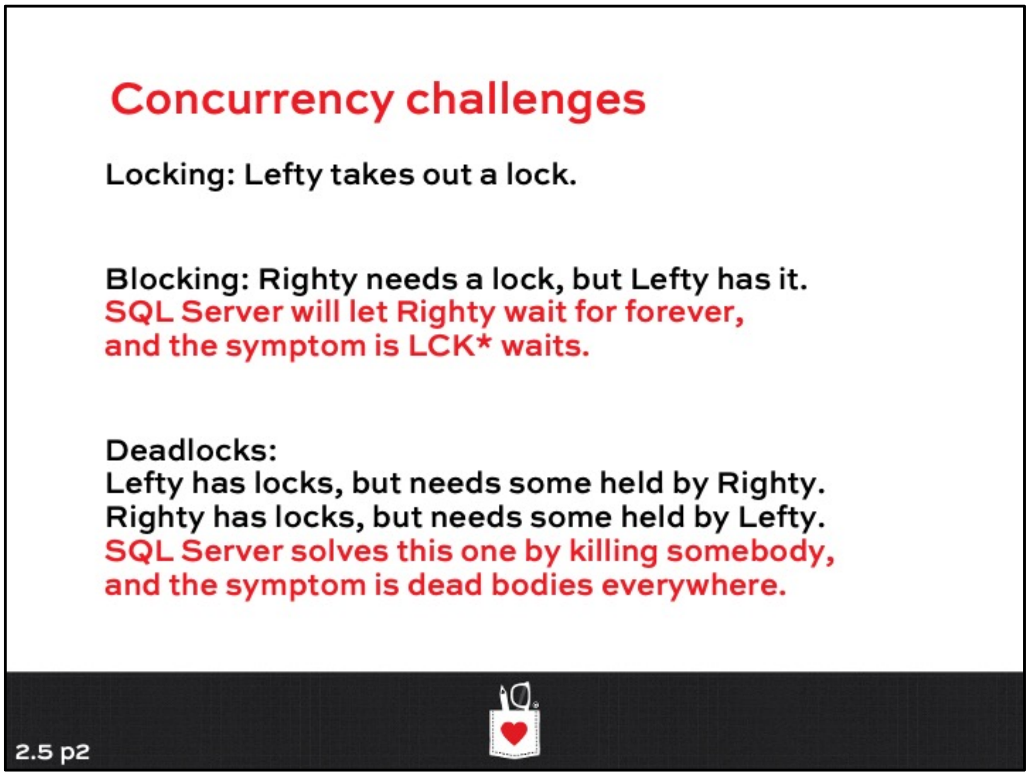
BRENT OZAR UNLIMITED®

How Indexes Help Avoid Blocking

2.5 p1



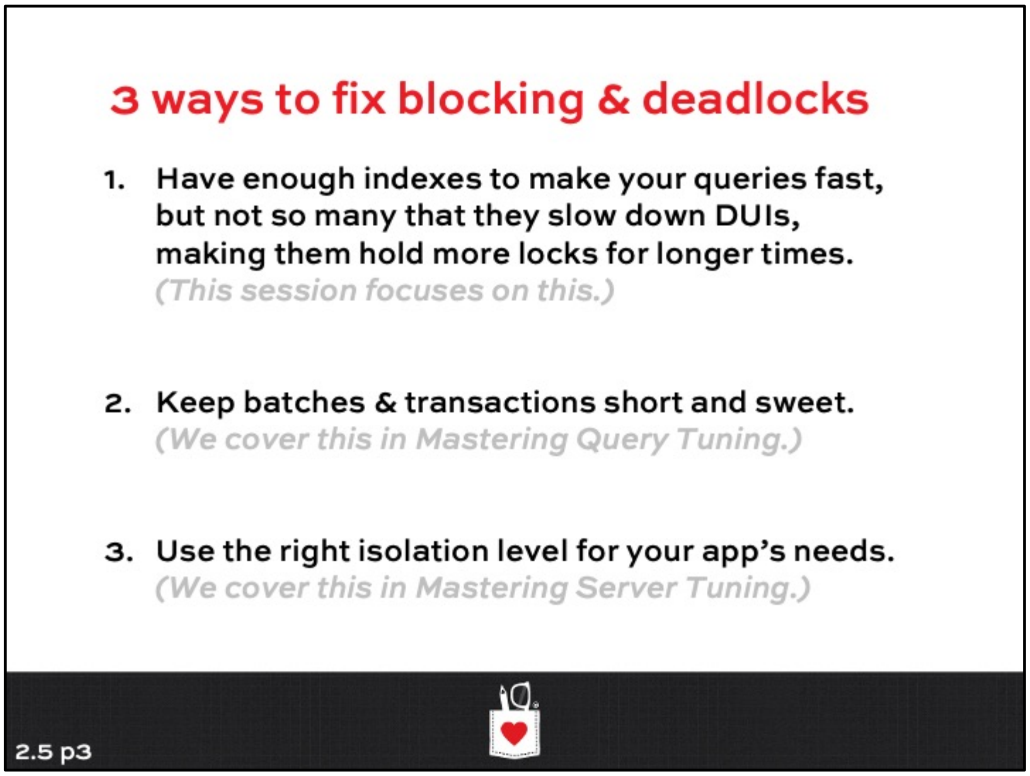
Concurrency challenges

Locking: Lefty takes out a lock.

Blocking: Righty needs a lock, but Lefty has it.
SQL Server will let Righty wait for forever,
and the symptom is LCK* waits.

Deadlocks:
Lefty has locks, but needs some held by Righty.
Righty has locks, but needs some held by Lefty.
SQL Server solves this one by killing somebody,
and the symptom is dead bodies everywhere.

2.5 p2



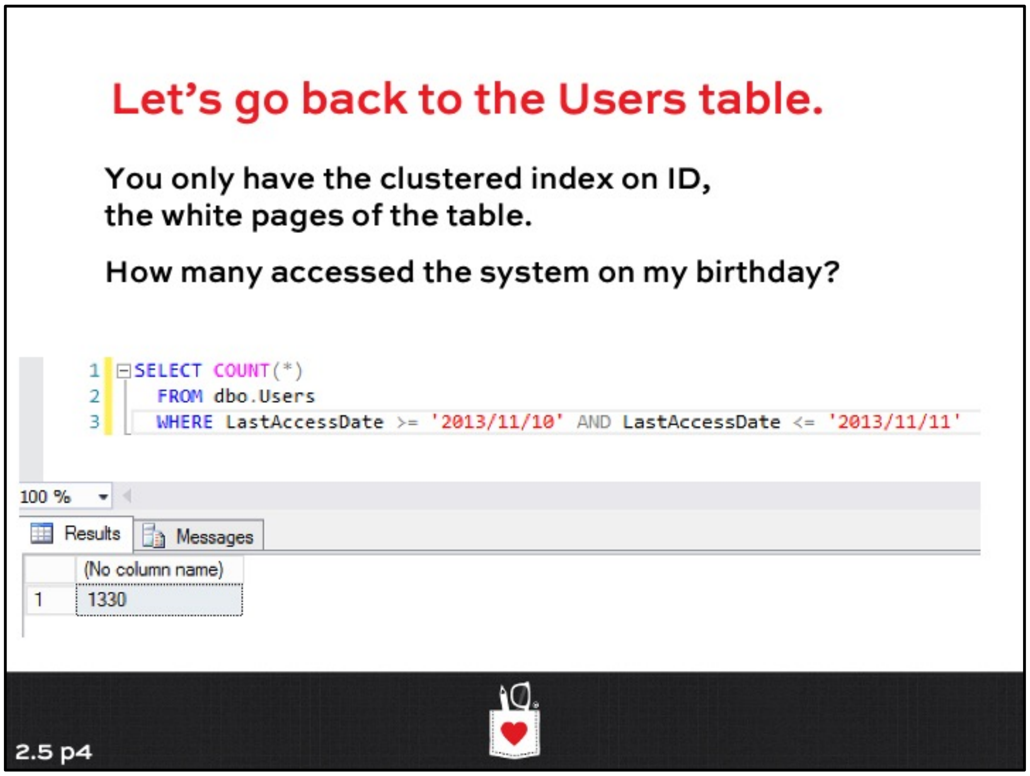
3 ways to fix blocking & deadlocks

1. Have enough indexes to make your queries fast, but not so many that they slow down DUIs, making them hold more locks for longer times.
   (This session focuses on this.)

2. Keep batches & transactions short and sweet.
   (We cover this in Mastering Query Tuning.)

3. Use the right isolation level for your app's needs.
   (We cover this in Mastering Server Tuning.)

2.5 p3



Let’s go back to the Users table.

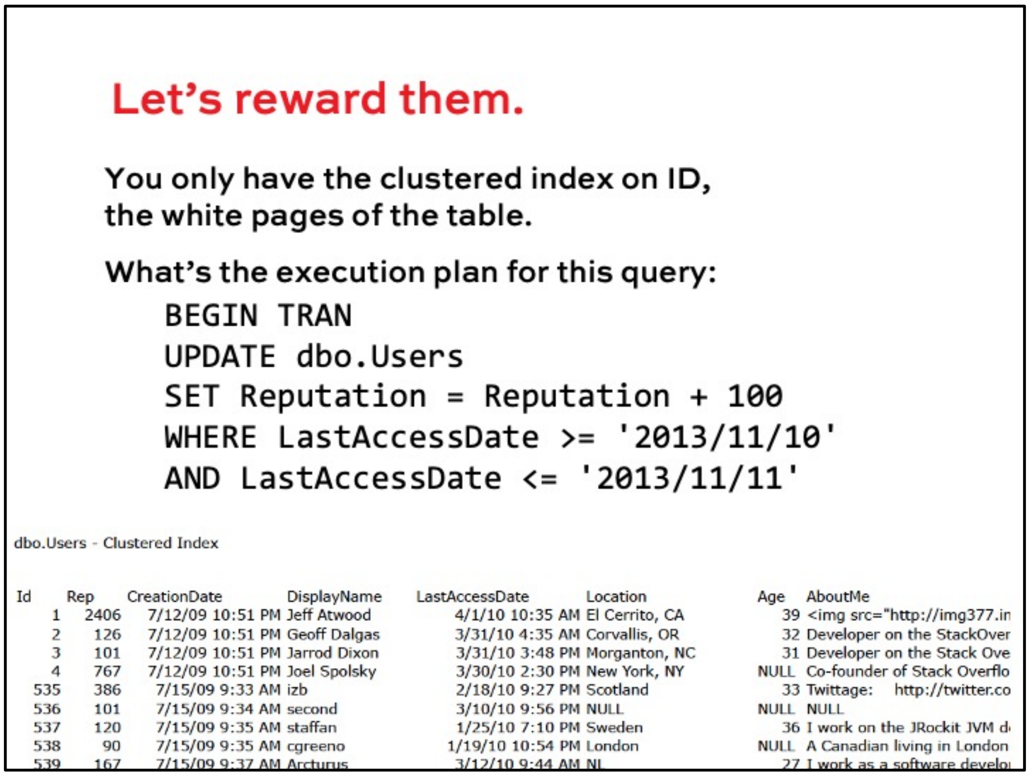
You only have the clustered index on ID, the white pages of the table.

How many accessed the system on my birthday?

```sql
SELECT COUNT(*)
  FROM dbo.Users
  WHERE LastAccessDate >= '2013/11/10' AND LastAccessDate <= '2013/11/11'
```

```
   (No column name)
1  1330
```

2.5 p4



Let’s reward them.

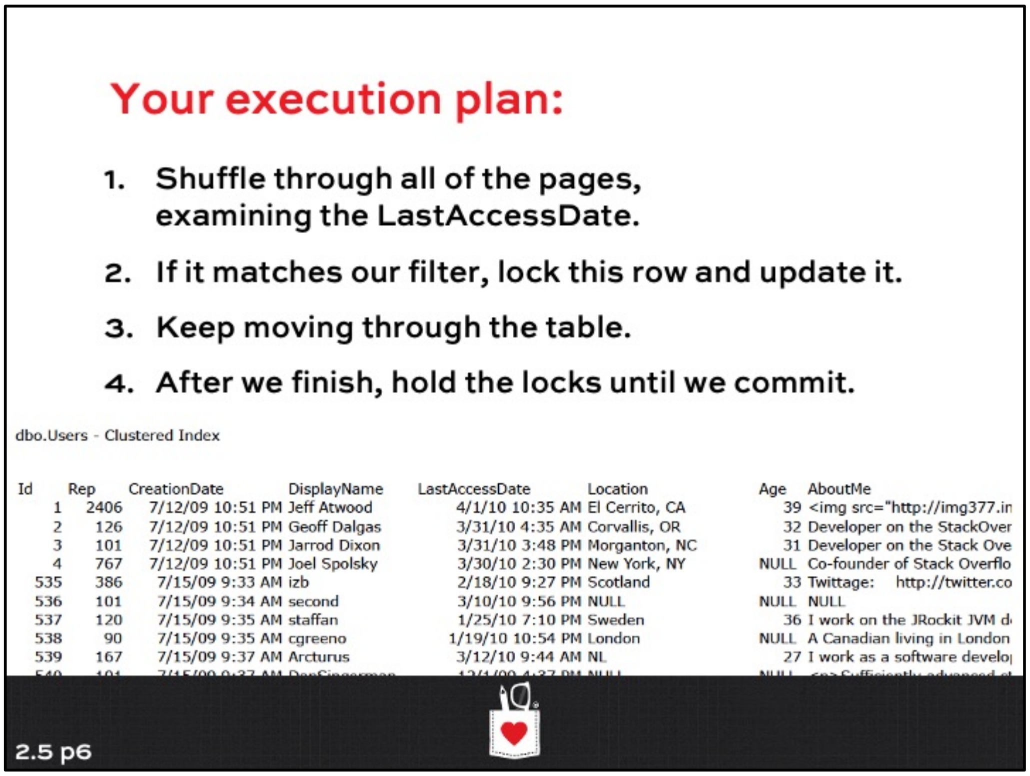
You only have the clustered index on ID, the white pages of the table.

