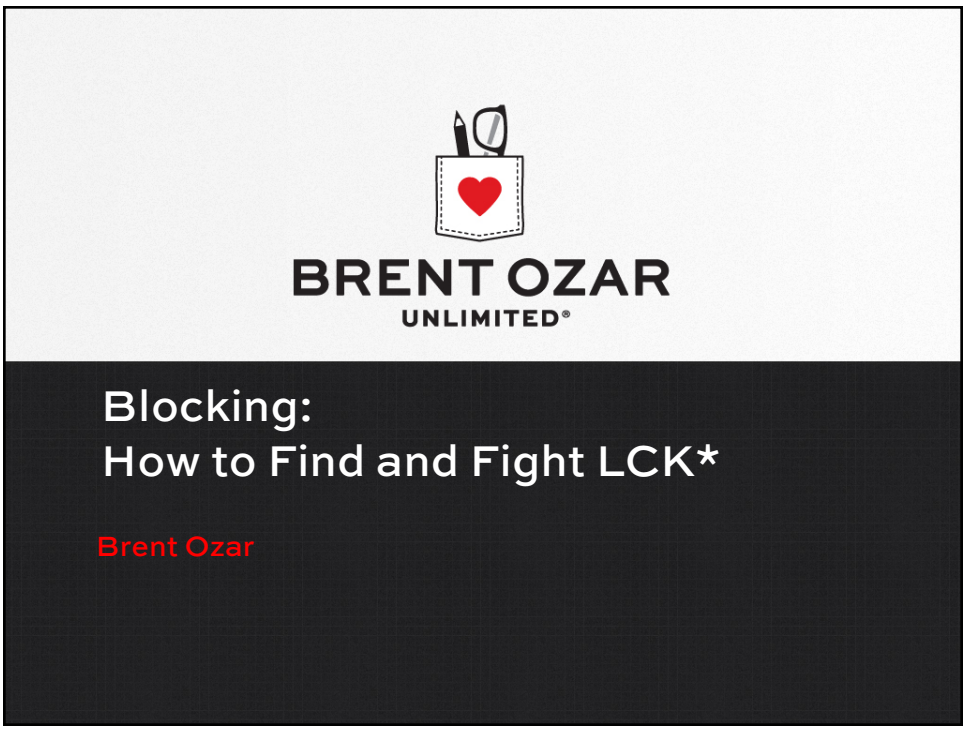


BrentOzar.com/go/lock

LCK* waits mean locks,

but specifically that queries are waiting to get locks because someone else has them.

It means concurrency.

Concurrency challenges

Locking: Larry takes out a lock.

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

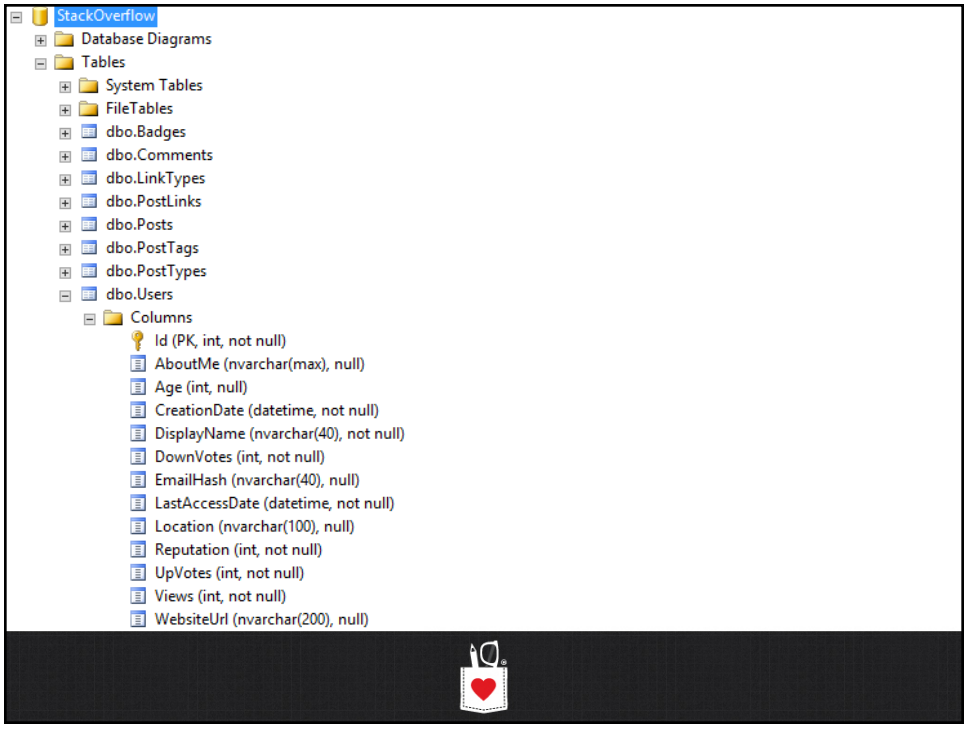
Deadlocks:
Larry has locks, but needs some held by Sarah.
Sarah has locks, but needs some held by Larry.
SQL Server solves this one by killing somebody,
and the symptom is dead bodies everywhere.

