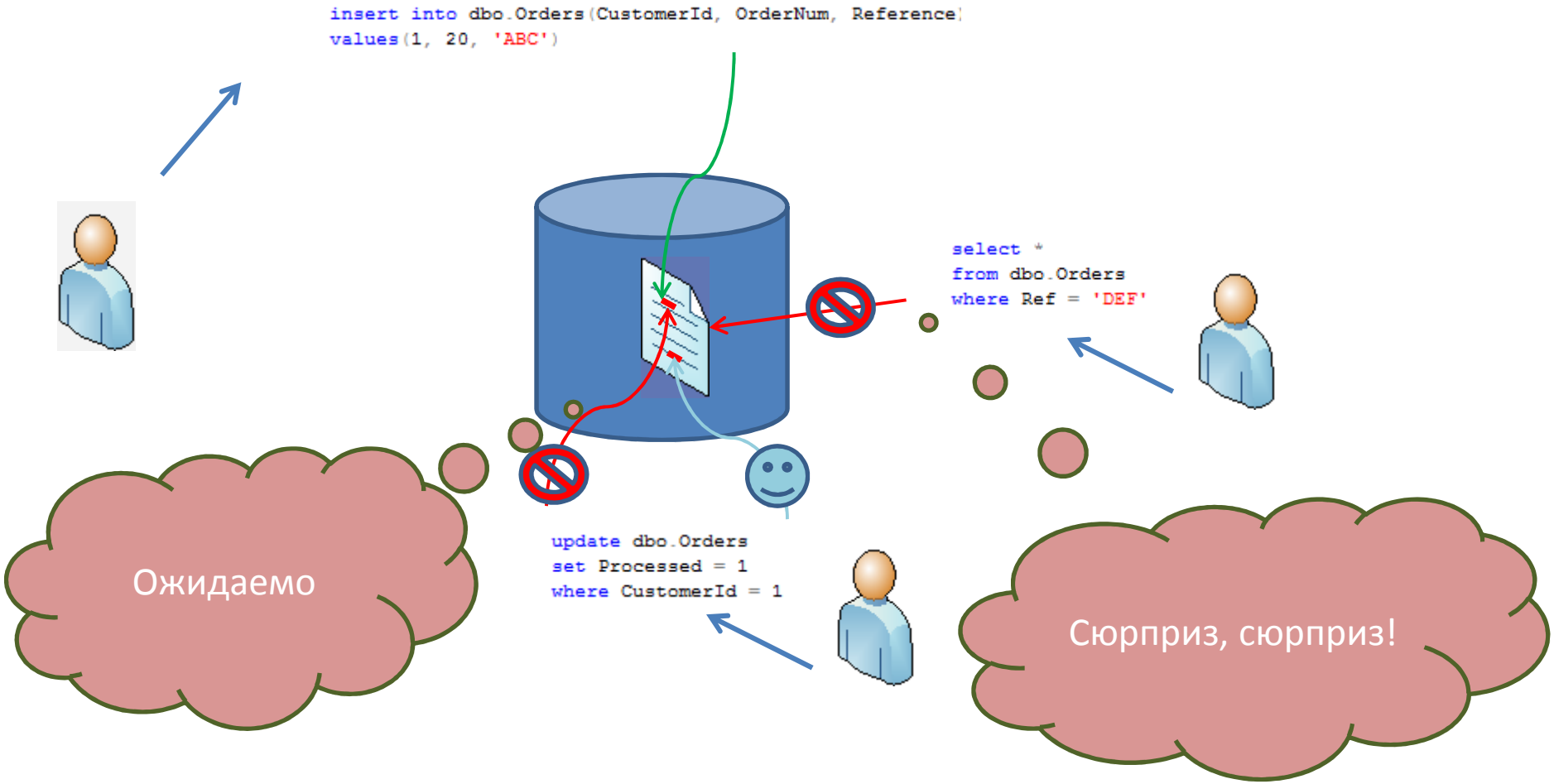
Здравствуйте!

- 20+ лет в IT
- 15+ лет работы с Microsoft SQL Server
- Microsoft SQL Server MVP
- Microsoft Certified Master
- Автор: “Pro SQL Server Internals”

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



Загадочные Блокировки



Эксклюзивные Блокировки (X)

- Берутся «писателями» перед модификацией данных
- Сохраняются до конца транзакции независимо от уровня изоляции

```
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

Блокировки на Обновление (U)

- Берутся «писателями» во время проверки, следует ли обновлять данные

```
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	72057594038927360	0	IX	LOCK
2	OBJECT		2		218295668	0	IX	LOCK
3	KEY		2	(7d00a258ee47)	72057594038927360	0	X	LOCK

Блокировки по намерению (I*)

- Указывают на блокировки на «нижнем уровне»
  - Эксклюзивная (X) блокировка на записи
  - Эксклюзивная блокировка по намерению (IX) на странице и таблице

```
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
```