What’s the execution plan for this query:

```
BEGIN TRAN
UPDATE dbo.Users
SET Reputation = Reputation + 100
WHERE LastAccessDate >= '2013/11/10'
AND LastAccessDate <= '2013/11/11'
```

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo



Your execution plan:

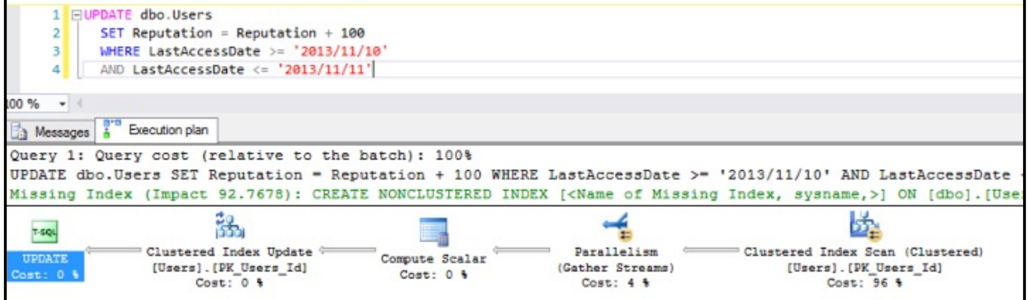
1. Shuffle through all of the pages, examining the LastAccessDate.
2. If it matches our filter, lock this row and update it.
3. Keep moving through the table.
4. After we finish, hold the locks until we commit.

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo

2.5 p6

SQL Server's execution plan

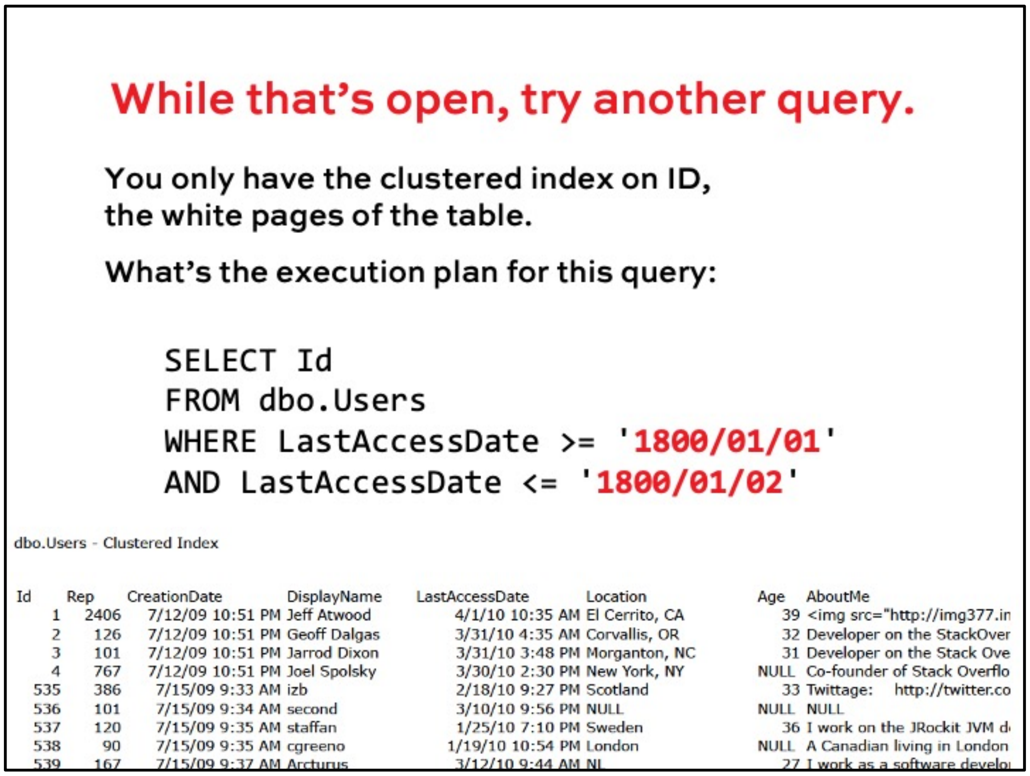


```
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11'
```

```
Query 1: Query cost (relative to the batch): 100%
UPDATE dbo.Users SET Reputation = Reputation + 100 WHERE LastAccessDate >= '2013/11/10' AND LastAccessDate
Missing Index (Impact 92.7678): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Use
```

It does show the clustered index scan, but doesn't show our locks.

2.5 p7



While that’s open, try another query.

You only have the clustered index on ID, the white pages of the table.

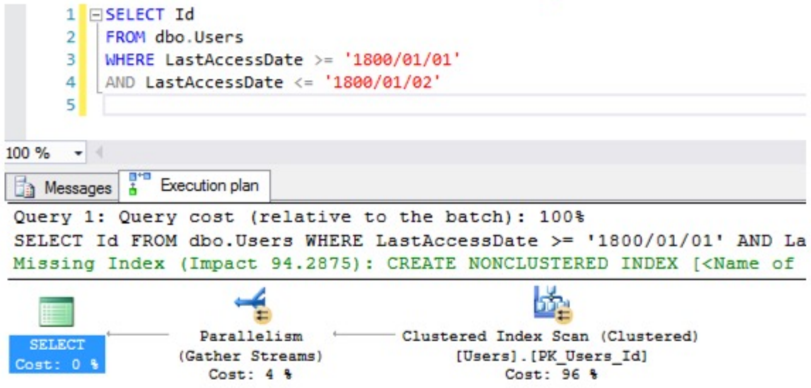
What’s the execution plan for this query:

```
SELECT Id
FROM dbo.Users
WHERE LastAccessDate >= '1800/01/01'
AND LastAccessDate <= '1800/01/02'
```

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo

SQL Server's execution plan



```
SELECT Id
FROM dbo.Users
WHERE LastAccessDate >= '1800/01/01'
AND LastAccessDate <= '1800/01/02'
```

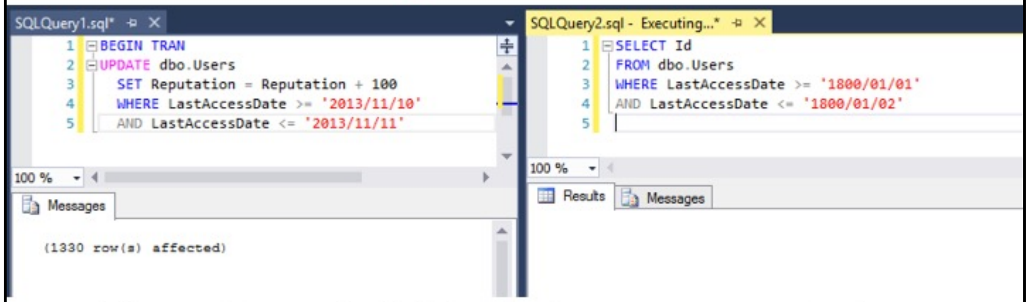
```
Query 1: Query cost (relative to the batch): 100%
SELECT Id FROM dbo.Users WHERE LastAccessDate >= '1800/01/01' AND La
Missing Index (Impact 94.2875): CREATE NONCLUSTERED INDEX [<Name of
```

We don't have an index on LastAccessDate, so we're going to have to scan the table for this.

If we run it, what happens?

2.5 p9

Our SELECT just sits there blocked.



```
BEGIN TRAN
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11'
```

```
(1330 row(s) affected)
```

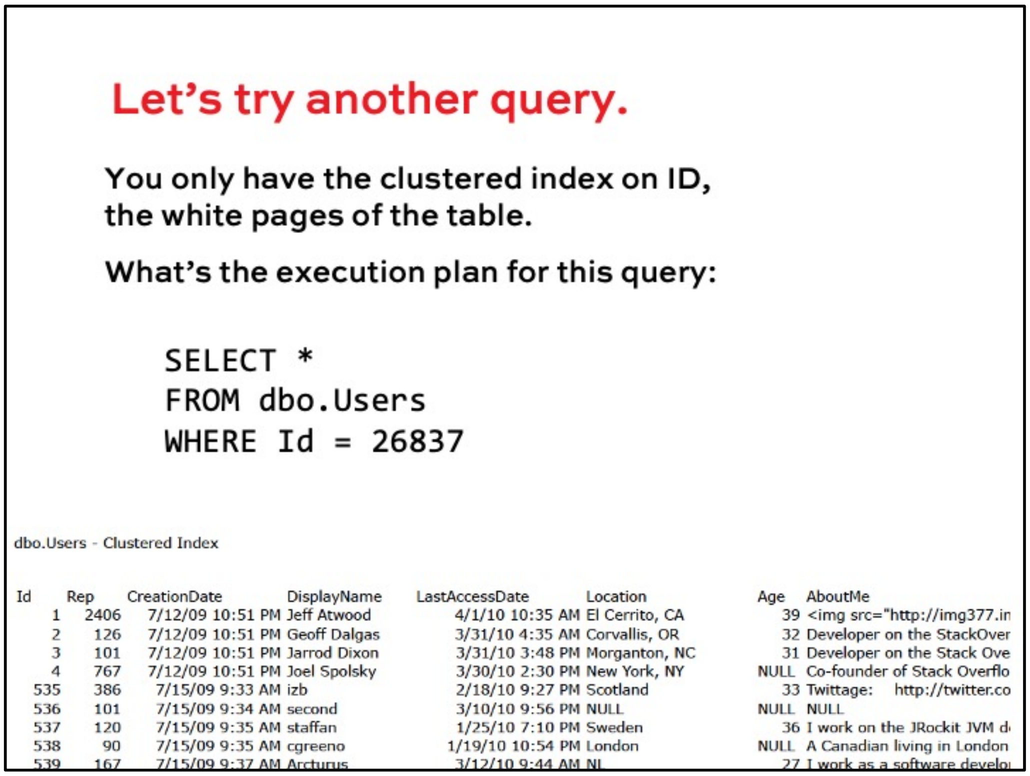
```
SELECT Id
FROM dbo.Users
WHERE LastAccessDate >= '1800/01/01'
AND LastAccessDate <= '1800/01/02'
```

The update on the left has locks on some rows in the Users table.

Until that lock is released, we don't know whether the locked rows match our WHERE clause filter.

So we'll wait. Forever.

2.5 p10



Let's try another query.

You only have the clustered index on ID, the white pages of the table.

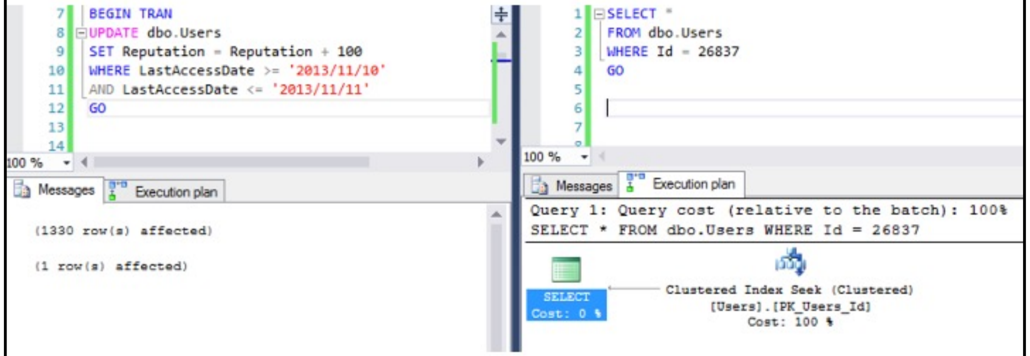
What's the execution plan for this query:

```
SELECT *
FROM dbo.Users
WHERE Id = 26837
```

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo

Will it work?



```
BEGIN TRAN
UPDATE dbo.Users
SET Reputation = Reputation + 100
WHERE LastAccessDate >= '2013/11/10'
AND LastAccessDate <= '2013/11/11'
GO
```

```
(1330 row(s) affected)

(1 row(s) affected)
```

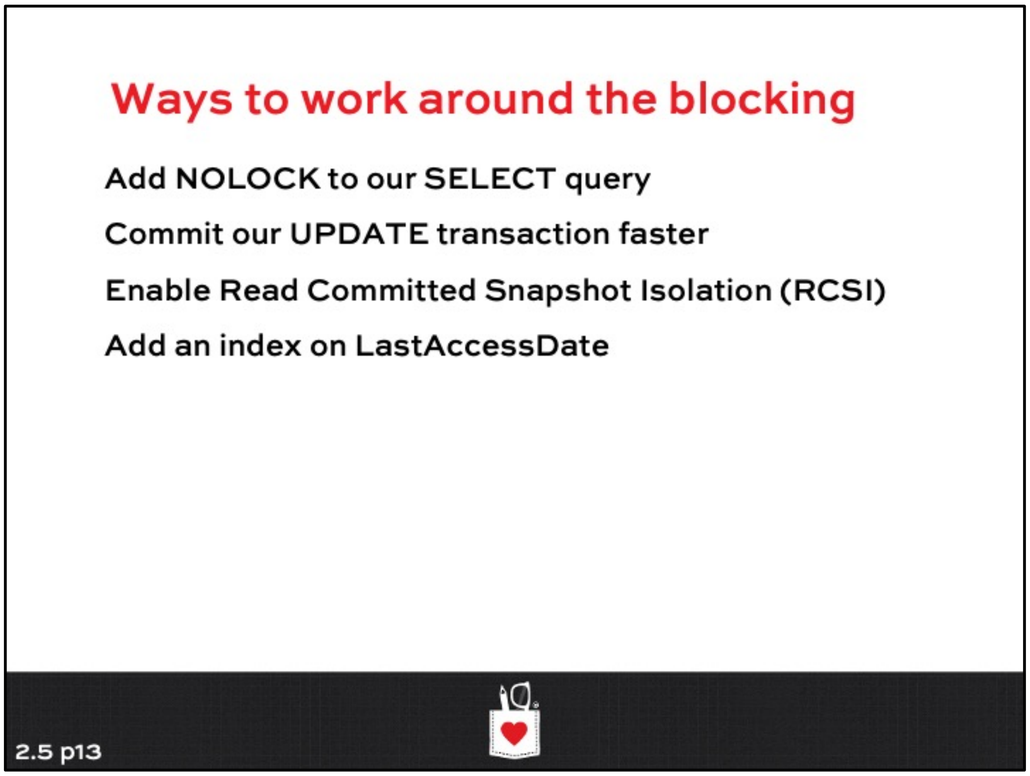
```
SELECT *
FROM dbo.Users
WHERE Id = 26837
GO
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM dbo.Users WHERE Id = 26837
```

Some of the rows of the clustered index (white pages) are locked, but not Brent's row.

Can Brent seek in and get unlocked data?