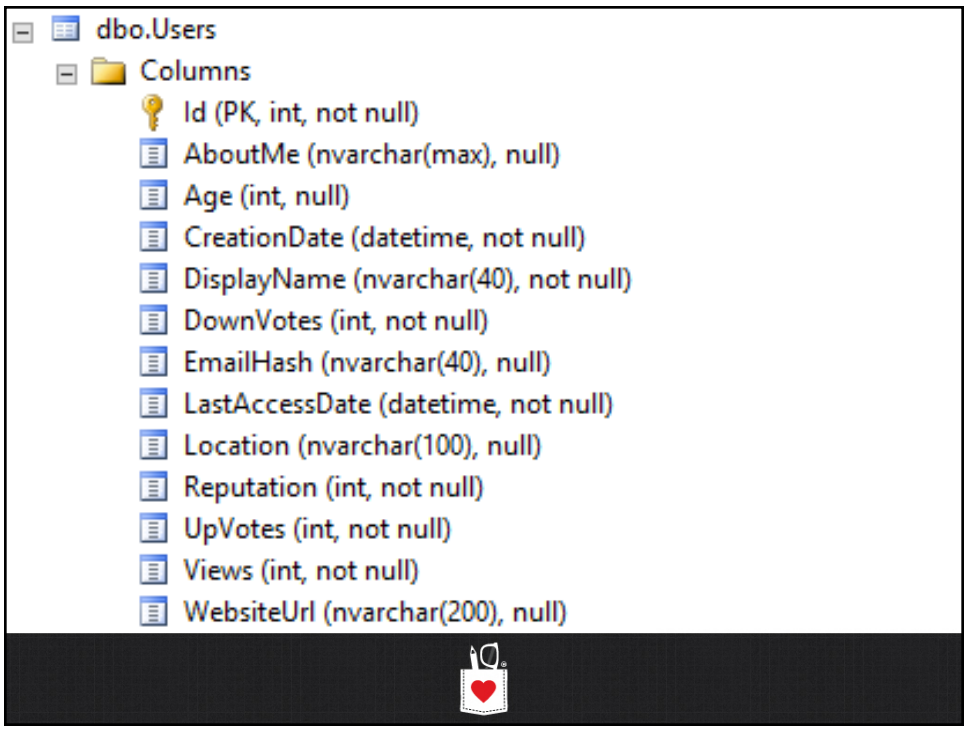
2 ways to fix blocking & deadlocks

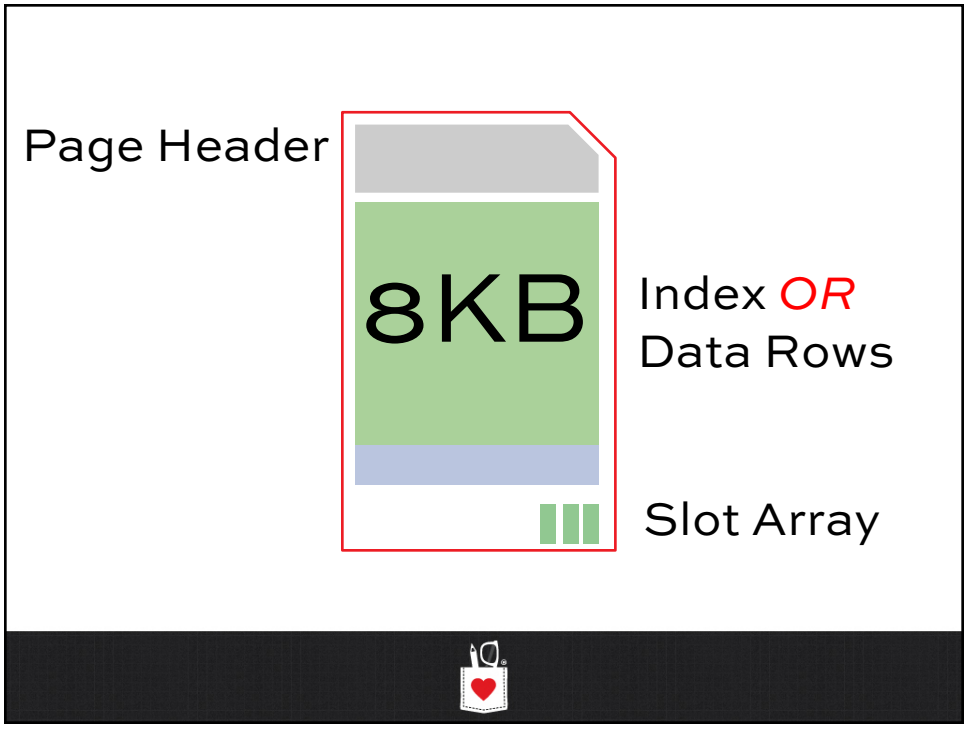
1. Have enough indexes to make your queries fast, but not so many that they slow down DUIs, making them hold more locks for longer times.