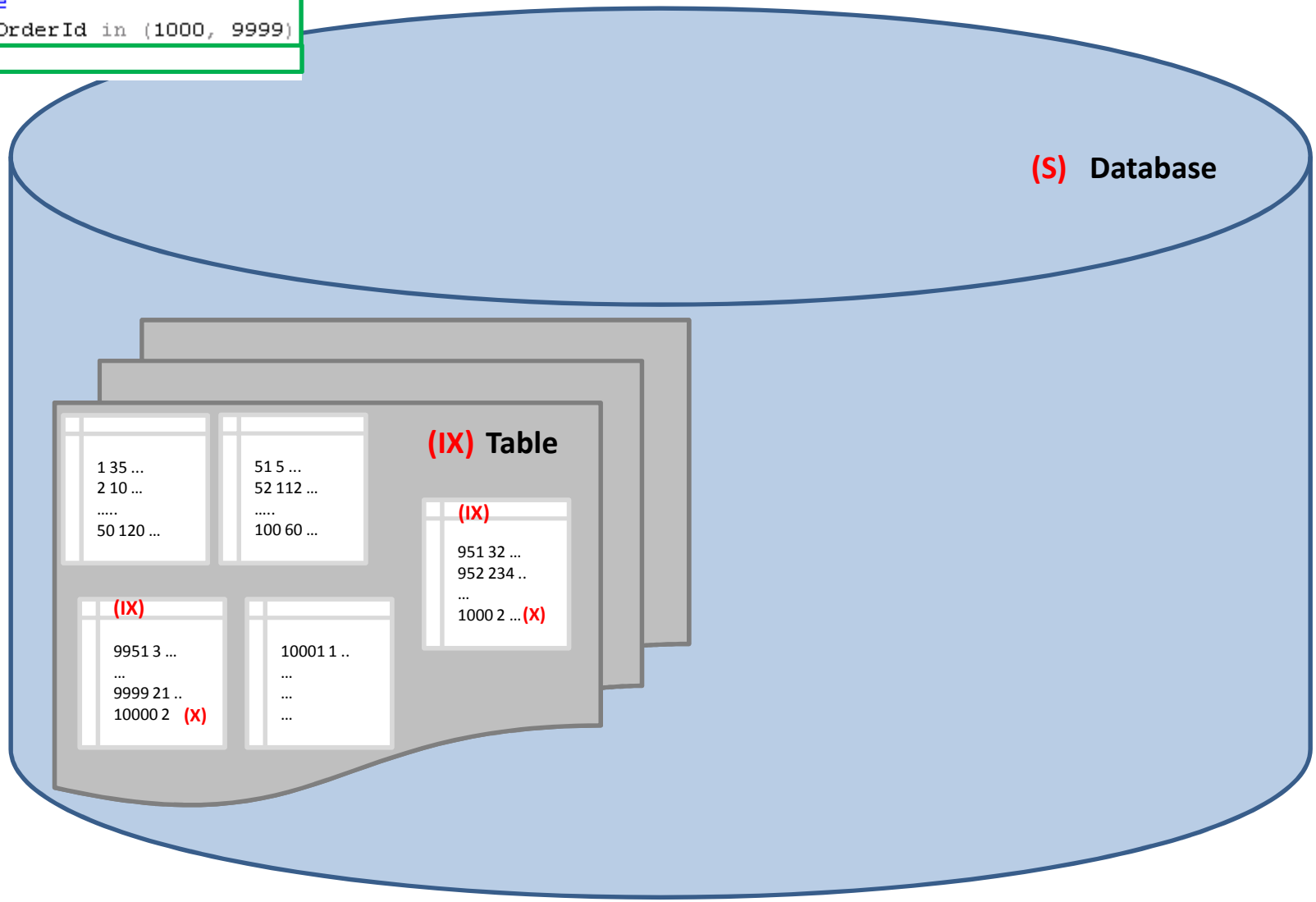
	resource_type	resource_subtype	resource_database_id	resource_description	resource_associated_entity_id	resource_lock_partition	request_mode	request_typ
1	PAGE		2	1:1438	7566282316938805248	0	IS	LOCK
2	OBJECT		2		218295668	0	IS	LOCK
3	KEY		2	(7d00a258ee47)	7566282316938805248	0	S	LOCK

Как оно работает

```
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)
)
```

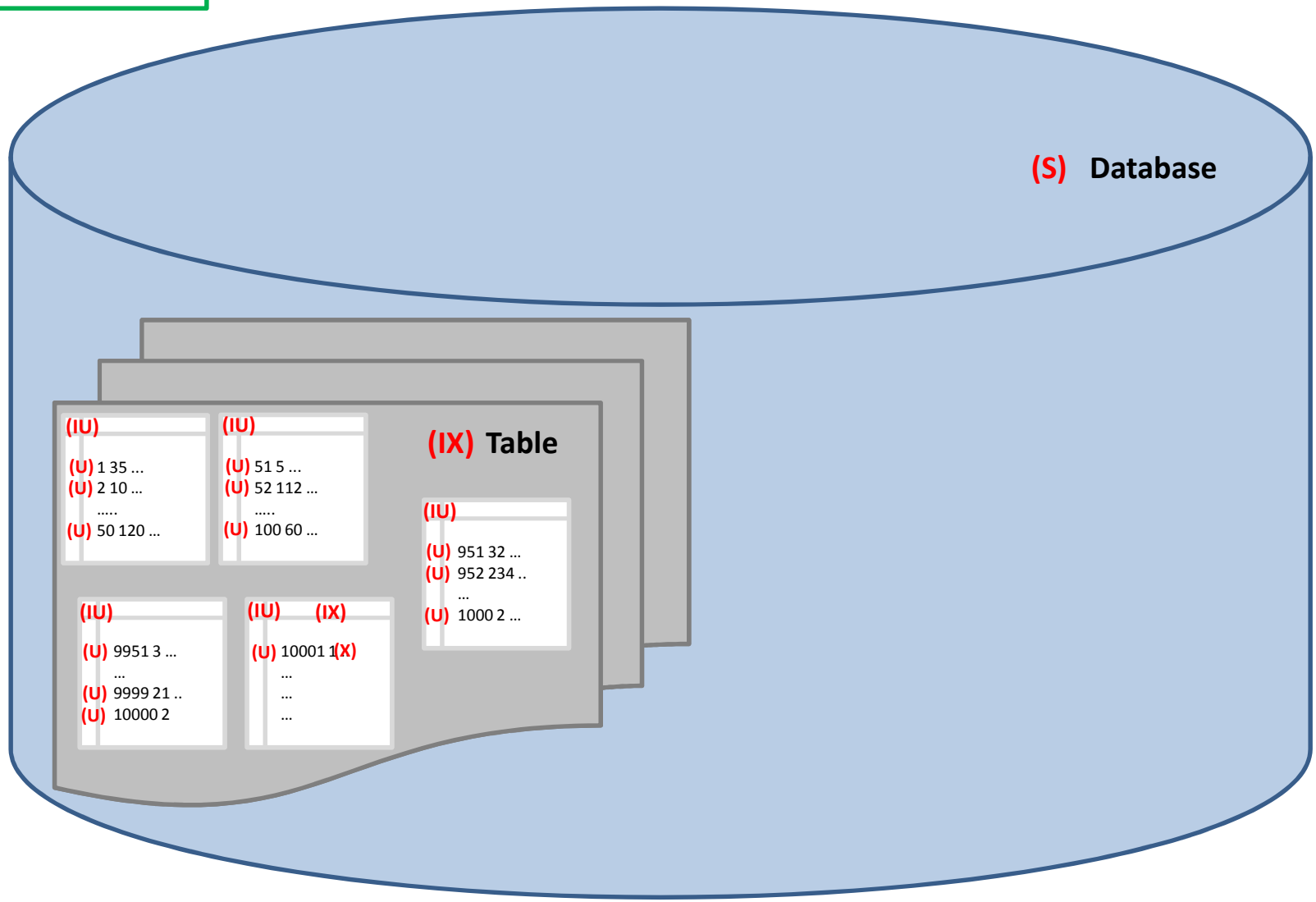
Как оно работает

```
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
```