2.5 p12



Ways to work around the blocking

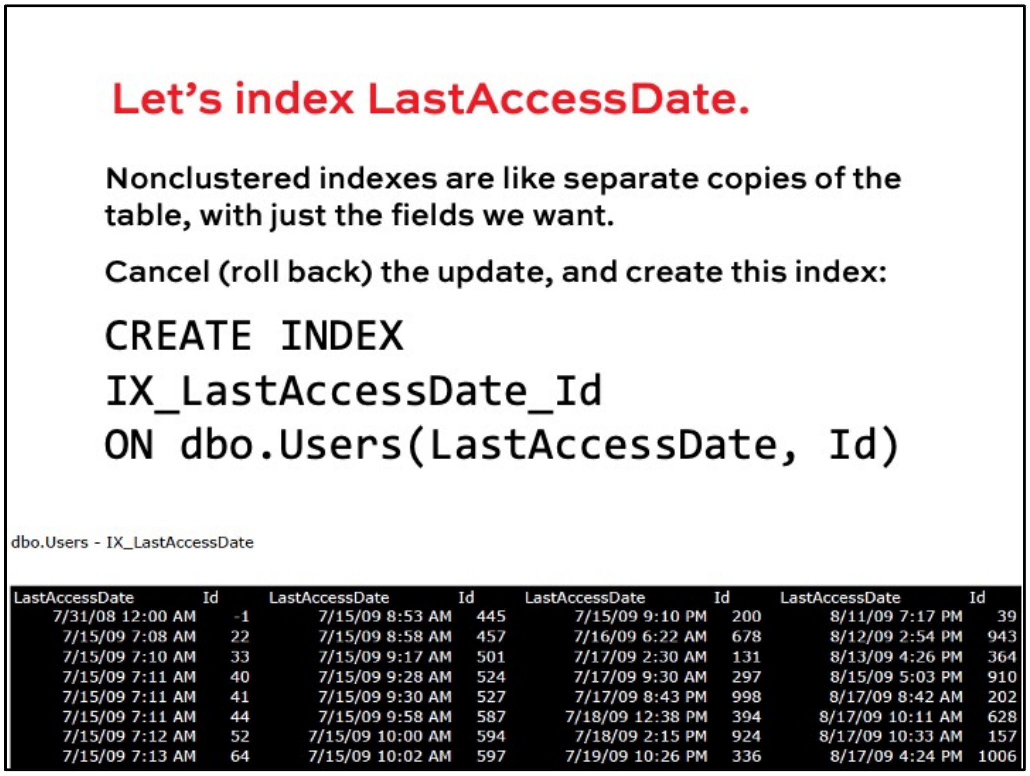
Add NOLOCK to our SELECT query

Commit our UPDATE transaction faster

Enable Read Committed Snapshot Isolation (RCSI)

Add an index on LastAccessDate

2.5 p13



Let’s index LastAccessDate.

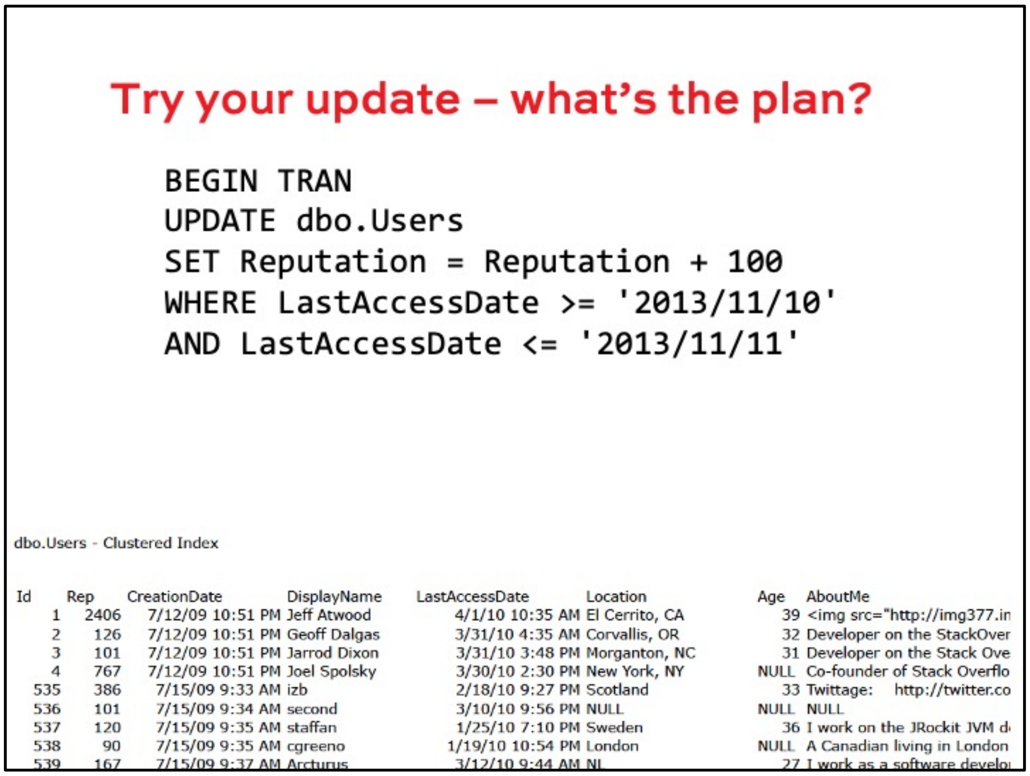
Nonclustered indexes are like separate copies of the table, with just the fields we want.

Cancel (roll back) the update, and create this index:

```
CREATE INDEX
IX_LastAccessDate_Id
ON dbo.Users(LastAccessDate, Id)
```

dbo.Users - IX_LastAccessDate

LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id
7/31/08 12:00 AM	-1	7/15/09 8:53 AM	445	7/15/09 9:10 PM	200	8/11/09 7:17 PM	39
7/15/09 7:08 AM	22	7/15/09 8:58 AM	457	7/16/09 6:22 AM	678	8/12/09 2:54 PM	943
7/15/09 7:10 AM	33	7/15/09 9:17 AM	501	7/17/09 2:30 AM	131	8/13/09 4:26 PM	364
7/15/09 7:11 AM	40	7/15/09 9:28 AM	524	7/17/09 9:30 AM	297	8/15/09 5:03 PM	910
7/15/09 7:11 AM	41	7/15/09 9:30 AM	527	7/17/09 8:43 PM	998	8/17/09 8:42 AM	202
7/15/09 7:11 AM	44	7/15/09 9:58 AM	587	7/18/09 12:38 PM	394	8/17/09 10:11 AM	628
7/15/09 7:12 AM	52	7/15/09 10:00 AM	594	7/18/09 2:15 PM	924	8/17/09 10:33 AM	157
7/15/09 7:13 AM	64	7/15/09 10:02 AM	597	7/19/09 10:26 PM	336	8/17/09 4:24 PM	1006

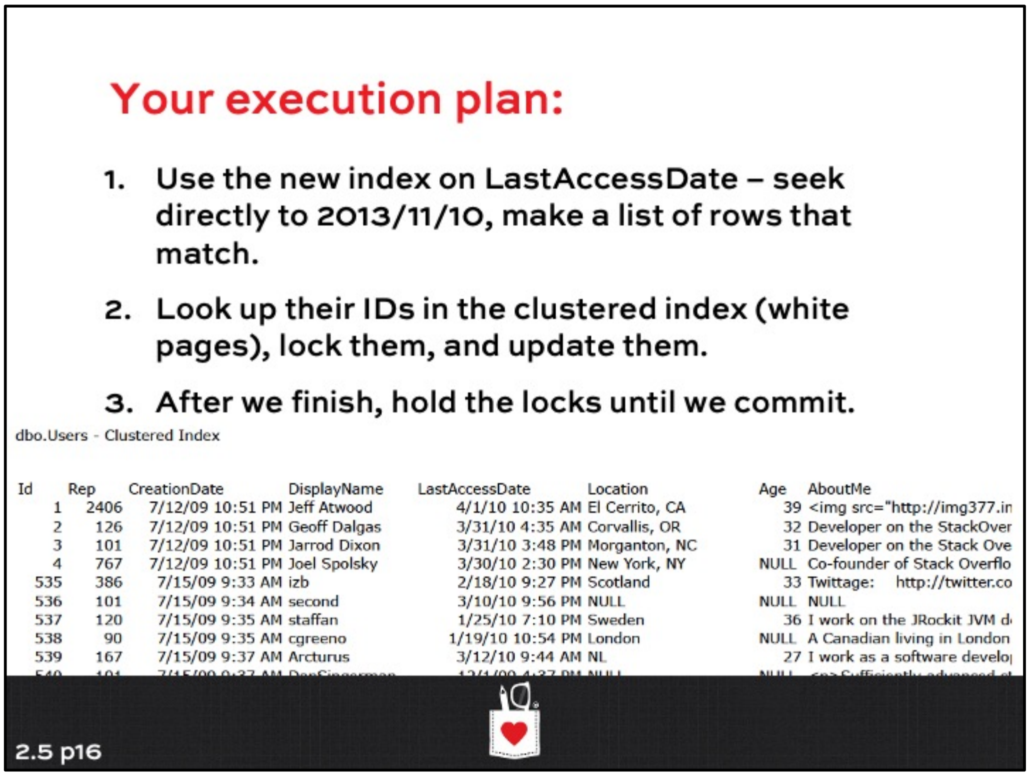


Try your update – what's the plan?

```
BEGIN TRAN
UPDATE dbo.Users
SET Reputation = Reputation + 100
WHERE LastAccessDate >= '2013/11/10'
AND LastAccessDate <= '2013/11/11'
```

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo



Your execution plan:

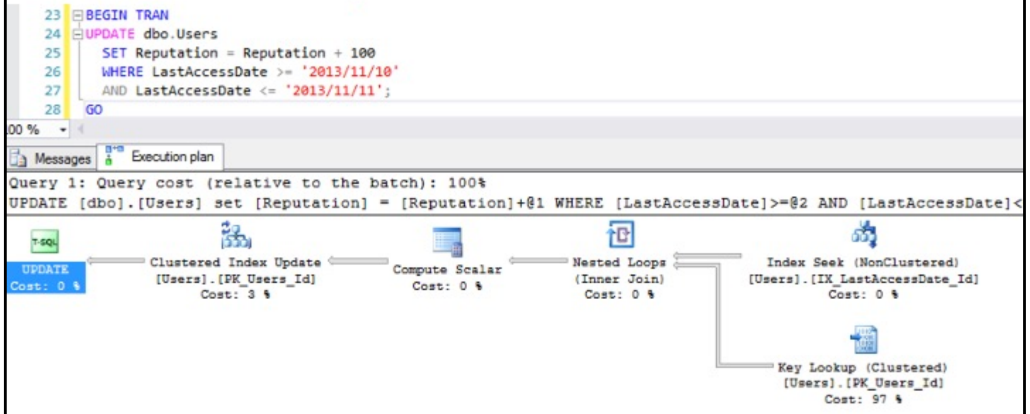
1. Use the new index on LastAccessDate – seek directly to 2013/11/10, make a list of rows that match.
2. Look up their IDs in the clustered index (white pages), lock them, and update them.
3. After we finish, hold the locks until we commit.

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo

2.5 p16

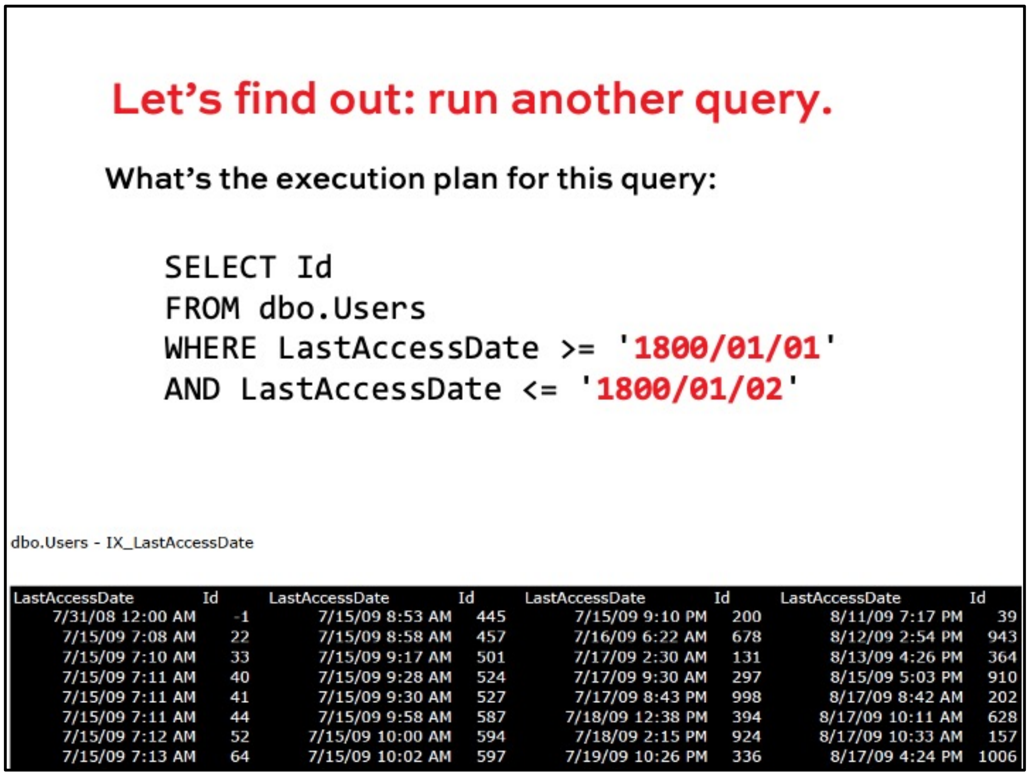
Here's the plan.



```
BEGIN TRAN
UPDATE dbo.Users
   SET Reputation = Reputation + 100
   WHERE LastAccessDate >= '2013/11/10'
   AND LastAccessDate <= '2013/11/11';
GO
```

```
Query 1: Query cost (relative to the batch): 100%
UPDATE [dbo].[Users] set [Reputation] = [Reputation]+@1 WHERE [LastAccessDate]>=@2 AND [LastAccessDate]<
```

But it doesn't show locks.
Do you need to lock the index (black pages)?



Let’s find out: run another query.