2. Use the right isolation level for your app's needs.

Right-sizing indexes







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
540	101	7/15/09 9:37 AM	DanSingerman	12/1/09 4:37 PM	NULL	NULL	<p>Sufficiently advanced st
541	647	7/15/09 9:37 AM	Alexis Hirst	4/1/10 6:37 AM	England	21	NULL
542	5921	7/15/09 9:38 AM	hyperslug	4/1/10 10:49 PM	NULL	NULL	<img src="http://www.hyp
543	490	7/15/09 9:38 AM	DavidWhitney	12/12/09 5:12 PM	London, UK	25	Primarily C# .NET develope
544	101	7/15/09 9:39 AM	Ben Laan	3/28/10 11:12 PM	Adelaide, Australia	34	C#, Delphi, T-SQL : Archite
545	985	7/15/09 9:40 AM	redacted	4/1/10 3:34 PM	NaN	23	NaN
546	101	7/15/09 9:41 AM	hitec	8/26/09 9:59 AM	Bombay	27	System.StackOverflowExcep
547	404	7/15/09 9:41 AM	Ant	4/1/10 8:24 AM	England	27	C# developer since 2005.
548	501	7/15/09 9:41 AM	Jeremy French	1/28/10 5:18 PM	UK	33	Seasoned web developer. W
549	173	7/15/09 9:42 AM	Joshua	2/26/10 5:53 PM	Orlando, FL	28	Software Engineer. Have wif
550	61	7/15/09 9:42 AM	hopsasa	3/16/10 4:42 PM	NULL	NULL	NULL
551	341	7/15/09 9:43 AM	Tom	4/1/10 10:39 AM	Worcester, UK	NULL	NULL
552	1	7/15/09 9:44 AM	Exodus	11/29/09 7:20 AM	Maracaibo, Venezuela	25	NULL
553	101	7/15/09 9:44 AM	moff	2/7/10 10:28 AM	Sweden	NULL	NULL
554	1158	7/15/09 9:44 AM	becomingGuru	4/1/10 8:43 PM	Bangalore / Hyderabad	25	curious learner!
555	1016	7/15/09 9:45 AM	MiffTheFox	4/1/10 6:11 PM	Huntsville, AL	NULL	<p><a href="http://forums
556	101	7/15/09 9:45 AM	Russ Cam	11/8/09 6:10 PM	UK	NULL	Interested in C#, ASP.NET,
557	1	7/15/09 9:45 AM	Beto	2/13/10 11:01 AM	NULL	NULL	NULL
558	953	7/15/09 9:46 AM	marco.ragogna	4/1/10 7:41 AM	IT	29	<h2>Software Developer</
559	593	7/15/09 9:46 AM	moobaa	2/18/10 7:14 AM	AU	39	Hey, at least I know 6502!
560	83	7/15/09 9:46 AM	rjstelling	3/4/10 6:47 PM	United Kingdom	30	<h3>Tech geek and hopeles
561	716	7/15/09 9:47 AM	balpha	4/1/10 4:00 PM	NULL	NULL	NULL
562	43	7/15/09 9:47 AM	Mike McClelland	3/16/10 8:45 AM	London	NULL	NULL
563	101	7/15/09 9:48 AM	arturh	10/29/09 2:11 PM	Bremen, Germany	24	NULL
564	960	7/15/09 9:49 AM	Sliff	4/1/10 6:59 AM	UK	30	<p>I'm a Software Enginee
565	121	7/15/09 9:49 AM	Franci Penov	3/5/10 12:13 AM	Kirkland, WA	37	<p>Father, SDE@Microsoft

From http://StackOverflow.com's Creative Commons Data Dump

You only have the clustered index.

How many accessed the system on my birthday?

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

100 %

Results | Messages

	(No column name)
1	1330

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

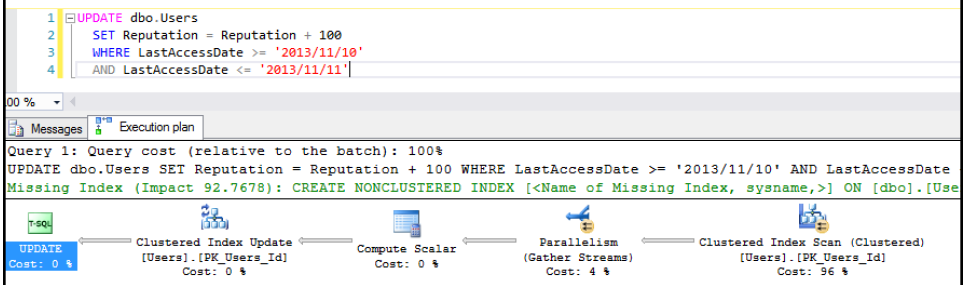
Your execution plan:

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

SQL Server's execution plan



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

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	NI	27	I work as a software develo

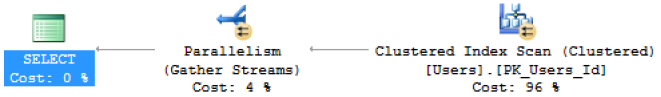
SQL Server's execution plan

```
1  SELECT Id
2  FROM dbo.Users
3  WHERE LastAccessDate >= '1800/01/01'
4  AND LastAccessDate <= '1800/01/02'
5
```