Как оно работает



Demo

(X) И (U) БЛОКИРОВКИ

Совместимость Блокировок

	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

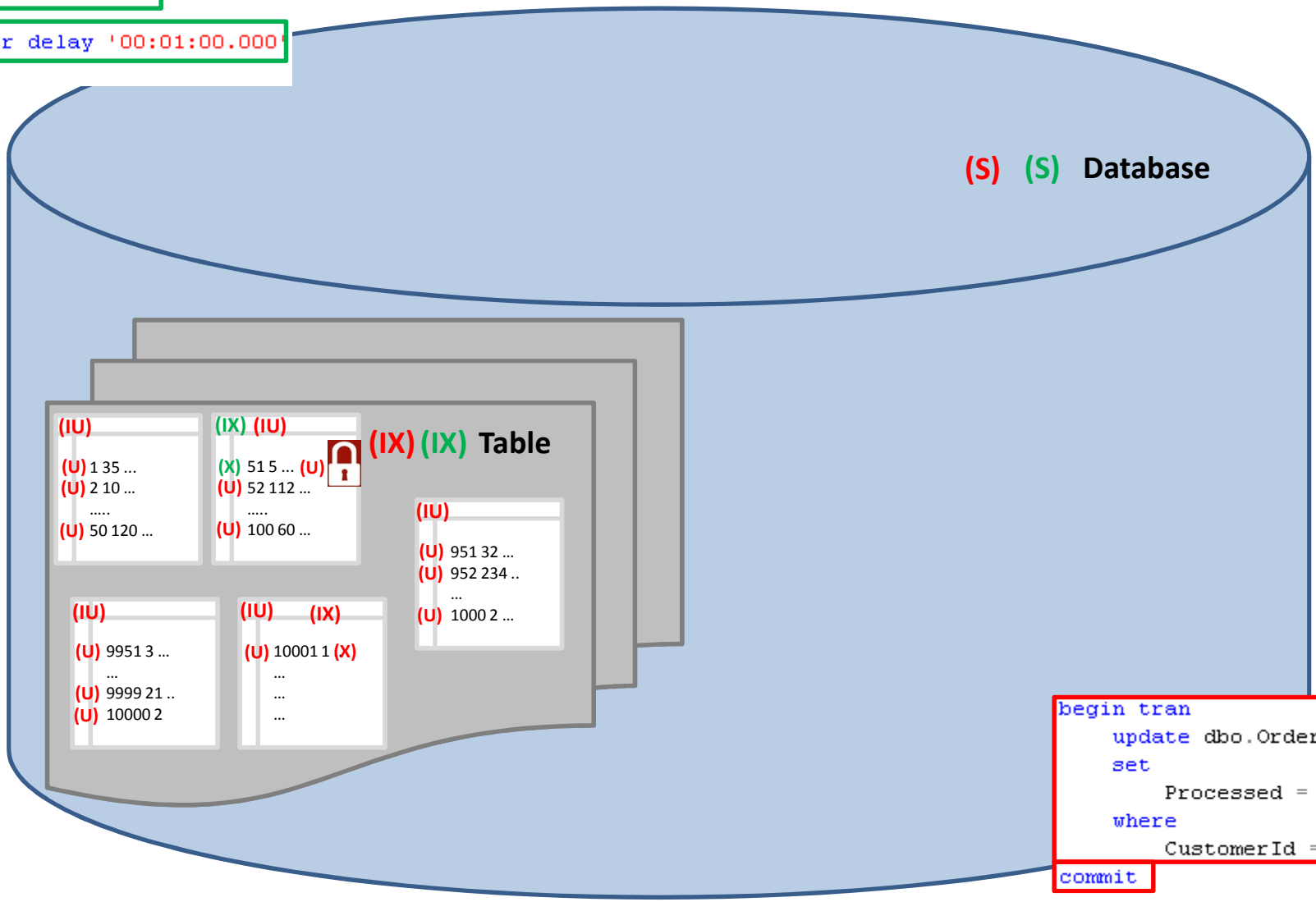
Совместимость Блокировок

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

- ✓ Блокировки по намерению (I*) совместимы друг с другом
- ✓ Эксклюзивные (X) блокировки несовместимы ни с чем
- ✓ Блокировки на обновление (U) не совместимы друг с другом
- ✓ (X) блокировки держатся до конца транзакции

Как оно работает

```
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
```

Разделяемые блокировки (S)