What’s the execution plan for this query:

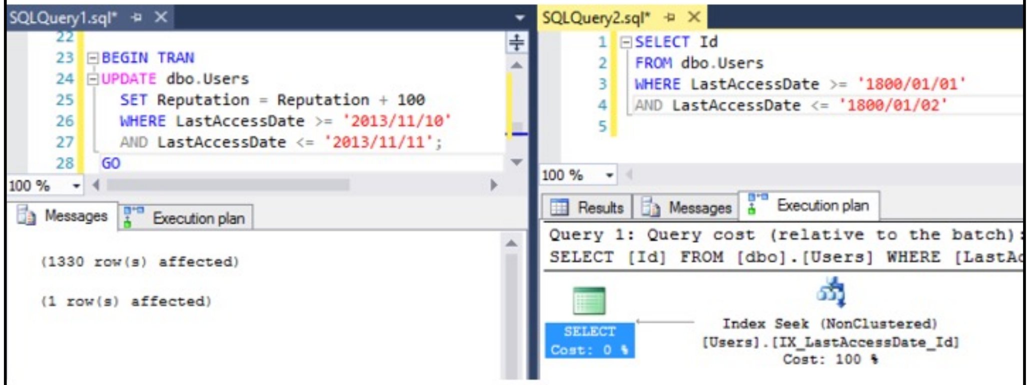
```
SELECT Id
FROM dbo.Users
WHERE LastAccessDate >= '1800/01/01'
AND LastAccessDate <= '1800/01/02'
```

dbo.Users - IX_LastAccessDate

LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id
7/31/08 12:00 AM	-1	7/15/09 8:53 AM	445	7/15/09 9:10 PM	200	8/11/09 7:17 PM	39
7/15/09 7:08 AM	22	7/15/09 8:58 AM	457	7/16/09 6:22 AM	678	8/12/09 2:54 PM	943
7/15/09 7:10 AM	33	7/15/09 9:17 AM	501	7/17/09 2:30 AM	131	8/13/09 4:26 PM	364
7/15/09 7:11 AM	40	7/15/09 9:28 AM	524	7/17/09 9:30 AM	297	8/15/09 5:03 PM	910
7/15/09 7:11 AM	41	7/15/09 9:30 AM	527	7/17/09 8:43 PM	998	8/17/09 8:42 AM	202
7/15/09 7:11 AM	44	7/15/09 9:58 AM	587	7/18/09 12:38 PM	394	8/17/09 10:11 AM	628
7/15/09 7:12 AM	52	7/15/09 10:00 AM	594	7/18/09 2:15 PM	924	8/17/09 10:33 AM	157
7/15/09 7:13 AM	64	7/15/09 10:02 AM	597	7/19/09 10:26 PM	336	8/17/09 4:24 PM	1006

The SELECT finishes immediately.

Presto! The black pages weren't locked.



```
BEGIN TRAN
UPDATE dbo.Users
   SET Reputation = Reputation + 100
   WHERE LastAccessDate >= '2013/11/10'
   AND LastAccessDate <= '2013/11/11';
GO
```

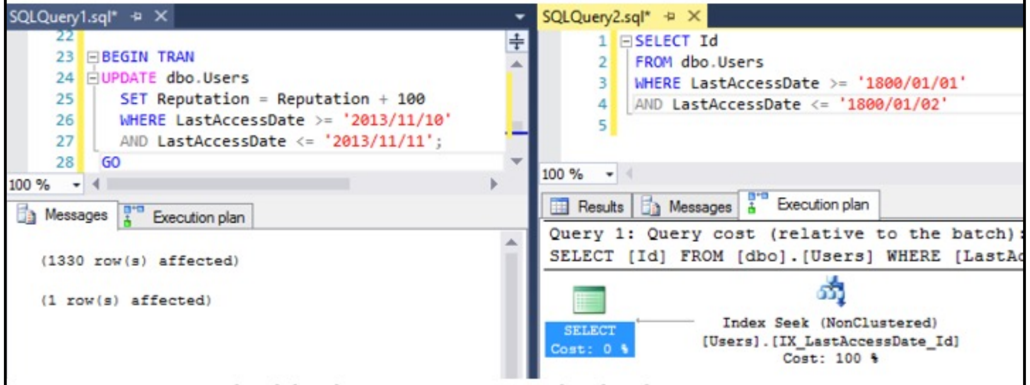
```
(1330 row(s) affected)

(1 row(s) affected)
```

```
SELECT Id
FROM dbo.Users
WHERE LastAccessDate >= '1800/01/01'
AND LastAccessDate <= '1800/01/02'
```

```
Query 1: Query cost (relative to the batch):
SELECT [Id] FROM [dbo].[Users] WHERE [LastAc
```

2.5 p19

The SELECT finishes immediately.


SQLQuery1.sql*

```
BEGIN TRAN
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11';
GO
```

```
(1330 row(s) affected)

(1 row(s) affected)
```

SQLQuery2.sql*

```
SELECT Id
FROM dbo.Users
WHERE LastAccessDate >= '1800/01/01'
AND LastAccessDate <= '1800/01/02'
```

```
Query 1: Query cost (relative to the batch):
SELECT [Id] FROM [dbo].[Users] WHERE [LastAc
```

Presto! The black pages weren't locked.

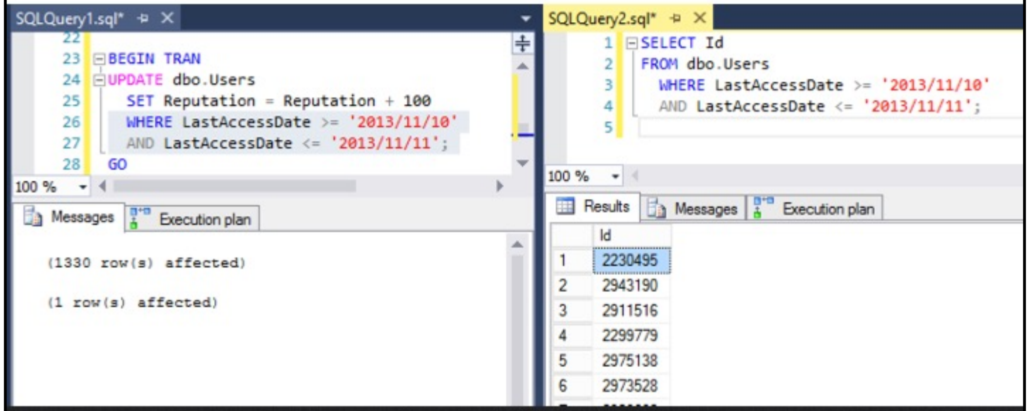
But what if we query the same dates that we're updating?

2.5 p20

Still no blocking.

Our SELECT finishes instantly.

Indexes are amazing and the cure to all ills.



SQLQuery1.sql*

```
BEGIN TRAN
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11';
GO
```

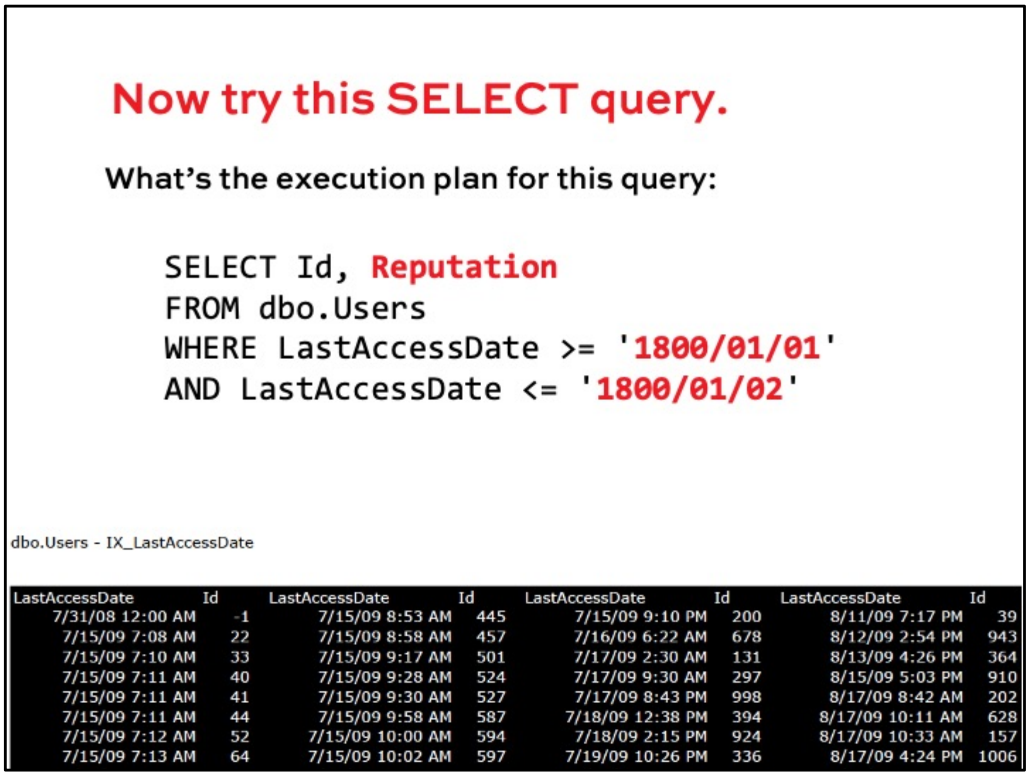
```
(1330 row(s) affected)

(1 row(s) affected)
```

SQLQuery2.sql*

```
SELECT Id
FROM dbo.Users
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11';
```

	Id
1	2230495
2	2943190
3	2911516
4	2299779
5	2975138
6	2973528

2.5 p21



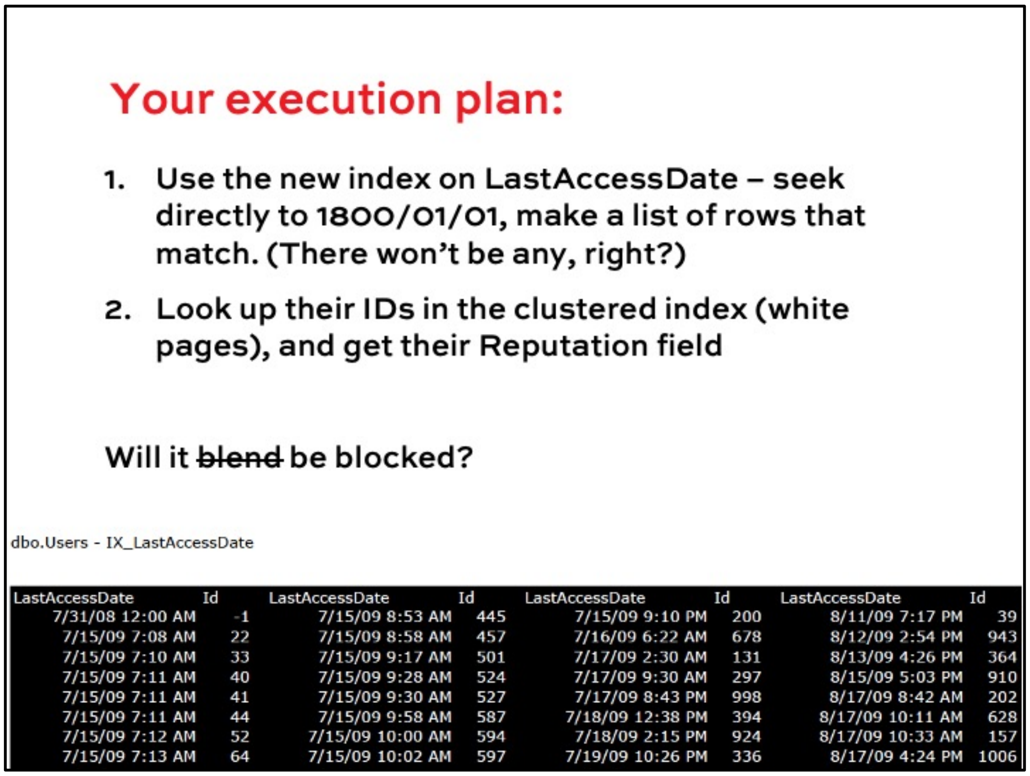
Now try this SELECT query.

What's the execution plan for this query:

```
SELECT Id, Reputation
FROM dbo.Users
WHERE LastAccessDate >= '1800/01/01'
AND LastAccessDate <= '1800/01/02'
```

dbo.Users - IX_LastAccessDate

LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id
7/31/08 12:00 AM	-1	7/15/09 8:53 AM	445	7/15/09 9:10 PM	200	8/11/09 7:17 PM	39
7/15/09 7:08 AM	22	7/15/09 8:58 AM	457	7/16/09 6:22 AM	678	8/12/09 2:54 PM	943
7/15/09 7:10 AM	33	7/15/09 9:17 AM	501	7/17/09 2:30 AM	131	8/13/09 4:26 PM	364
7/15/09 7:11 AM	40	7/15/09 9:28 AM	524	7/17/09 9:30 AM	297	8/15/09 5:03 PM	910
7/15/09 7:11 AM	41	7/15/09 9:30 AM	527	7/17/09 8:43 PM	998	8/17/09 8:42 AM	202
7/15/09 7:11 AM	44	7/15/09 9:58 AM	587	7/18/09 12:38 PM	394	8/17/09 10:11 AM	628
7/15/09 7:12 AM	52	7/15/09 10:00 AM	594	7/18/09 2:15 PM	924	8/17/09 10:33 AM	157
7/15/09 7:13 AM	64	7/15/09 10:02 AM	597	7/19/09 10:26 PM	336	8/17/09 4:24 PM	1006



Your execution plan:

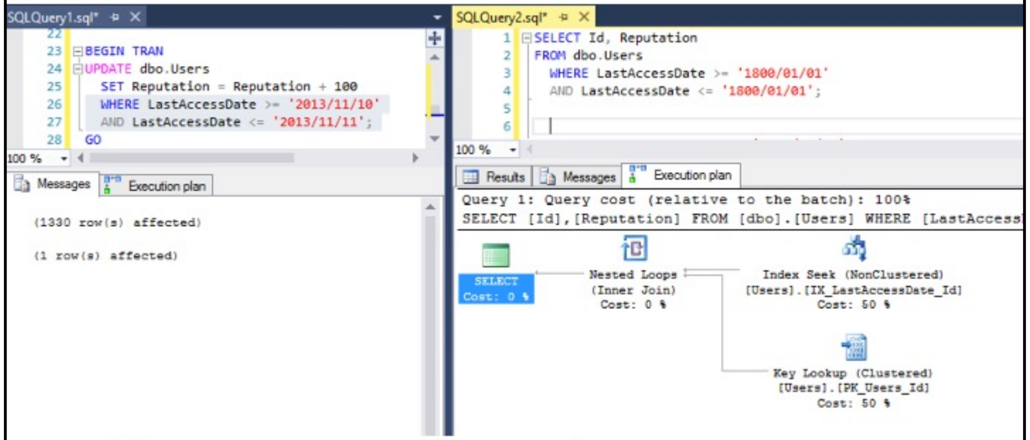
1. Use the new index on LastAccessDate – seek directly to 1800/01/01, make a list of rows that match. (There won't be any, right?)
2. Look up their IDs in the clustered index (white pages), and get their Reputation field