100 %

Messages | Execution plan

```
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?

Our SELECT just sits there blocked.

SQLQuery1.sql*

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

100 %

Messages

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

SQLQuery2.sql - Executing...*

```
1  SELECT Id
2  FROM dbo.Users
3  WHERE LastAccessDate >= '1800/01/01'
4  AND LastAccessDate <= '1800/01/02'
5
```

100 %

Results | Messages

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.

Let's try another query.

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

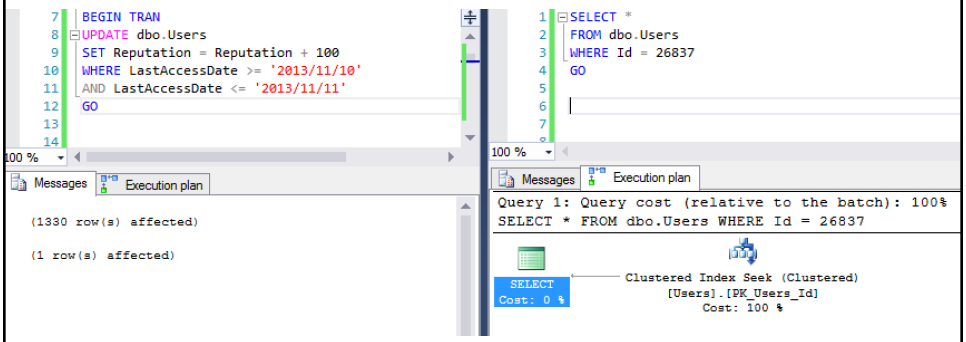
What's the execution plan for this query:

```
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?



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?

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

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

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
7/15/09 7:13 AM	65	7/15/09 10:21 AM	618	7/20/09 1:06 PM	849	8/18/09 8:06 AM	511
7/15/09 7:14 AM	68	7/15/09 10:25 AM	347	7/21/09 7:22 AM	881	8/18/09 9:00 AM	262
7/15/09 7:15 AM	73	7/15/09 10:26 AM	623	7/23/09 11:53 AM	503	8/18/09 9:43 AM	210
7/15/09 7:17 AM	87	7/15/09 10:28 AM	629	7/23/09 12:56 PM	446	8/18/09 10:22 AM	673
7/15/09 7:18 AM	92	7/15/09 10:32 AM	638	7/24/09 12:15 AM	407	8/18/09 1:05 PM	959
7/15/09 7:20 AM	110	7/15/09 10:36 AM	642	7/24/09 12:08 PM	488	8/18/09 1:14 PM	643
7/15/09 7:26 AM	139	7/15/09 10:46 AM	661	7/24/09 3:17 PM	144	8/18/09 1:32 PM	525
7/15/09 7:36 AM	173	7/15/09 10:59 AM	686	7/26/09 8:48 AM	615	8/18/09 1:38 PM	159
7/15/09 7:38 AM	179	7/15/09 11:14 AM	722	7/27/09 2:27 PM	337	8/18/09 2:28 PM	690
7/15/09 7:46 AM	223	7/15/09 11:26 AM	756	7/28/09 3:17 AM	174	8/19/09 9:22 AM	750
7/15/09 7:57 AM	258	7/15/09 11:48 AM	803	7/28/09 12:34 PM	118	8/19/09 2:46 PM	887
7/15/09 7:58 AM	259	7/15/09 11:50 AM	681	7/28/09 9:28 PM	201	8/20/09 3:45 AM	749
7/15/09 8:03 AM	279	7/15/09 11:51 AM	808	7/28/09 10:53 PM	656	8/20/09 8:29 AM	268
7/15/09 8:06 AM	286	7/15/09 12:00 PM	822	7/29/09 12:06 PM	751	8/20/09 10:32 AM	901
7/15/09 8:12 AM	305	7/15/09 12:07 PM	841	7/30/09 12:26 PM	754	8/20/09 12:05 PM	752
7/15/09 8:13 AM	313	7/15/09 12:07 PM	842	7/31/09 6:36 PM	11	8/20/09 3:03 PM	272
7/15/09 8:14 AM	320	7/15/09 12:18 PM	878	8/1/09 12:09 PM	775	8/21/09 7:29 PM	487
7/15/09 8:16 AM	323	7/15/09 12:18 PM	779	8/1/09 7:29 PM	670	8/22/09 11:58 AM	580
7/15/09 8:25 AM	349	7/15/09 12:47 PM	973	8/1/09 11:33 PM	85	8/24/09 11:22 AM	56
7/15/09 8:26 AM	354	7/15/09 12:48 PM	819	8/3/09 10:13 PM	300	8/24/09 11:33 PM	889
7/15/09 8:27 AM	97	7/15/09 12:52 PM	987	8/5/09 12:06 PM	918	8/25/09 8:57 PM	150
7/15/09 8:27 AM	363	7/15/09 12:54 PM	992	8/6/09 5:46 AM	726	8/26/09 2:43 AM	851
7/15/09 8:33 AM	384	7/15/09 1:28 PM	931	8/6/09 10:42 AM	370	8/26/09 2:53 AM	117
7/15/09 8:37 AM	402	7/15/09 2:19 PM	728	8/6/09 12:37 PM	398	8/26/09 9:59 AM	546
7/15/09 8:47 AM	431	7/15/09 2:20 PM	217	8/9/09 5:28 AM	45	8/27/09 3:54 AM	239
7/15/09 8:52 AM	440	7/15/09 4:23 PM	975	8/10/09 12:25 AM	389	8/27/09 2:10 PM	271