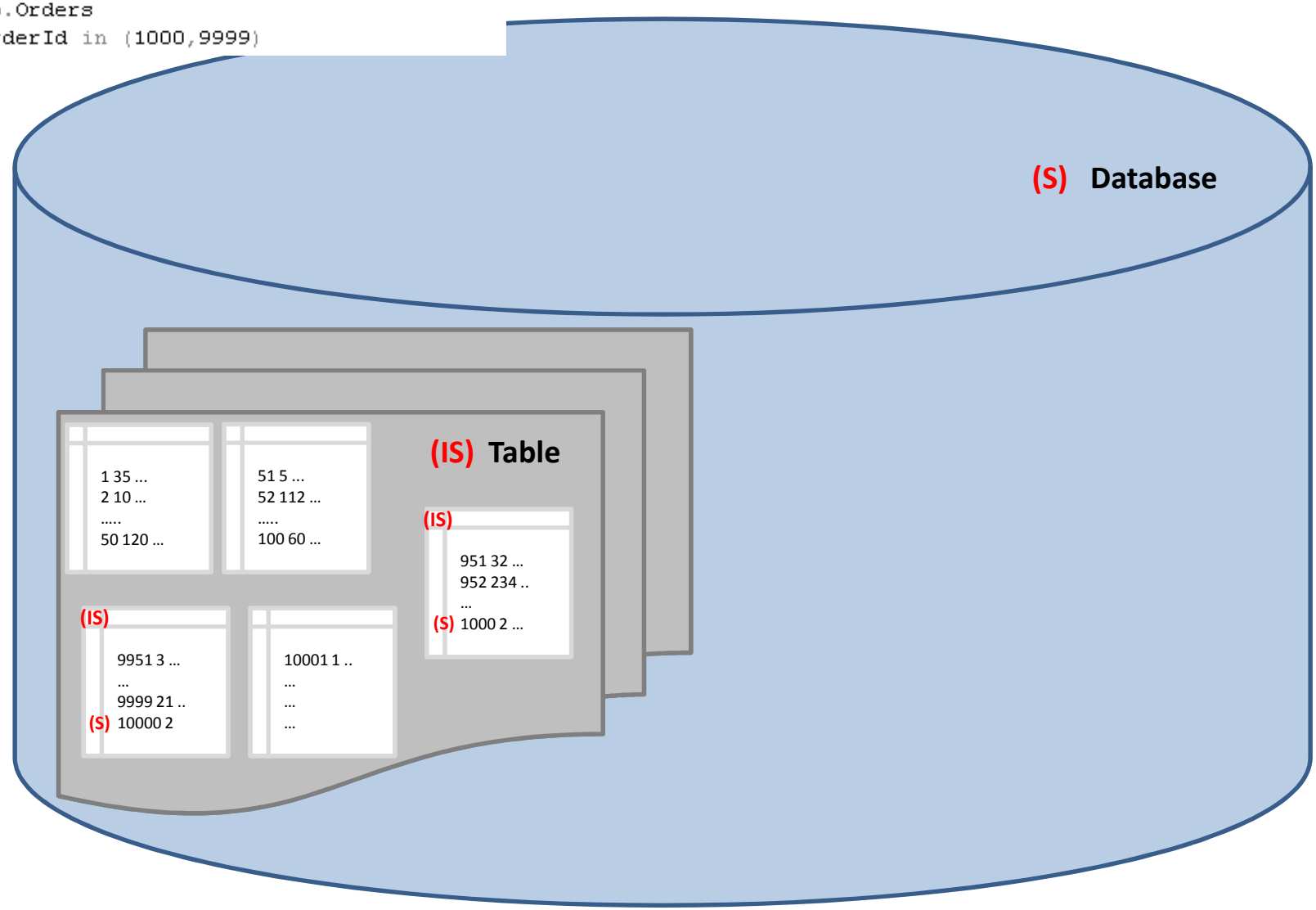
- Берутся «читателями» (SELECT)

```
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

Как оно работает

```
set transaction isolation level read committed

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



Совместимость Блокировок

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

- ✓ Разделяемые (S) блокировки совместимы с (S) и (U) и несовместимы с (X)

Разделяемые Блокировки

Уровень Изоляции	(S) Блокировки	Хинт
Read Uncommitted	(S) не берутся	NOLOCK
Read Committed	Берутся и сразу освобождаются	READCOMMITTED
Repeatable Read	Держатся до конца транзакции	REPEATABLEREAD
Serializable	**Блокировки на интервалы** держатся до конца транзакции	HOLDLOCK

- Проблемы:
  - Dirty Reads – сессия читает незакомиченные (грязные) данные
  - Non-Repeatable Reads – данные могут модифицироваться между чтениями
  - Phantom Reads & Ghost Rows – вставка и удаление данных между чтениями
- В некоторых случаях SQL Server «оптимизирует» блокировки
  - Пример: Read Committed - (S) на уровне страницы

Demo

УРОВНИ ИЗОЛЯЦИИ И БЛОКИРОВКИ

Диагностика

- Основной источник проблем – неоптимизированные запросы
- Два подхода
  - В реальном времени: DMVs (sys.dm_tran_locks)
  - Получение “Blocked Process Report”
    - SQL Trace
    - Extended Events
    - Event Notifications (Скрипт у меня в блоге)