Will it ~~blend~~ be blocked?

dbo.Users - IX_LastAccessDate

LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id
7/31/08 12:00 AM	-1	7/15/09 8:53 AM	445	7/15/09 9:10 PM	200	8/11/09 7:17 PM	39
7/15/09 7:08 AM	22	7/15/09 8:58 AM	457	7/16/09 6:22 AM	678	8/12/09 2:54 PM	943
7/15/09 7:10 AM	33	7/15/09 9:17 AM	501	7/17/09 2:30 AM	131	8/13/09 4:26 PM	364
7/15/09 7:11 AM	40	7/15/09 9:28 AM	524	7/17/09 9:30 AM	297	8/15/09 5:03 PM	910
7/15/09 7:11 AM	41	7/15/09 9:30 AM	527	7/17/09 8:43 PM	998	8/17/09 8:42 AM	202
7/15/09 7:11 AM	44	7/15/09 9:58 AM	587	7/18/09 12:38 PM	394	8/17/09 10:11 AM	628
7/15/09 7:12 AM	52	7/15/09 10:00 AM	594	7/18/09 2:15 PM	924	8/17/09 10:33 AM	157
7/15/09 7:13 AM	64	7/15/09 10:02 AM	597	7/19/09 10:26 PM	336	8/17/09 4:24 PM	1006

It finishes instantly!



```
BEGIN TRAN
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11';
GO
```

```
(1330 row(s) affected)

(1 row(s) affected)
```

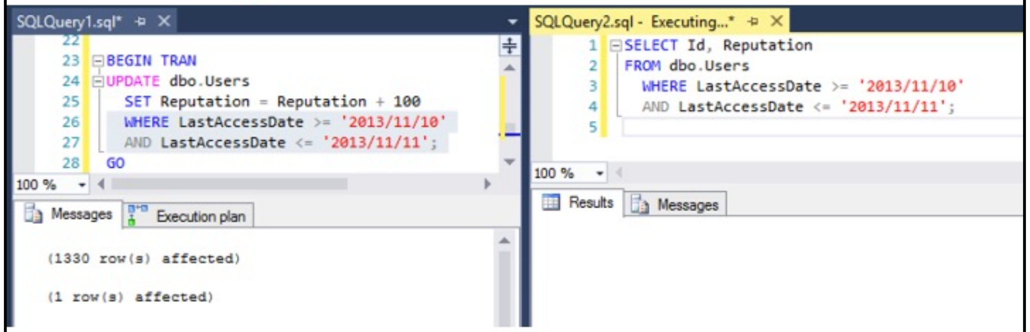
```
SELECT Id, Reputation
FROM dbo.Users
  WHERE LastAccessDate >= '1800/01/01'
  AND LastAccessDate <= '1800/01/01';
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Id],[Reputation] FROM [dbo].[Users] WHERE [LastAccess
```

Of course, no rows are returned.
But what happens if we look at 2013/11/10?

2.5 p24

Awww, shucks.



```
BEGIN TRAN
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11';
GO
```

```
(1330 row(s) affected)

(1 row(s) affected)
```

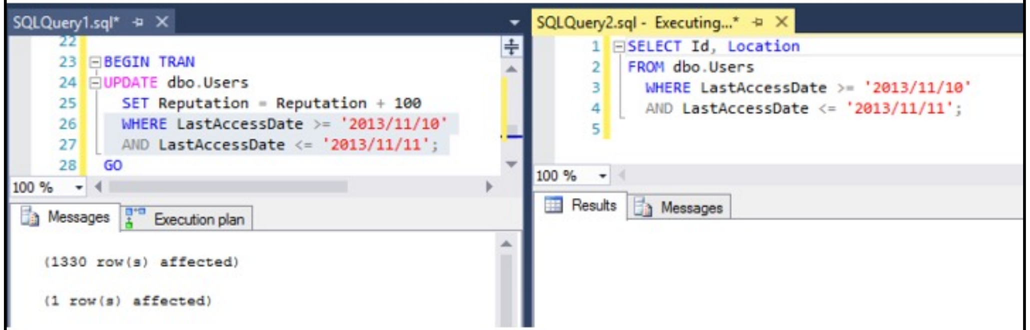
```
SELECT Id, Reputation
FROM dbo.Users
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11';
```

The SELECT on the right is blocked because we need Reputation, which is currently being edited.

But what if we skip Reputation, and get Location?

2.5 p25

It’s still blocked.



```sql
BEGIN TRAN
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11';
GO
```

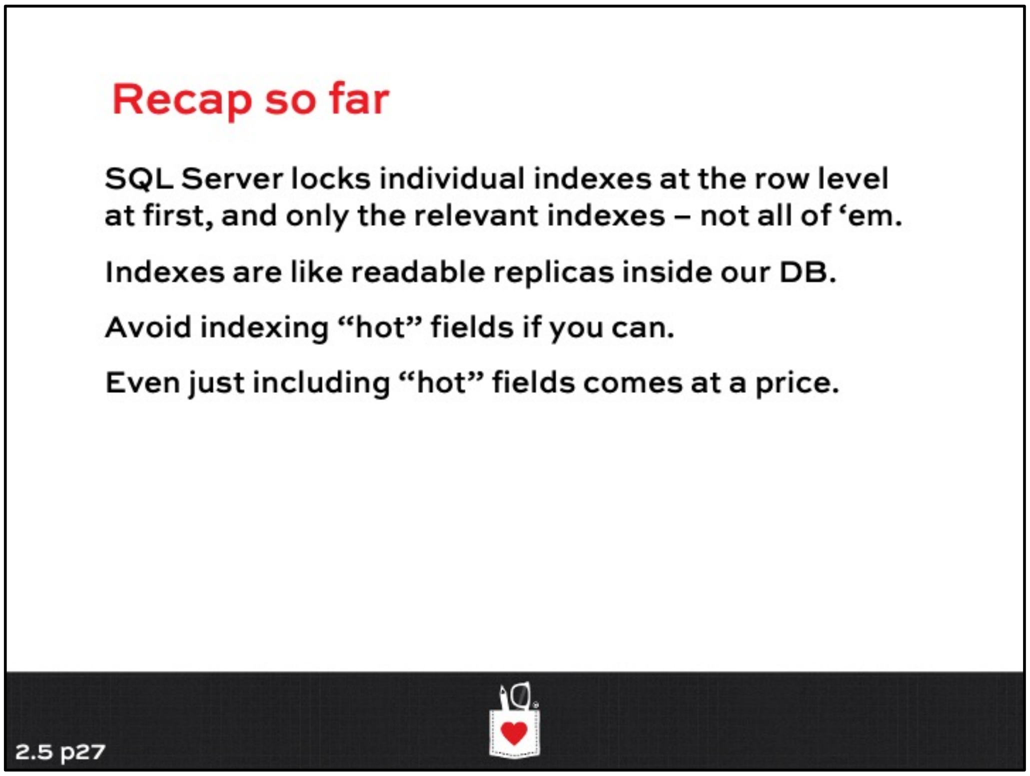
```
(1330 row(s) affected)

(1 row(s) affected)
```

```sql
SELECT Id, Location
FROM dbo.Users
  WHERE LastAccessDate >= '2013/11/10'
  AND LastAccessDate <= '2013/11/11';
```

The lock is on the clustered index row, not specific fields.

So if I wanted to query Id and Location, while I was updating Reputation, and not get blocked, what could I do?

2.5 p26



Recap so far

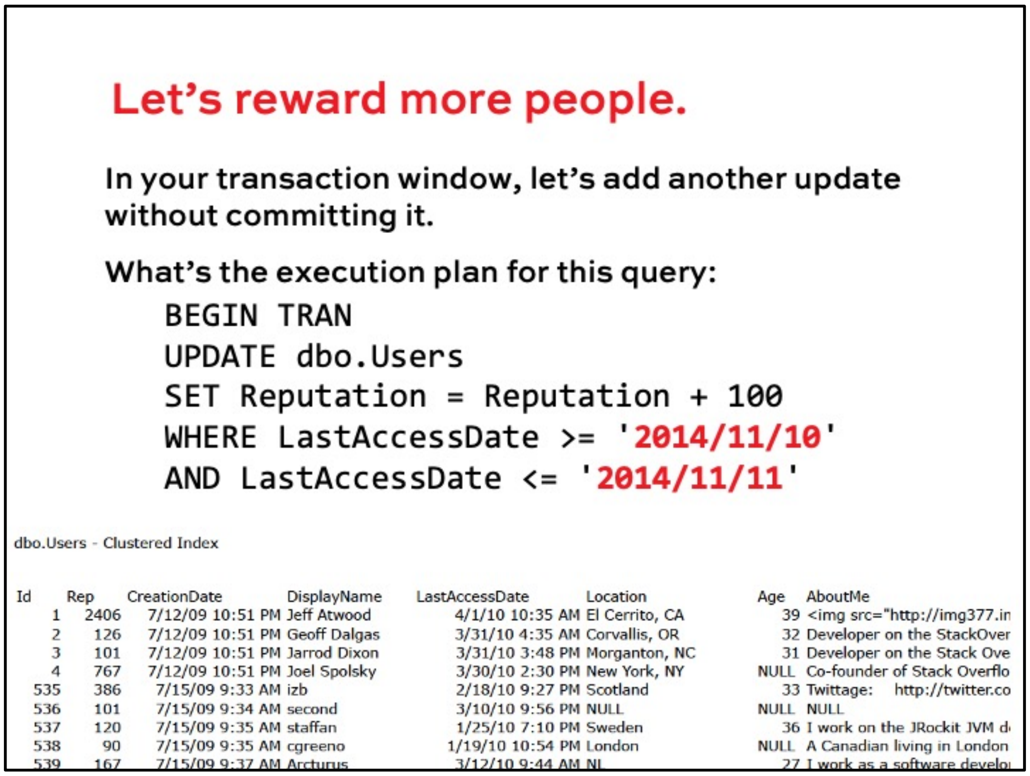
SQL Server locks individual indexes at the row level at first, and only the relevant indexes – not all of ‘em.

Indexes are like readable replicas inside our DB.

Avoid indexing “hot” fields if you can.

Even just including “hot” fields comes at a price.

2.5 p27



Let’s reward more people.

In your transaction window, let’s add another update without committing it.

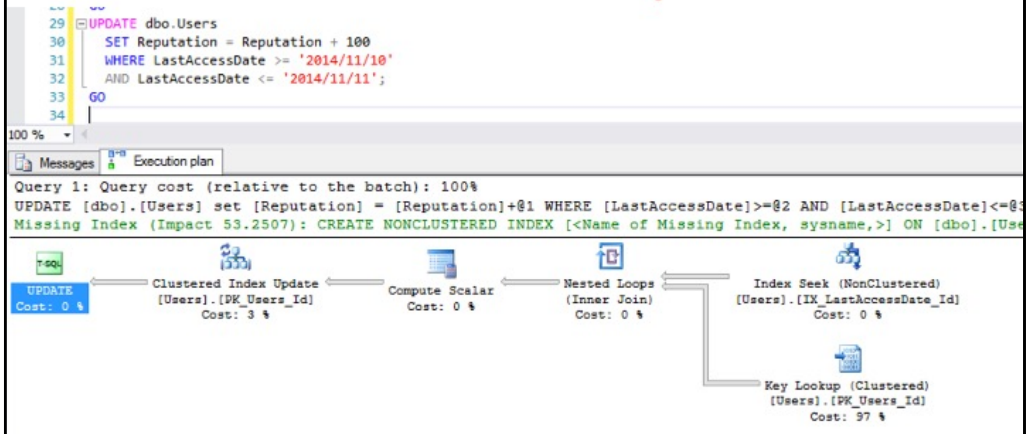
What’s the execution plan for this query:

```
BEGIN TRAN
UPDATE dbo.Users
SET Reputation = Reputation + 100
WHERE LastAccessDate >= '2014/11/10'
AND LastAccessDate <= '2014/11/11'
```

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo

Still the same execution plan.



```
UPDATE dbo.Users
   SET Reputation = Reputation + 100
   WHERE LastAccessDate >= '2014/11/10'
   AND LastAccessDate <= '2014/11/11';
GO
```

```
Query 1: Query cost (relative to the batch): 100%
UPDATE [dbo].[Users] set [Reputation] = [Reputation]+@1 WHERE [LastAccessDate]>=@2 AND [LastAccessDate]<=@3
Missing Index (Impact 53.2507): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Use
```

100 %
Messages
Execution plan
UPDATE Cost: 0 %
Clustered Index Update [Users].[PK_Users_Id] Cost: 3 %
Compute Scalar Cost: 0 %
Nested Loops (Inner Join) Cost: 0 %
Index Seek (NonClustered) [Users].[IX_LastAccessDate_Id] Cost: 0 %
Key Lookup (Clustered) [Users].[PK_Users_Id] Cost: 97 %

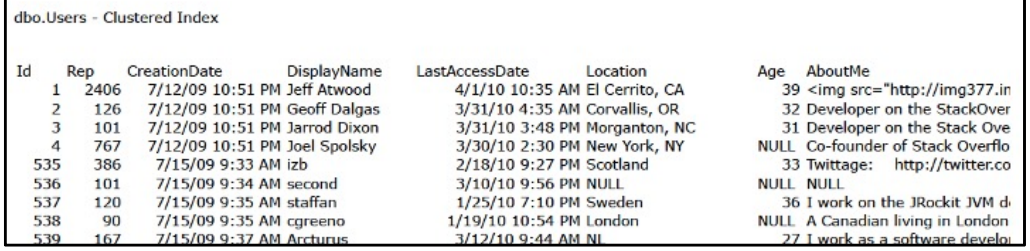
So far, so good.