From http://StackOverflow.com's Creative Commons Data Dump

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

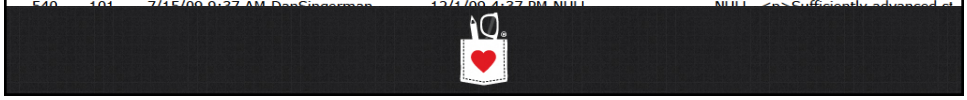
Id	Rep	CreationDate	DisplayName	LastAccessDate	Location	Age	AboutMe
1	2406	7/12/09 10:51 PM	Jeff Atwood	4/1/10 10:35 AM	El Cerrito, CA	39	<img src="http://img377.in
2	126	7/12/09 10:51 PM	Geoff Dalgas	3/31/10 4:35 AM	Corvallis, OR	32	Developer on the StackOver
3	101	7/12/09 10:51 PM	Jarrod Dixon	3/31/10 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	NI	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



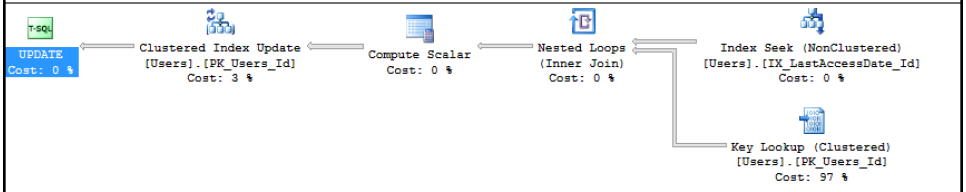
Here's the plan.