Demo

ДИАГНОСТИКА БЛОКИРОВОК

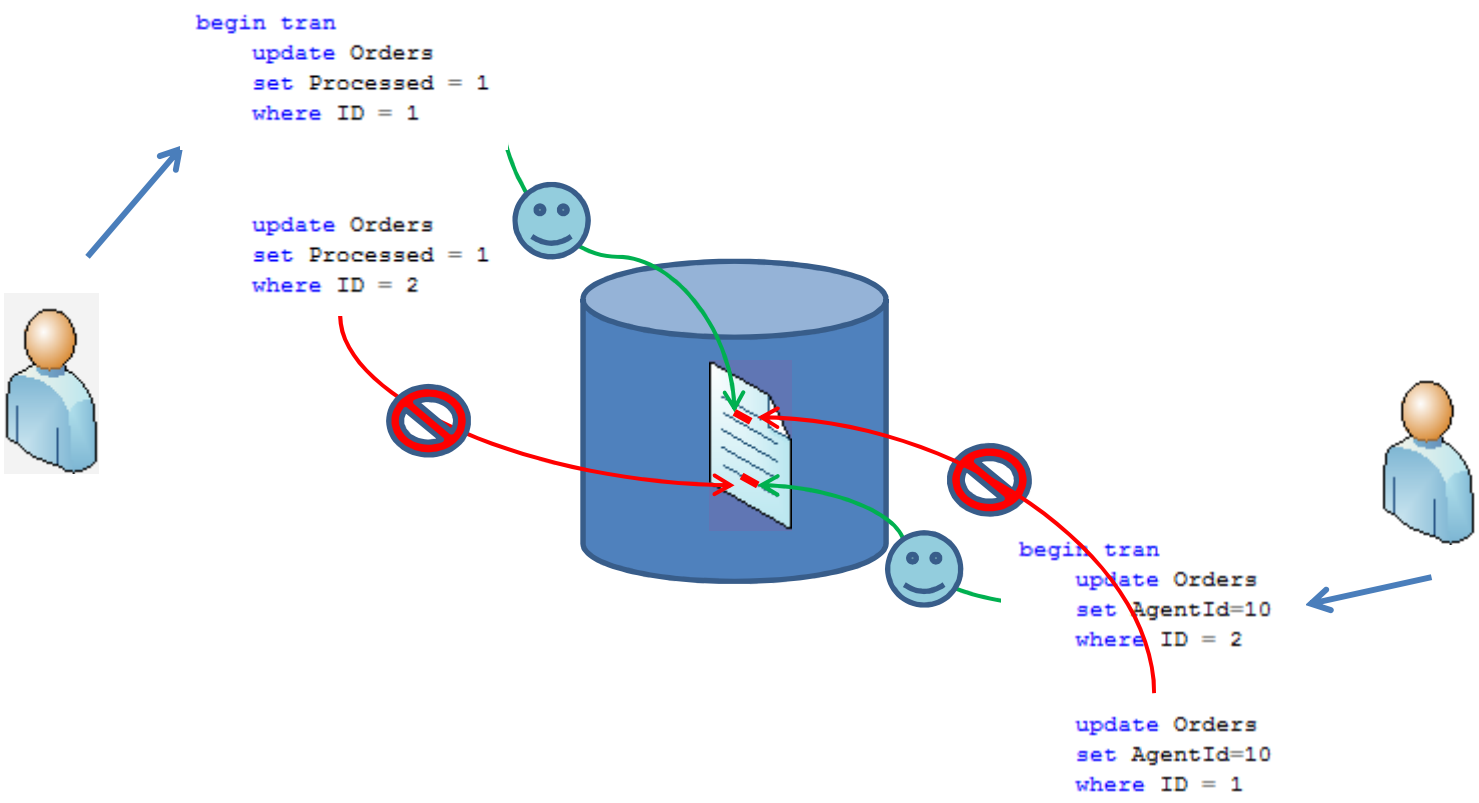
Handle…

```
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

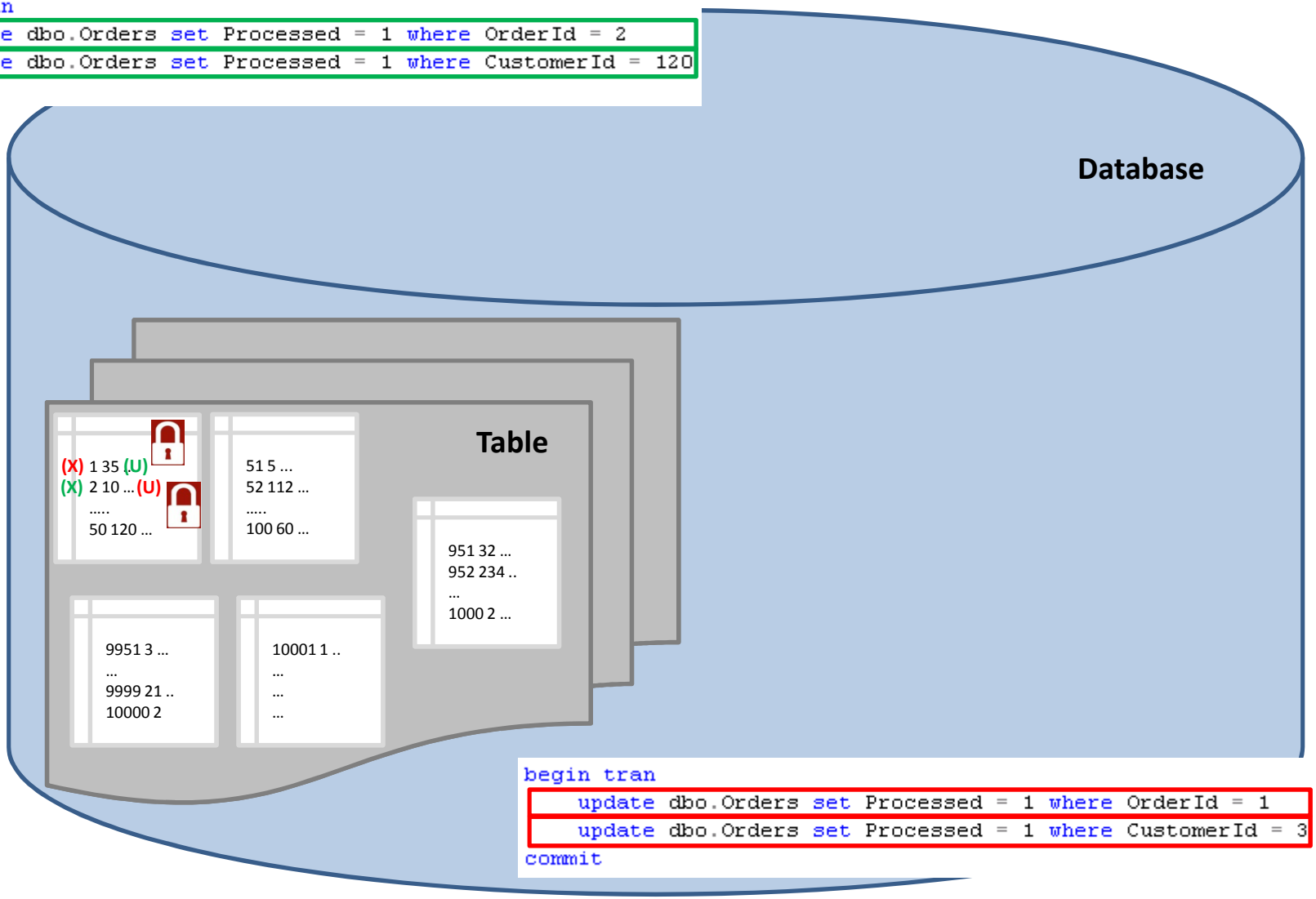
query_plan	(No column name)	sql_handle
<ShowPlanXML xmlns="http://schemas.microsoft.com...	insert into dbo.DataRecords(ID,Field1,Field2,Field3...	0x03000C00949D7F1

Классический Дедлок



Реальный Дедлок