2.5 p29

While that’s open, run another query

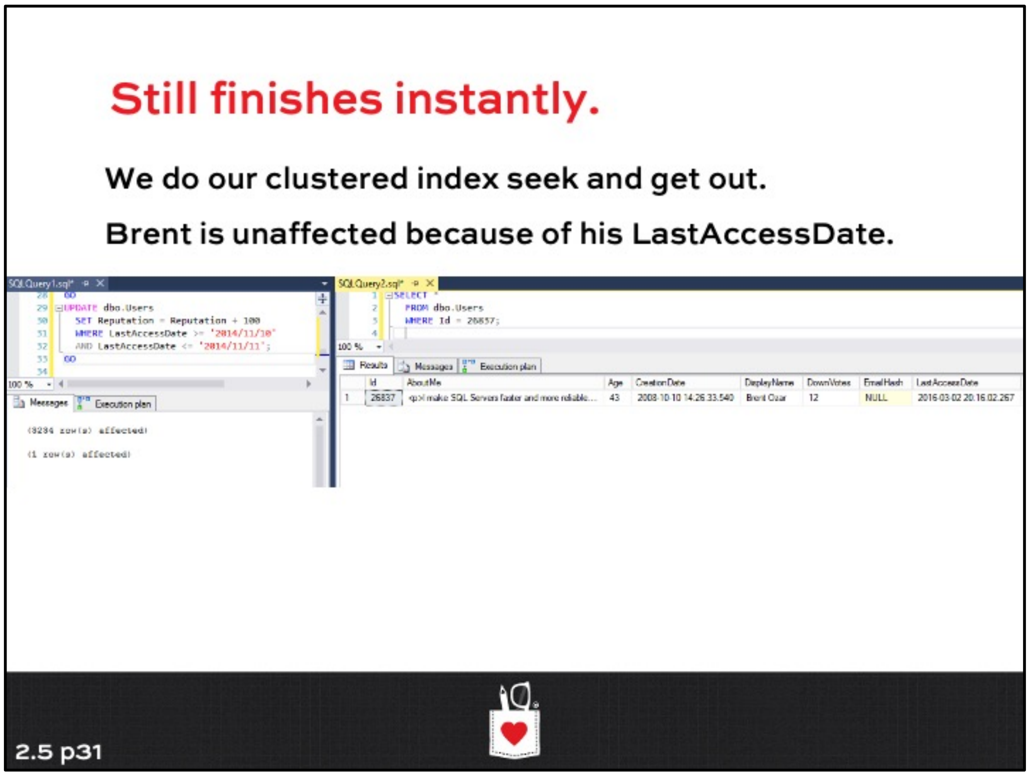
You’ve run this one before – but let’s try it again:

```
SELECT *
FROM dbo.Users
WHERE Id = 26837
```



dbo.Users - Clustered Index

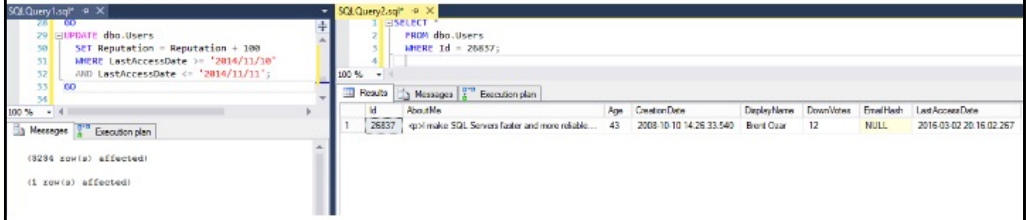
Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo



Still finishes instantly.

We do our clustered index seek and get out.

Brent is unaffected because of his LastAccessDate.



SQLQuery1.sql*

```
GO
UPDATE dbo.Users
   SET Reputation = Reputation + 100
   WHERE LastAccessDate >= '2014/11/10'
   AND LastAccessDate <= '2014/11/11';
GO
```

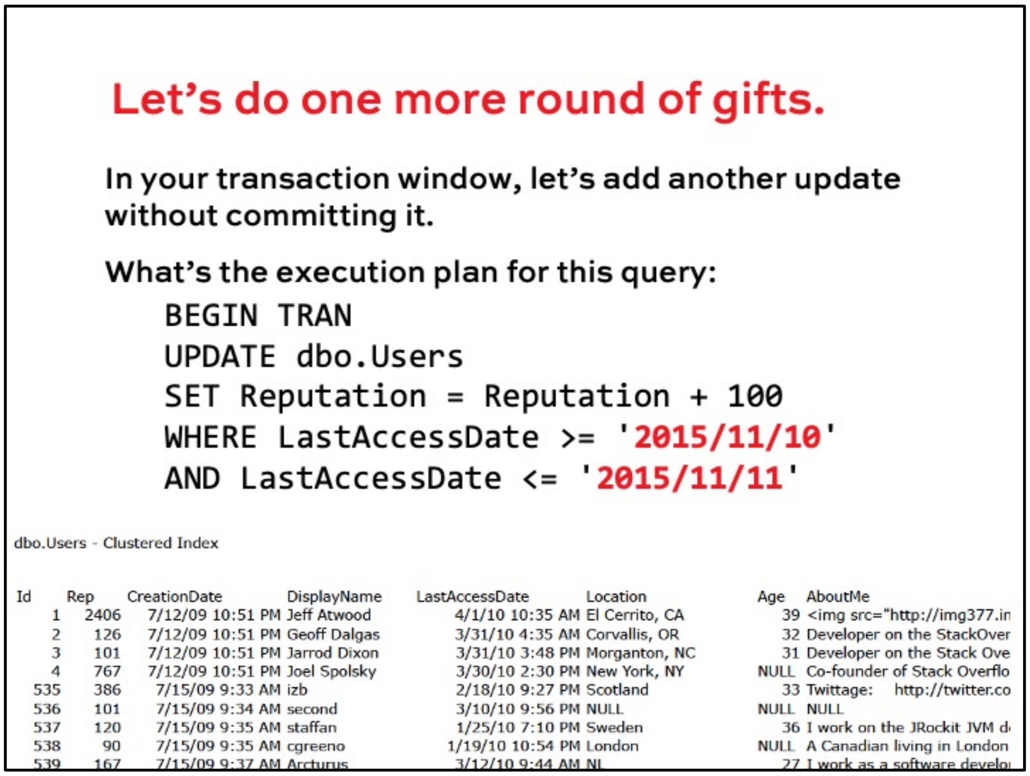
```
(8294 row(s) affected)

(1 row(s) affected)
```

SQLQuery2.sql*

```
SELECT *
  FROM dbo.Users
  WHERE Id = 26837;
```

	Id	AboutMe	Age	CreationDate	DisplayName	DownVotes	EmailHash	LastAccessDate
1	26837	<p>I make SQL Servers faster and more reliable...	43	2008-10-10 14:26:33.540	Brent Ozar	12	NULL	2016-03-02 20:16:02.267

2.5 p31



Let's do one more round of gifts.

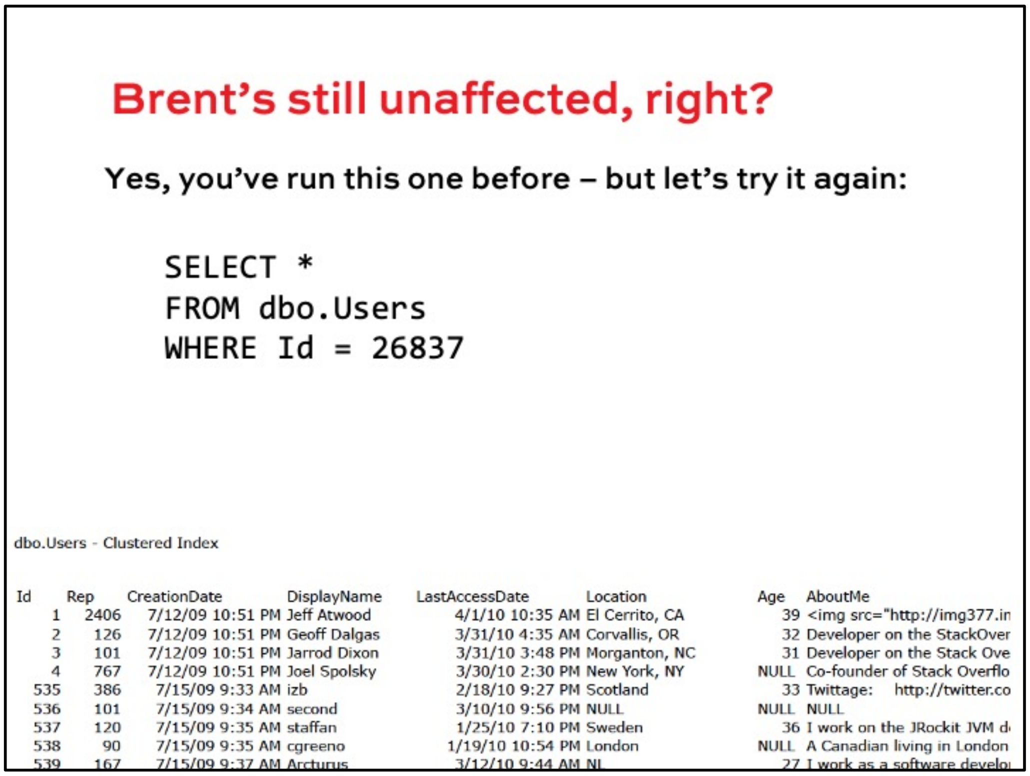
In your transaction window, let's add another update without committing it.

What's the execution plan for this query:

```
BEGIN TRAN
UPDATE dbo.Users
SET Reputation = Reputation + 100
WHERE LastAccessDate >= '2015/11/10'
AND LastAccessDate <= '2015/11/11'
```

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo



Brent's still unaffected, right?

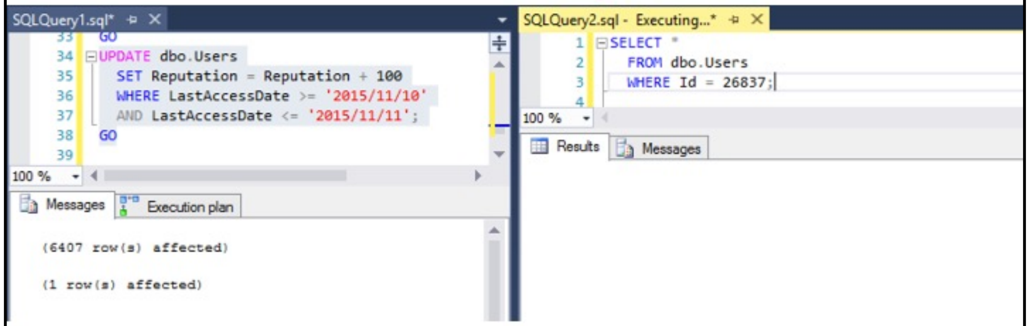
Yes, you've run this one before – but let's try it again:

```
SELECT *
FROM dbo.Users
WHERE Id = 26837
```

dbo.Users - Clustered Index

Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 3:48 PM	Morganton, NC	31	Developer on the Stack Ove
4	767	7/12/09 10:51 PM	Joel Spolsky	3/30/10 2:30 PM	New York, NY	NULL	Co-founder of Stack Overflo
535	386	7/15/09 9:33 AM	izb	2/18/10 9:27 PM	Scotland	33	Twittage: http://twitter.co
536	101	7/15/09 9:34 AM	second	3/10/10 9:56 PM	NULL	NULL	NULL
537	120	7/15/09 9:35 AM	staffan	1/25/10 7:10 PM	Sweden	36	I work on the JRockit JVM d
538	90	7/15/09 9:35 AM	cgreeno	1/19/10 10:54 PM	London	NULL	A Canadian living in London
539	167	7/15/09 9:37 AM	Arcturus	3/12/10 9:44 AM	NL	27	I work as a software develo

Wait a minute - literally



```
GO
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2015/11/10'
  AND LastAccessDate <= '2015/11/11';
GO
```

```
(6407 row(s) affected)

(1 row(s) affected)
```

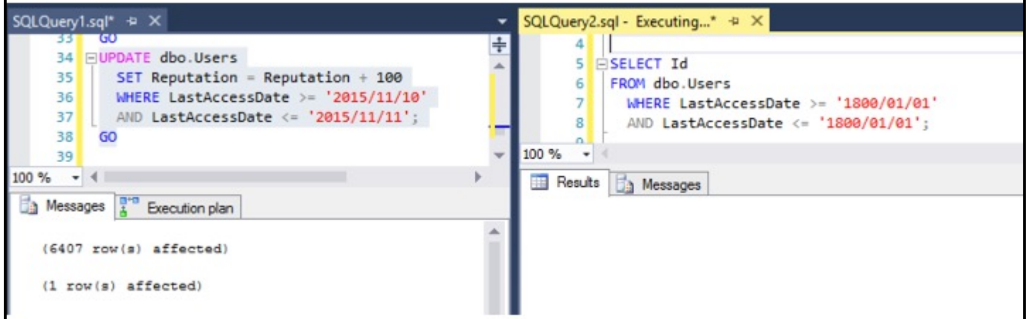
```
SELECT *
  FROM dbo.Users
  WHERE Id = 26837;
```

Brent shouldn't be blocked, but this SELECT hangs.

SQL Server has gone from locking individual rows of the clustered index, up to locking the entire index.

2.5 p34

Can we query by LastAccessDate?



```
GO
UPDATE dbo.Users
  SET Reputation = Reputation + 100
  WHERE LastAccessDate >= '2015/11/10'
  AND LastAccessDate <= '2015/11/11';
GO
```

```
(6407 row(s) affected)

(1 row(s) affected)
```

```
SELECT Id
FROM dbo.Users
  WHERE LastAccessDate >= '1800/01/01'
  AND LastAccessDate <= '1800/01/01';
```

Nope, that's blocked too now, even for 1800.

Our row-level locks got escalated to table locks.

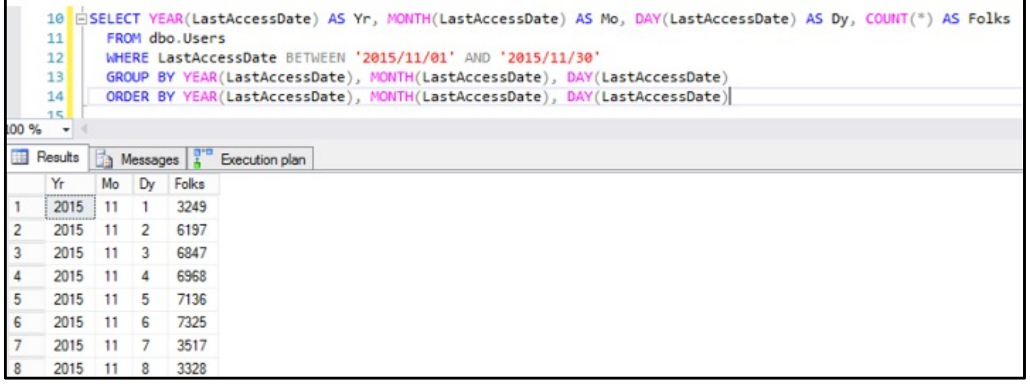
2.5 p35

That escalated quickly

SQL Server needs memory to track locks.

When queries hold thousands of row-level locks, SQL Server escalates those locks to table-level.

Depending on what day(s) of data you're updating, you might get row-level or table-level locks.

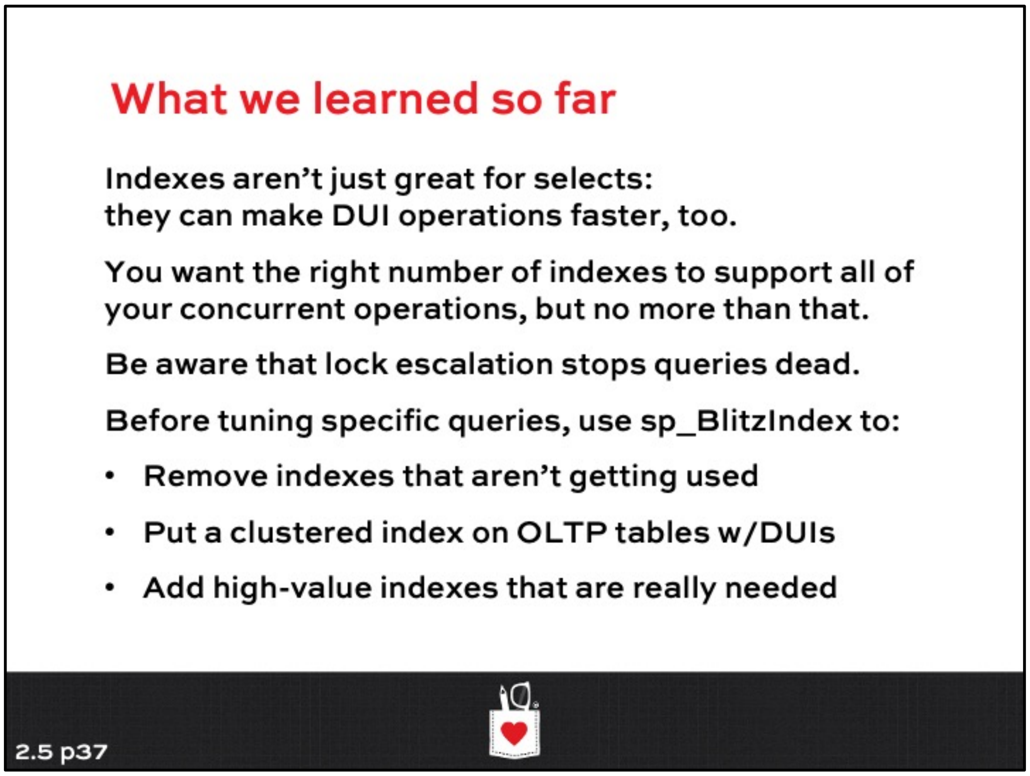


```sql
SELECT YEAR(LastAccessDate) AS Yr, MONTH(LastAccessDate) AS Mo, DAY(LastAccessDate) AS Dy, COUNT(*) AS Folks
  FROM dbo.Users
  WHERE LastAccessDate BETWEEN '2015/11/01' AND '2015/11/30'
  GROUP BY YEAR(LastAccessDate), MONTH(LastAccessDate), DAY(LastAccessDate)
  ORDER BY YEAR(LastAccessDate), MONTH(LastAccessDate), DAY(LastAccessDate)
```

100 %

Results | Messages | Execution plan

	Yr	Mo	Dy	Folks
1	2015	11	1	3249
2	2015	11	2	6197
3	2015	11	3	6847
4	2015	11	4	6968
5	2015	11	5	7136
6	2015	11	6	7325
7	2015	11	7	3517
8	2015	11	8	3328



What we learned so far

Indexes aren't just great for selects:
they can make DUI operations faster, too.

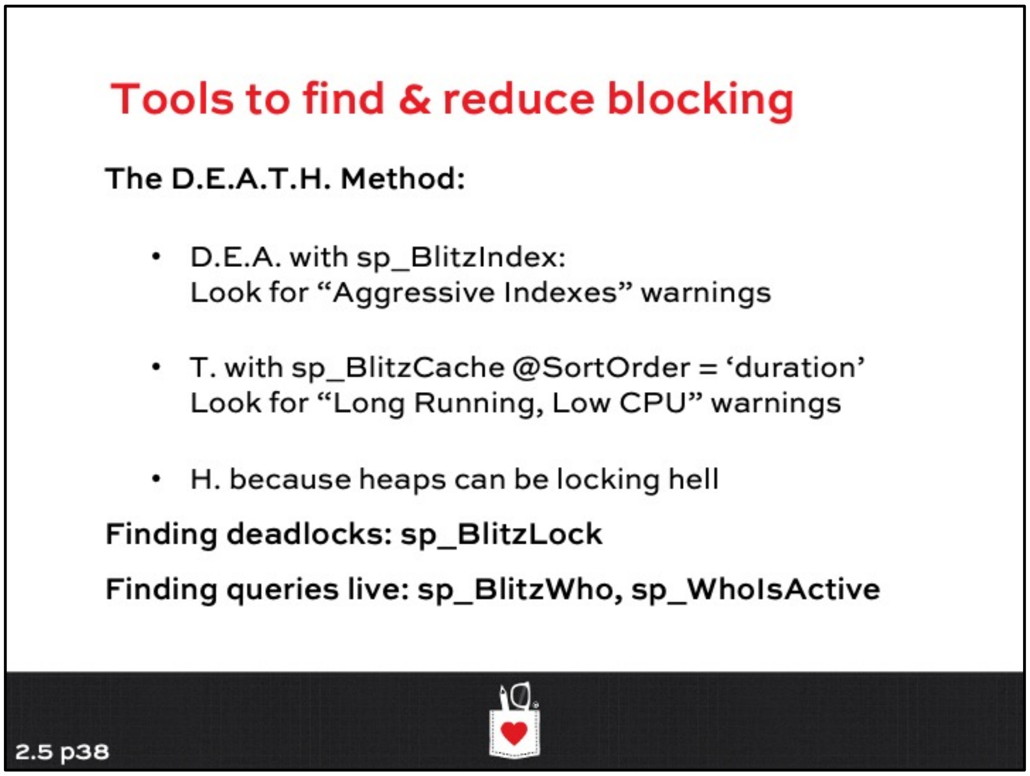
You want the right number of indexes to support all of your concurrent operations, but no more than that.

Be aware that lock escalation stops queries dead.

Before tuning specific queries, use sp_BlitzIndex to:

- Remove indexes that aren't getting used
- Put a clustered index on OLTP tables w/DUIs
- Add high-value indexes that are really needed

2.5 p37



Tools to find & reduce blocking

The D.E.A.T.H. Method:

- D.E.A. with sp_BlitzIndex:
  Look for "Aggressive Indexes" warnings
- T. with sp_BlitzCache @SortOrder = 'duration'
  Look for "Long Running, Low CPU" warnings
- H. because heaps can be locking hell

Finding deadlocks: sp_BlitzLock

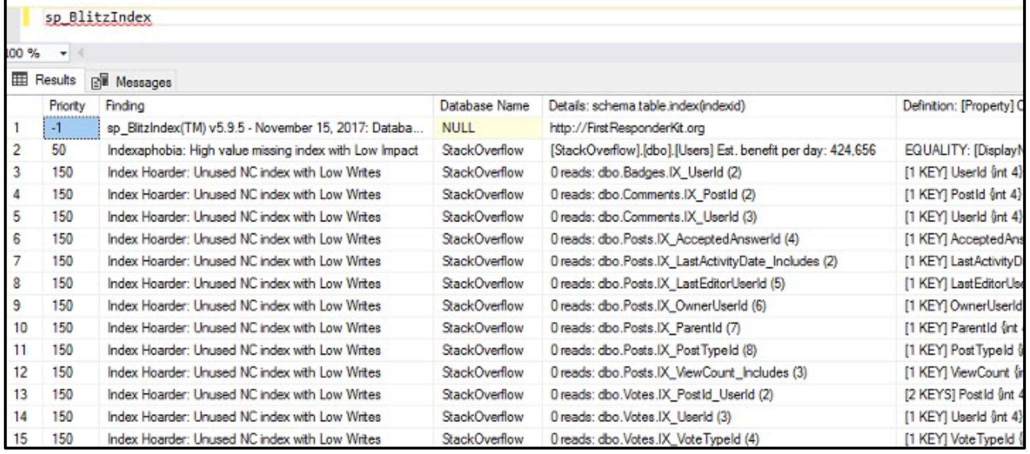
Finding queries live: sp_BlitzWho, sp_WhoIsActive

2.5 p38

About sp_BlitzIndex

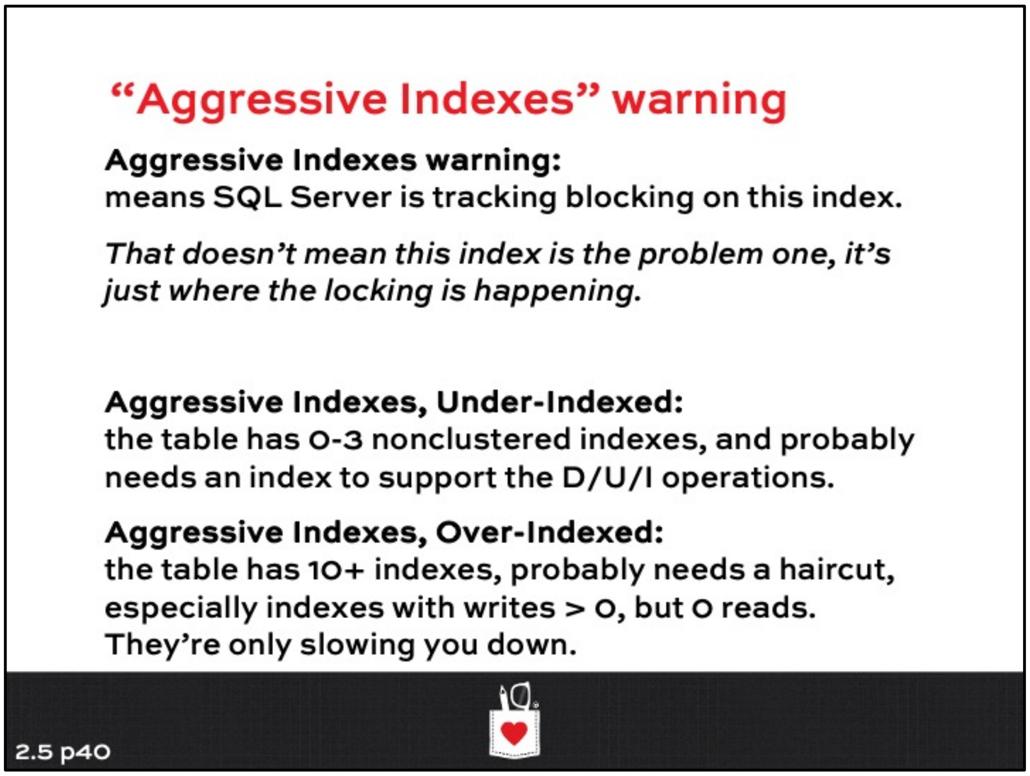
Totally free index health check from BrentOzar.com

Gives your indexes a psychiatric exam



```
sp_BlitzIndex
```

	Priority	Finding	Database Name	Details: schema.table.index(indexid)	Definition: [Property] C
1	-1	sp_BlitzIndex(TM) v5.9.5 - November 15, 2017: Databa...	NULL	http://FirstResponderKit.org	
2	50	Indexaphobia: High value missing index with Low Impact	StackOverflow	[StackOverflow].[dbo].[Users] Est. benefit per day: 424,656	EQUALITY: [DisplayN
3	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Badges.IX_UserId (2)	[1 KEY] UserId {int 4}
4	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Comments.IX_PostId (2)	[1 KEY] PostId {int 4}
5	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Comments.IX_UserId (3)	[1 KEY] UserId {int 4}
6	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Posts.IX_AcceptedAnswerId (4)	[1 KEY] AcceptedAns
7	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Posts.IX_LastActivityDate_Includes (2)	[1 KEY] LastActivityD
8	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Posts.IX_LastEditorUserId (5)	[1 KEY] LastEditorUse
9	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Posts.IX_OwnerUserId (6)	[1 KEY] OwnerUserId
10	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Posts.IX_ParentId (7)	[1 KEY] ParentId {int
11	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Posts.IX_PostTypeId (8)	[1 KEY] PostTypeId {
12	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Posts.IX_ViewCount_Includes (3)	[1 KEY] ViewCount {i
13	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Votes.IX_PostId_UserId (2)	[2 KEYS] PostId {int 4
14	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Votes.IX_UserId (3)	[1 KEY] UserId {int 4}
15	150	Index Hoarder: Unused NC index with Low Writes	StackOverflow	0 reads: dbo.Votes.IX_VoteTypeId (4)	[1 KEY] VoteTypeId {



“Aggressive Indexes” warning

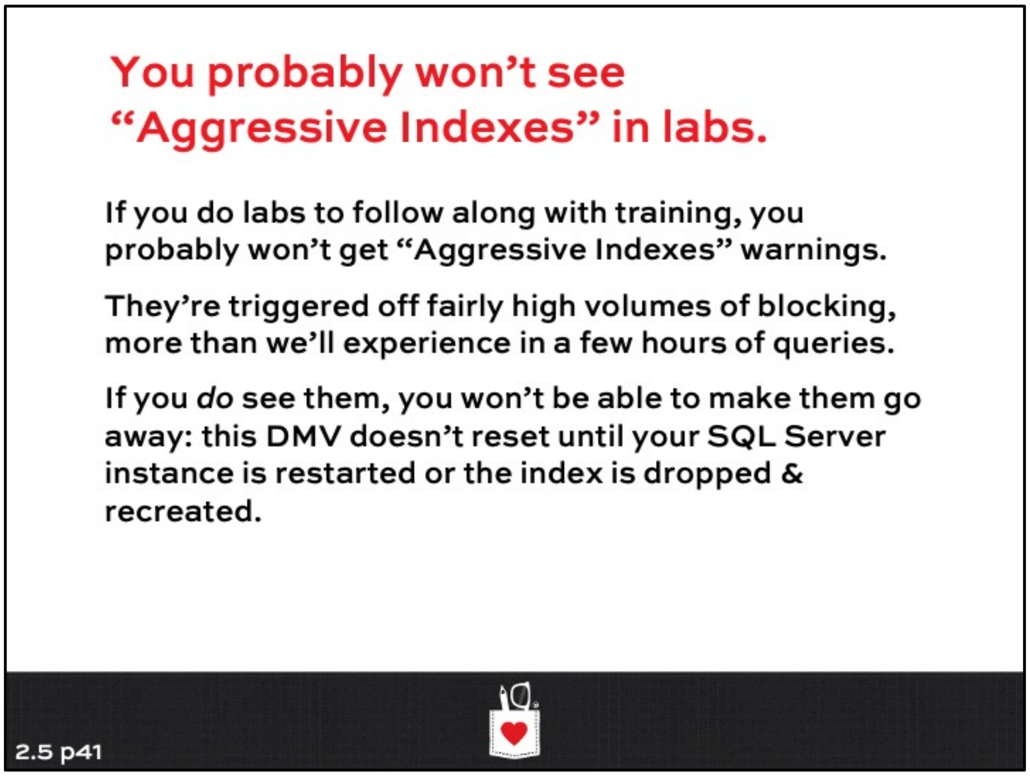
Aggressive Indexes warning:
means SQL Server is tracking blocking on this index.