```
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
```

Диагностика Дедлоков

- SQL Profiler – deadlock graph
- Trace flag 1222 – запись в error log

  DBCC TRACEON (1222, -1)
- Extended Events
  - Включаются в System_Health

Demo

ДЕДЛОКИ

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
```

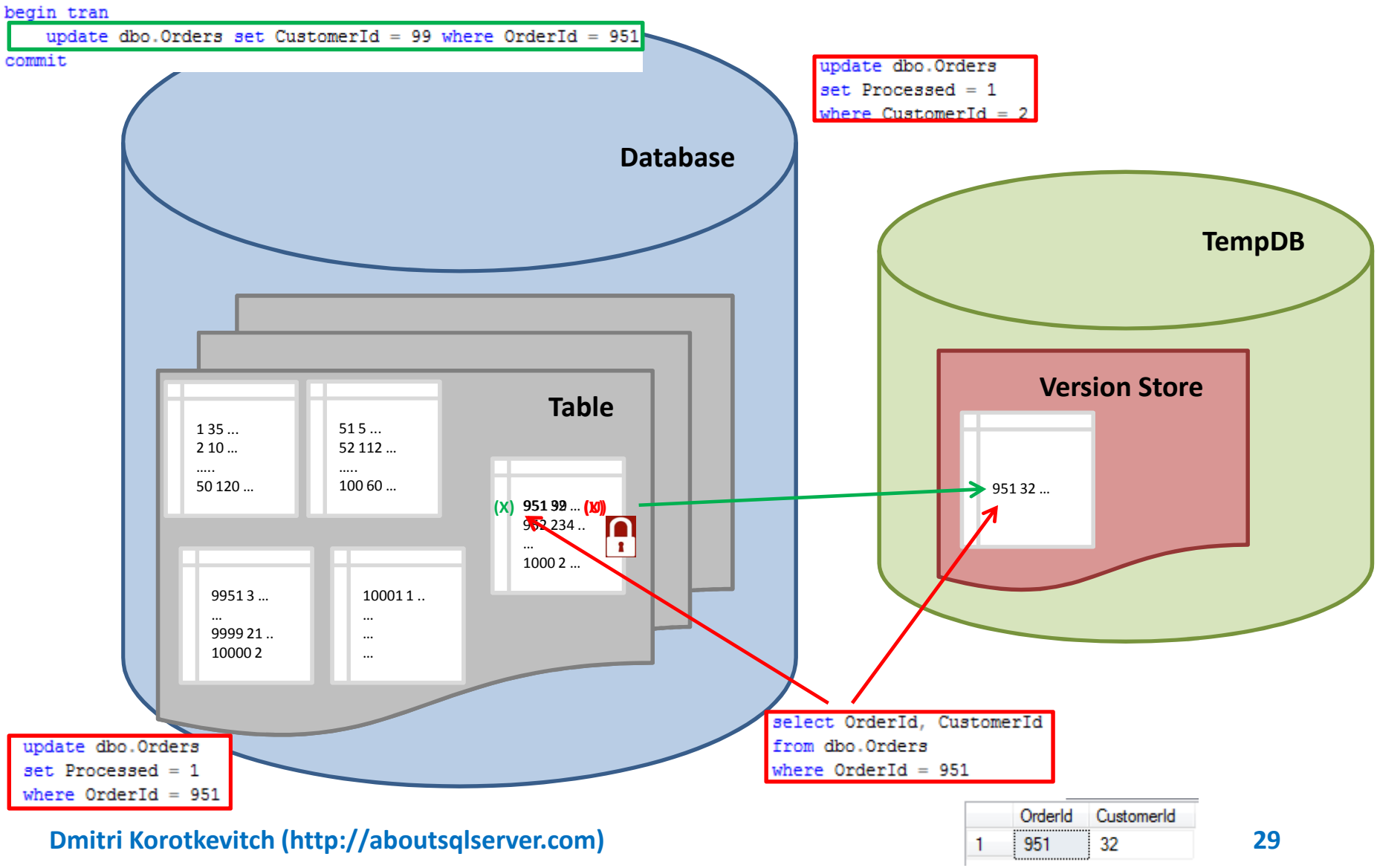
Эскалация Блокировок

- SQL Server производит эскалацию блокировок на уровень таблицы/секции
  - Первоначально: ~5,000 блокировок на объект
  - Если не получилось – попытка повторяется каждые ~1,250 блокировок
  - Это нормально. Пока не вызывает проблемы
- Проблемы: Пакетная операция вызывает эскалацию и блокирует остальные сессии.
- Один из признаков – большой процент ожиданий блокировок по намерению (LCK_M_I*) в статистике ожиданий
- Решения
  - T1211 (Уровень сервера) – не рекомендуется
  - SQL Server 2008+: *alter table .. set lock_escalation*
  - Оптимистичные уровни изоляции (версионность)

Demo

ЭСКАЛАЦИЯ БЛОКИРОВОК

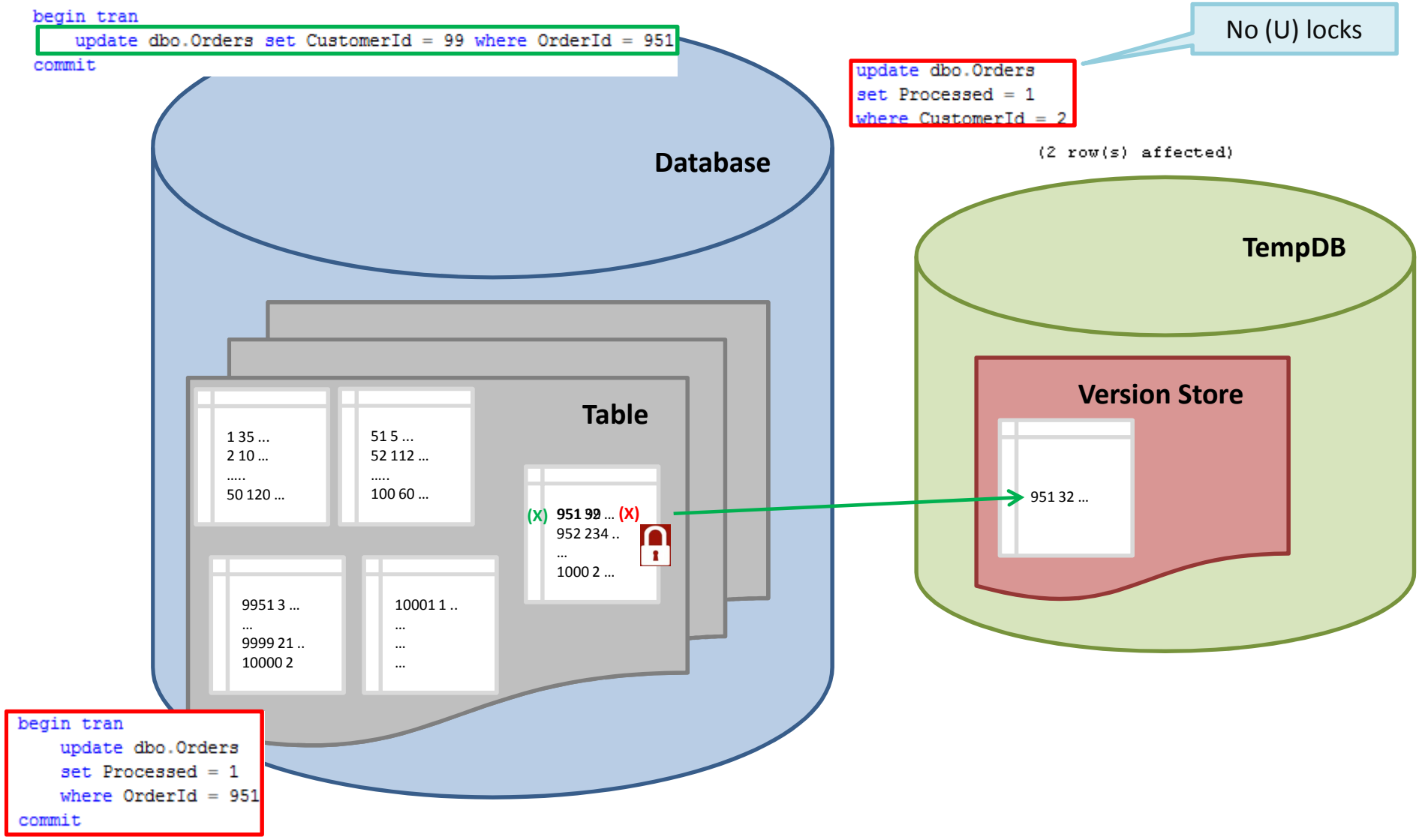
Read Committed Snapshot



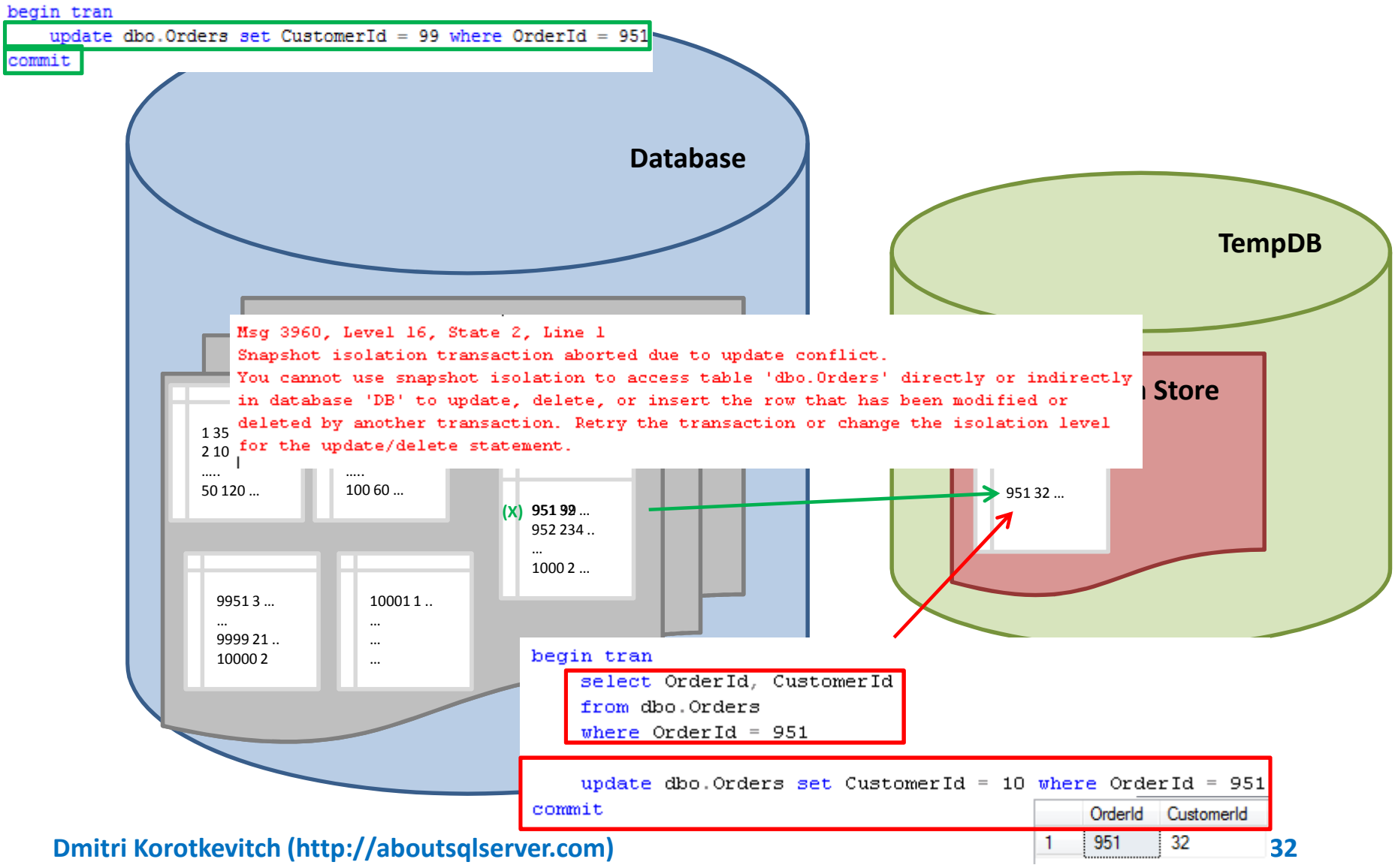
Demo

READ COMMITTED SNAPSHOT

Snapshot (U/X блокировки)