That doesn’t mean this index is the problem one, it’s just where the locking is happening.

Aggressive Indexes, Under-Indexed:
the table has 0-3 nonclustered indexes, and probably needs an index to support the D/U/I operations.

Aggressive Indexes, Over-Indexed:
the table has 10+ indexes, probably needs a haircut, especially indexes with writes > 0, but 0 reads. They’re only slowing you down.

2.5 p40



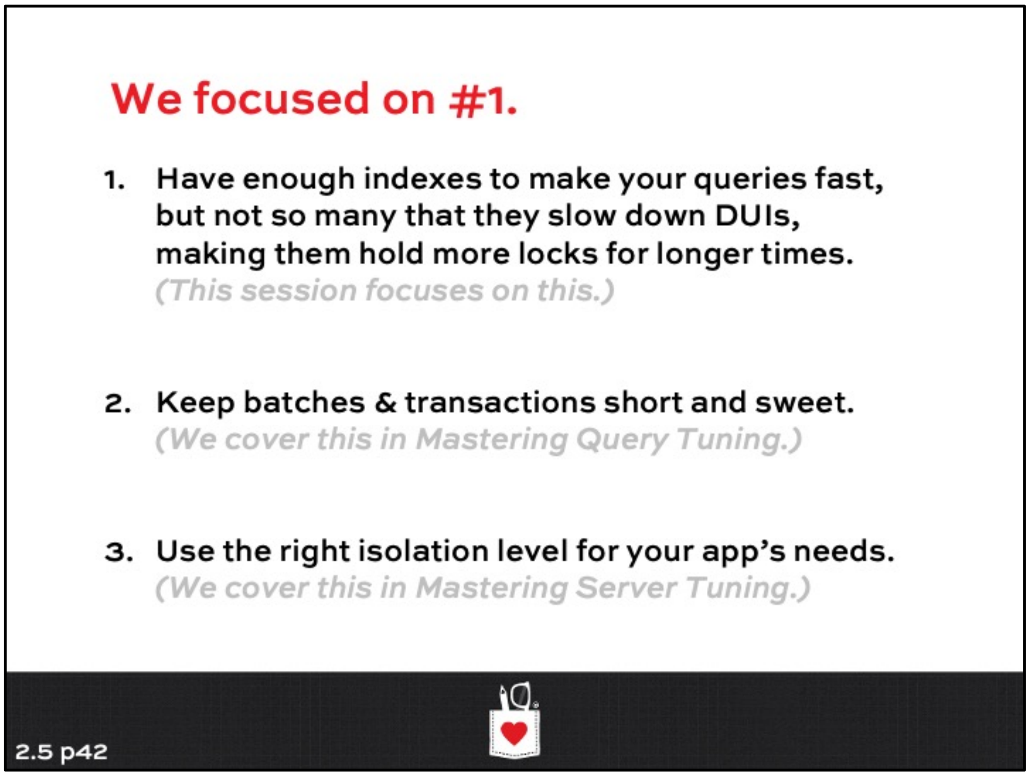
You probably won't see "Aggressive Indexes" in labs.

If you do labs to follow along with training, you probably won't get "Aggressive Indexes" warnings.

They're triggered off fairly high volumes of blocking, more than we'll experience in a few hours of queries.

If you *do* see them, you won't be able to make them go away: this DMV doesn't reset until your SQL Server instance is restarted or the index is dropped & recreated.

2.5 p41



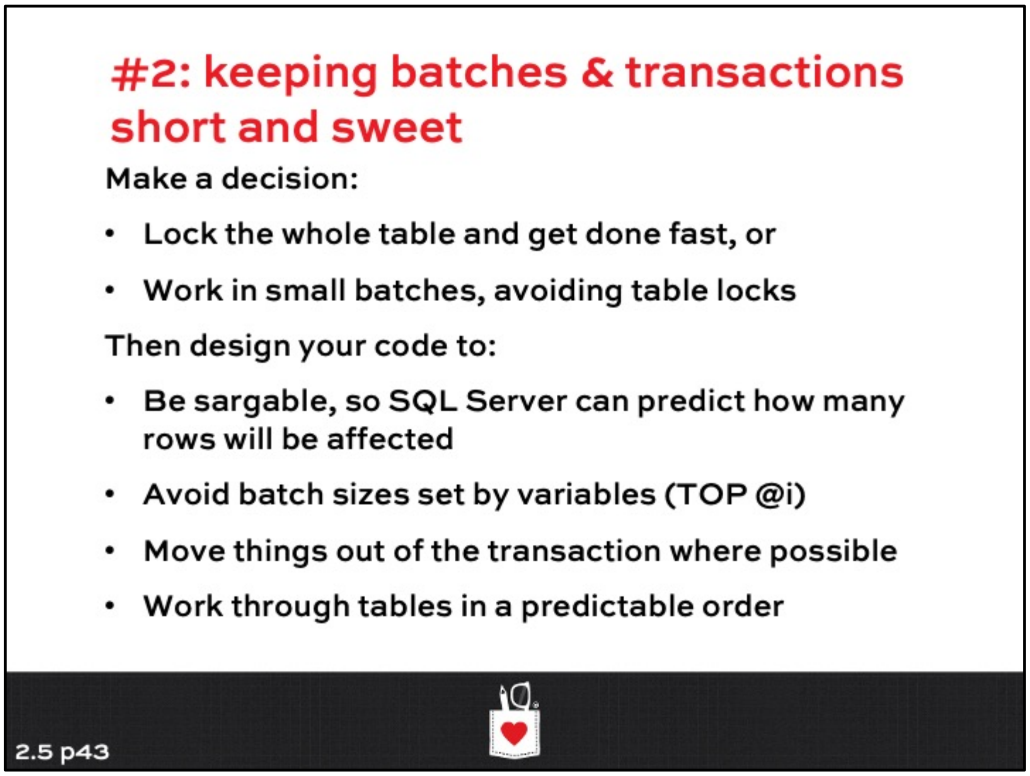
We focused on #1.

1. Have enough indexes to make your queries fast, but not so many that they slow down DUIs, making them hold more locks for longer times.
   (This session focuses on this.)

2. Keep batches & transactions short and sweet.
   (We cover this in Mastering Query Tuning.)

3. Use the right isolation level for your app's needs.
   (We cover this in Mastering Server Tuning.)

2.5 p42



#2: keeping batches & transactions short and sweet

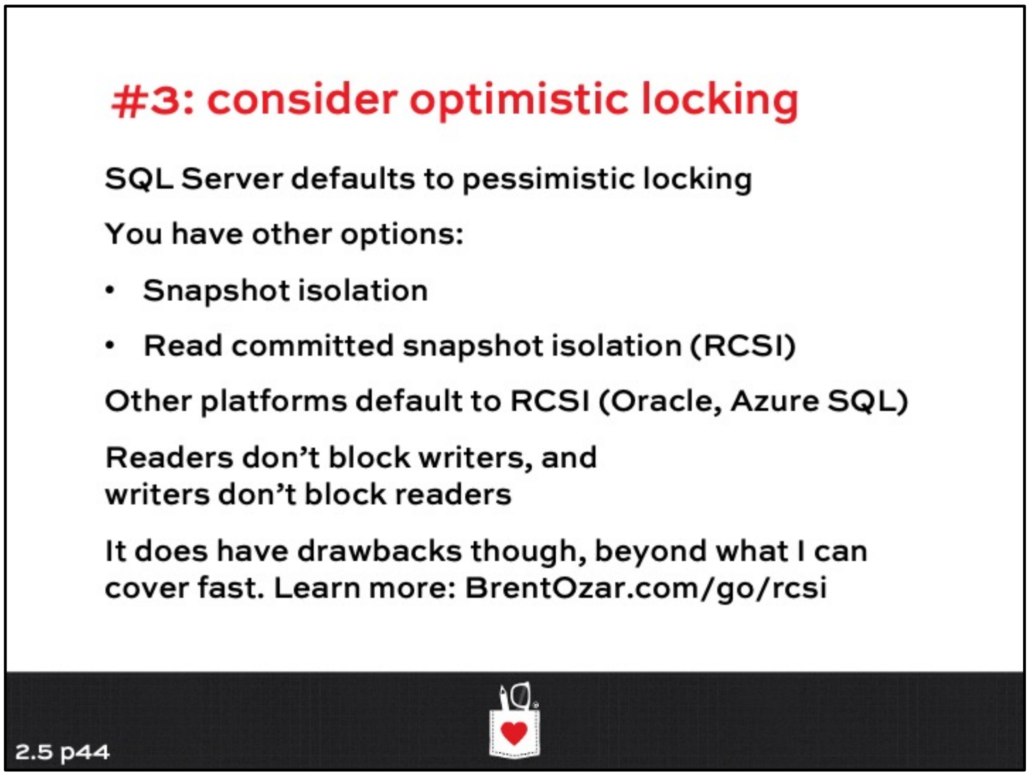
Make a decision:

- Lock the whole table and get done fast, or
- Work in small batches, avoiding table locks

Then design your code to:

- Be sargable, so SQL Server can predict how many rows will be affected
- Avoid batch sizes set by variables (TOP @i)
- Move things out of the transaction where possible
- Work through tables in a predictable order

2.5 p43



#3: consider optimistic locking

SQL Server defaults to pessimistic locking

You have other options:

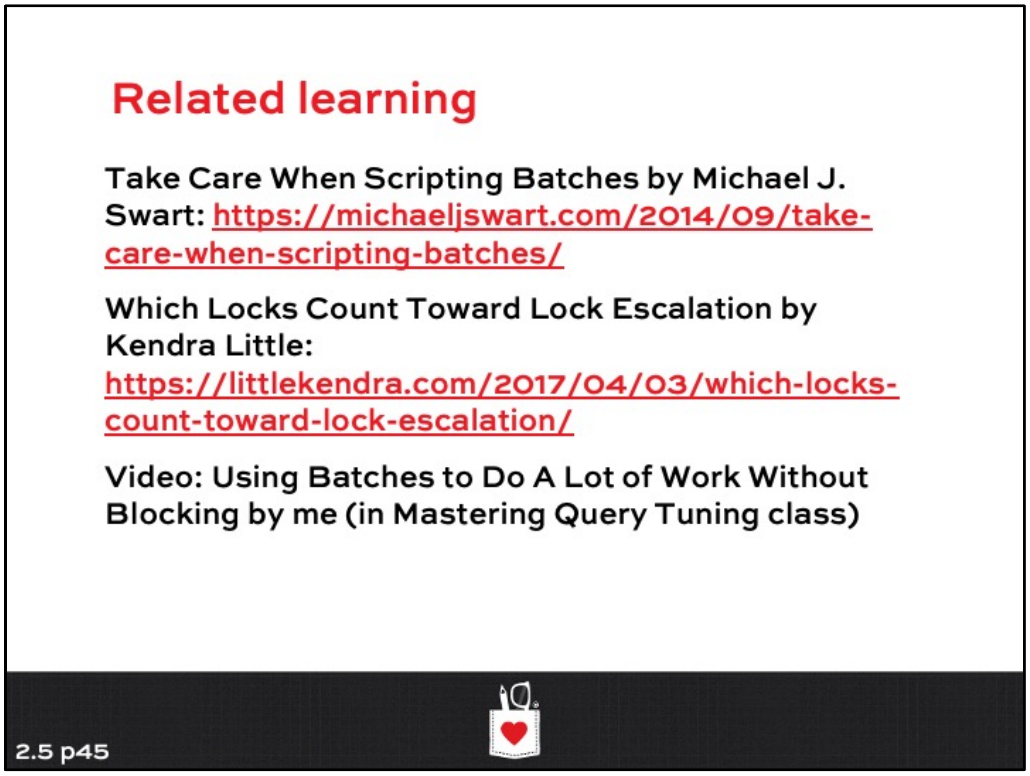
- Snapshot isolation
- Read committed snapshot isolation (RCSI)

Other platforms default to RCSI (Oracle, Azure SQL)

Readers don't block writers, and
writers don't block readers

It does have drawbacks though, beyond what I can
cover fast. Learn more: BrentOzar.com/go/rcsi

2.5 p44



Related learning

Take Care When Scripting Batches by Michael J. Swart: https://michaeljswart.com/2014/09/take-care-when-scripting-batches/

Which Locks Count Toward Lock Escalation by Kendra Little: https://littlekendra.com/2017/04/03/which-locks-count-toward-lock-escalation/

Video: Using Batches to Do A Lot of Work Without Blocking by me (in Mastering Query Tuning class)

2.5 p45