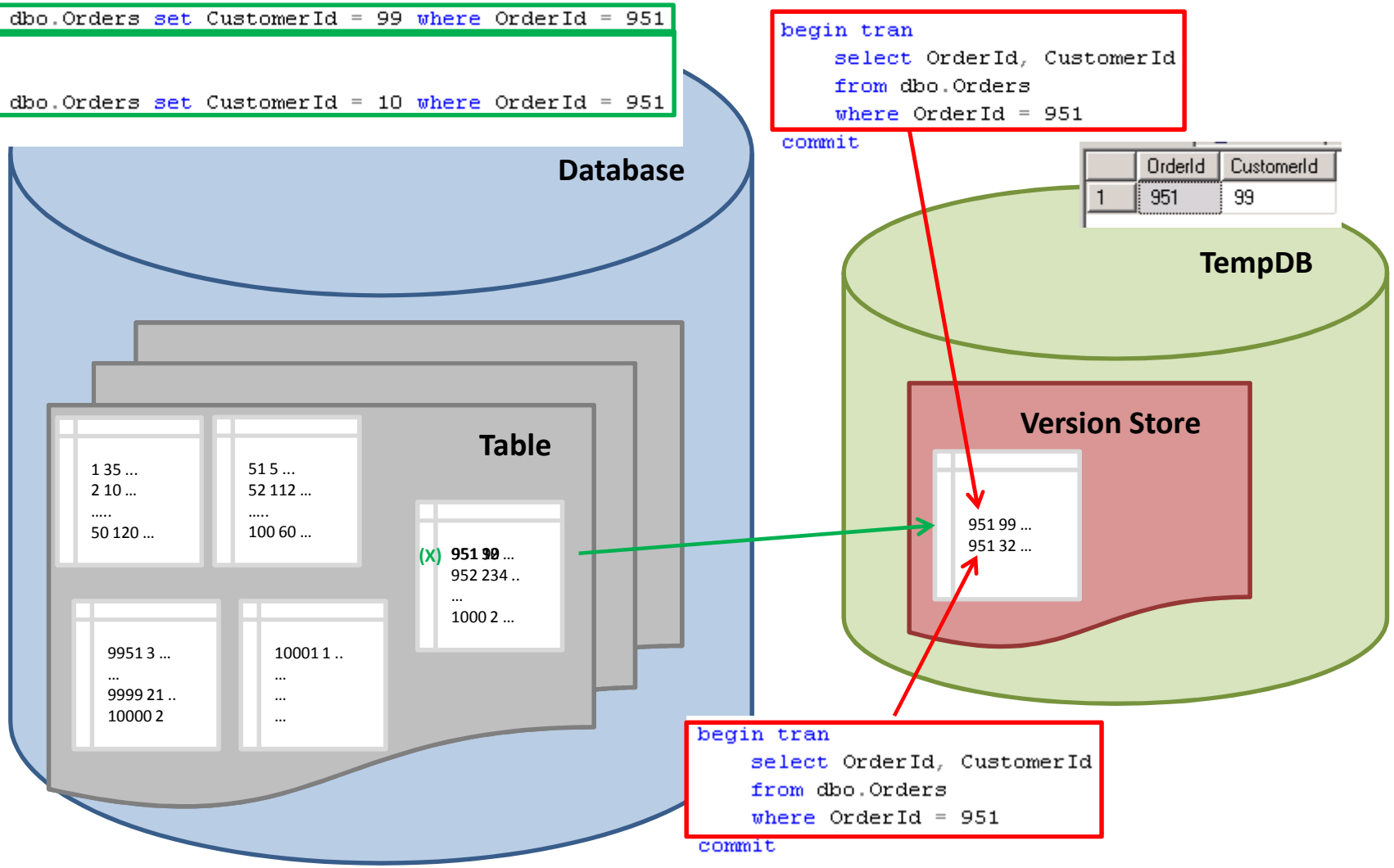
Snapshot (Updates)



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

Оптимистичные Уровни Изоляции

- Версионность
  - «Писатели» не блокируют «читателей»
- Read committed snapshot (Целостность данных на уровне запроса)
  - Не требует изменений в коде
  - Может помочь (как временное решение) при наличии проблем в системе
- Snapshot (Целостность данных на уровне транзакции)
- ЭЛДЭЭНБЭ (ланчей даром не бывает)
  - Увеличение размера записи (14-byte указатель)
  - **Увеличение фрагментации индексов**
    - Не используйте FILLFACTOR = 100
  - Нагрузка на Tempdb
  - Изменения в коде
    - Ссылочная целостность
    - Другое поведение при обновлении данных

Demo

ФРАГМЕНТАЦИЯ

Блокировки на уровне схемы

- Стабильность схемы (SCH-S) препятствует изменению схемы объекта
- DDL операции требуют блокировки модификации схемы (SCH-M)
  - Объект недоступен до окончания транзакции
- Проблемы:
  - Работа со схемами секционирования
  - Пересоздание индексов (даже online)
- В SQL Server 2014 есть low priority locks (вторая очередь блокировок)

Demo

БЛОКИРОВКИ НА УРОВНЕ СХЕМЫ

Выбор уровня изоляции

- It depends (зависит от других факторов) ☺
- Не используйте READ UNCOMMITTED если целостность данных имеет значение
- Read Committed – *OK* для OLTP если запросы оптимизированы
- Оптимистичные уровни изоляции
  - Идеальны для DW/Отчетов
  - Под вопросом для OLTP
    - RCSI?
      - Фрагментация индексов
      - Нагрузка на tempdb

Выводы

- 1e правило при решении проблем с блокировками – оптимизация
- 2e правило при решении проблем с блокировками – оптимизация
- 3e правило при решении проблем с блокировками – оптимизация
- Выбирайте правильный уровень изоляции
- Используйте короткие транзакции
- Не изменяйте запись несколько раз в одной транзакции
- Будьте осторожны с фреймворками
- http://aboutsqlserver.com – много информации