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

00 %

Messages | Execution plan

```
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:

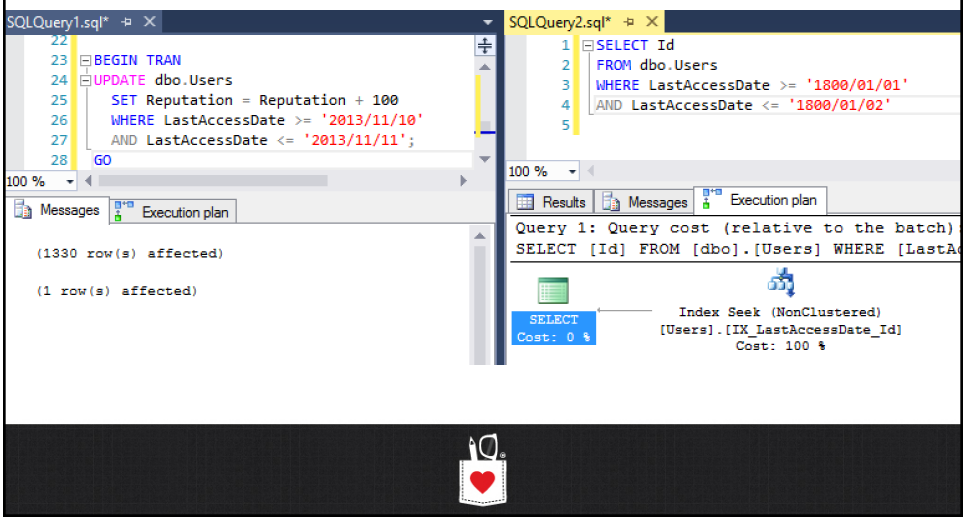
```
SELECT Id
FROM dbo.Users
WHERE LastAccessDate >= '1800/01/01'
AND LastAccessDate <= '1800/01/02'
```

dbo.Users - IX_LastAccessDate

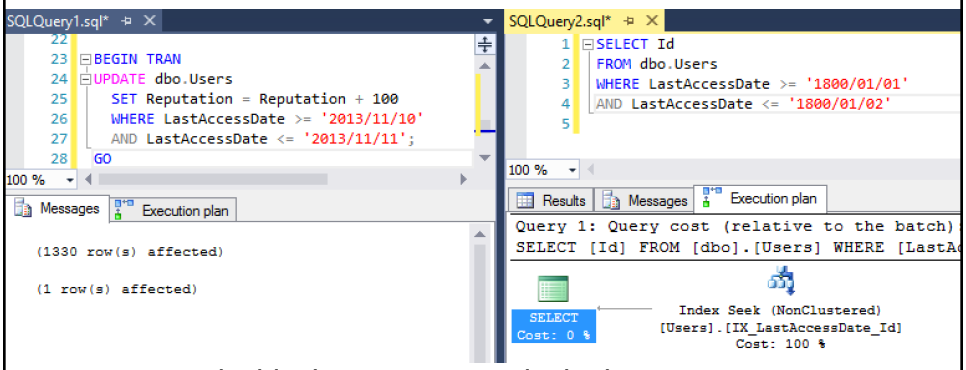
LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id
7/31/08 12:00 AM	-1	7/15/09 8:53 AM	445	7/15/09 9:10 PM	200	8/11/09 7:17 PM	39
7/15/09 7:08 AM	22	7/15/09 8:58 AM	457	7/16/09 6:22 AM	678	8/12/09 2:54 PM	943
7/15/09 7:10 AM	33	7/15/09 9:17 AM	501	7/17/09 2:30 AM	131	8/13/09 4:26 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.



The SELECT finishes immediately.



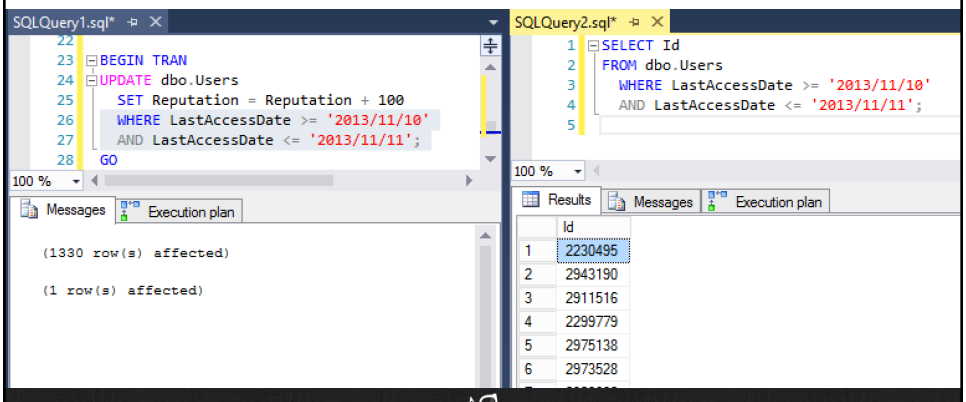
Presto! The black pages weren't locked.

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

Still no blocking.

Our SELECT finishes instantly.

Indexes are amazing and the cure to all ills.



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:

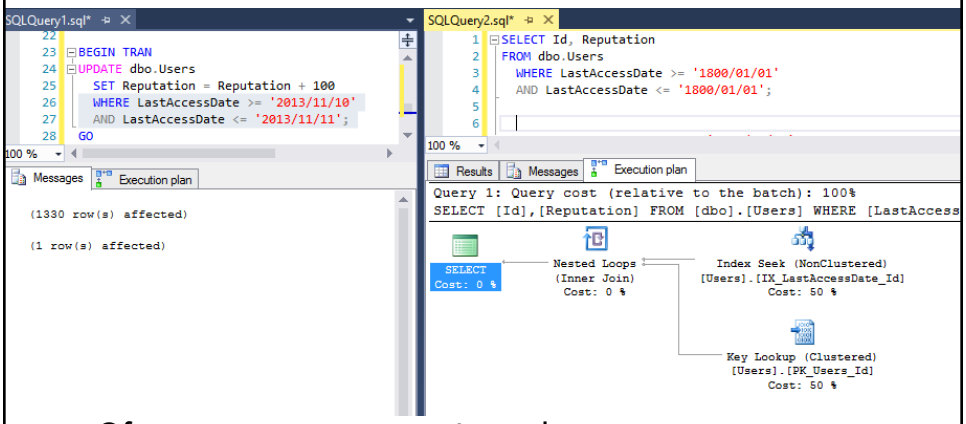
1. Use the new index on LastAccessDate – seek directly to 1800/01/01, make a list of rows that match. (There won't be any, right?)
2. Look up their IDs in the clustered index (white pages), and get their Reputation field

Will it ~~blend~~ be blocked?

dbo.Users - IX_LastAccessDate

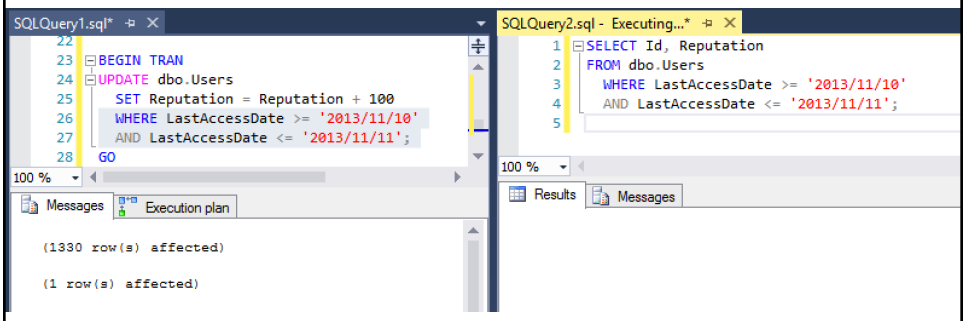
LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id	LastAccessDate	Id
7/31/08 12:00 AM	-1	7/15/09 8:53 AM	445	7/15/09 9:10 PM	200	8/11/09 7:17 PM	39
7/15/09 7:08 AM	22	7/15/09 8:58 AM	457	7/16/09 6:22 AM	678	8/12/09 2:54 PM	943
7/15/09 7:10 AM	33	7/15/09 9:17 AM	501	7/17/09 2:30 AM	131	8/13/09 4:26 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!



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

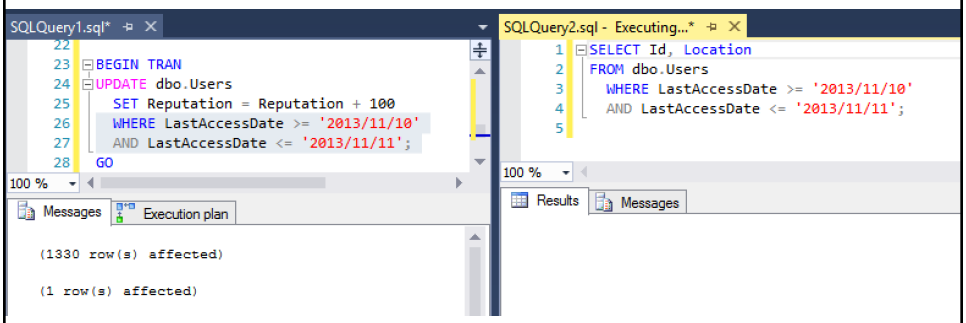
Awww, shucks.



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?

It's still blocked.



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?

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.

Let's reward more people.

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

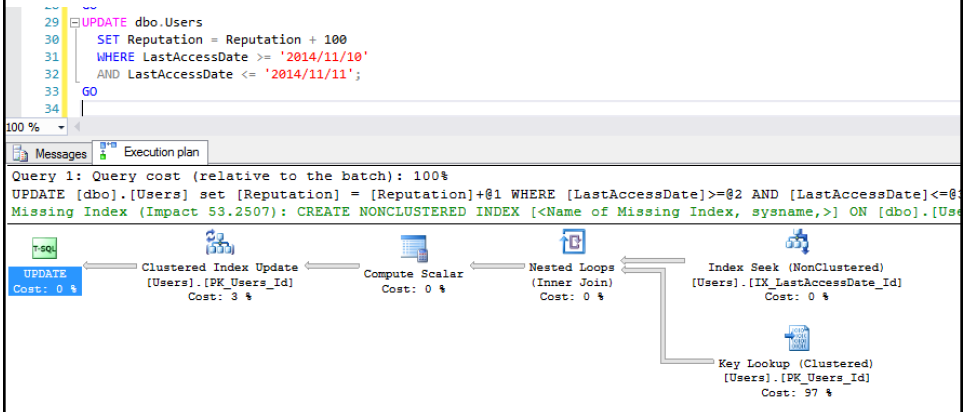
What's the execution plan for this query:

```
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.



So far, so good.

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
```

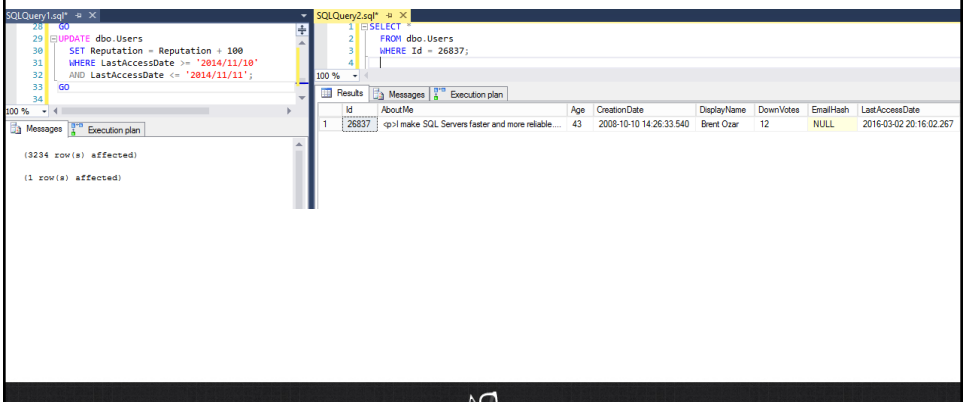
dbo.Users - Clustered Index

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.



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?

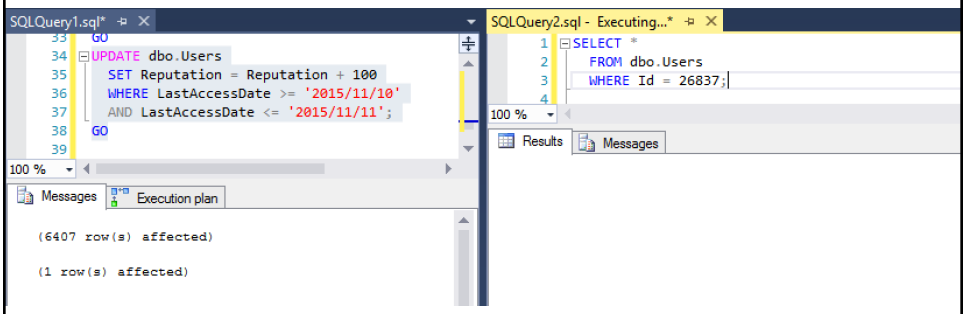
Yes, you've run this one before – but let's try it again:

```
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

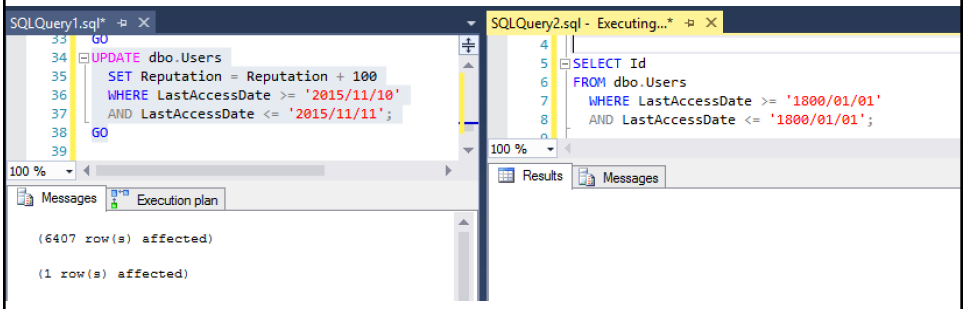
Wait a minute - literally



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.

Can we query by LastAccessDate?



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

Our row-level locks got escalated to table locks.

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.

```
10  SELECT YEAR(LastAccessDate) AS Yr, MONTH(LastAccessDate) AS Mo, DAY(LastAccessDate) AS Dy, COUNT(*) AS Folks
11    FROM dbo.Users
12    WHERE LastAccessDate BETWEEN '2015/11/01' AND '2015/11/30'
13    GROUP BY YEAR(LastAccessDate), MONTH(LastAccessDate), DAY(LastAccessDate)
14    ORDER BY YEAR(LastAccessDate), MONTH(LastAccessDate), DAY(LastAccessDate)
15
```

00 %

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

Additional considerations

Is your data growing over time?

Do your deletes/updates/inserts affect larger and larger numbers of rows?

Are you using variable batch sizes?

Is your WHERE clause SARGable?

Are you doing a function row-by-row,
like DATEPART(month, LastAccessDate) = 7

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

Recap: right-sizing indexes

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

Aim for 5 or less indexes per table,
5 or less fields per index.

Doing batch operations? Watch lock escalation.

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

Isolation levels

SQL Server is pessimistic by default

You've got 3 marbles

An update starts

```
UPDATE dbo.Marbles
SET Color='White'
WHERE MarbleID=2;
```

It takes out locks

```
UPDATE dbo.Marbles
SET Color='White'
WHERE MarbleID=2;
```

Before it completes, a read comes in...

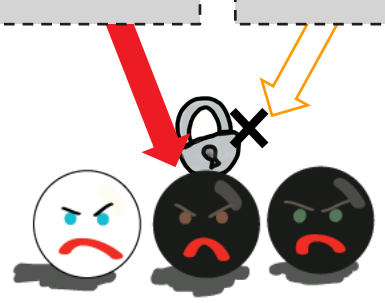
```
UPDATE dbo.Marbles
SET Color='White'
WHERE MarbleID=2;
```

```
SELECT COUNT(*)
FROM dbo.Marbles
WHERE Color='Black'
```

It stops and waits for the lock to be released

```
UPDATE dbo.Marbles
SET Color='White'
WHERE MarbleID=2;
```

```
SELECT COUNT(*)
FROM dbo.Marbles
WHERE Color='Black'
```



The update finishes and releases its locks

```
UPDATE dbo.Marbles
SET Color='White'
WHERE MarbleID=2;
```

```
SELECT COUNT(*)
FROM dbo.Marbles
WHERE Color='Black'
```

The count is 1!

You can choose optimistic locking

You've got 3 marbles

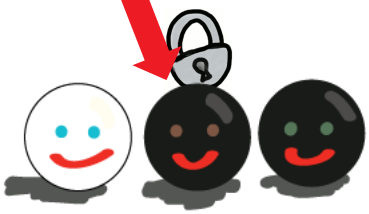
We're optimistic!

An update starts

```
UPDATE dbo.Marbles
SET Color='White'
WHERE MarbleID=2;
```

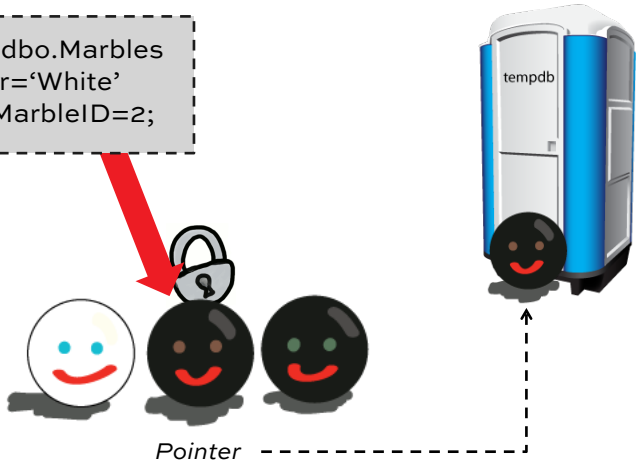
It takes out locks (this is still true)

```
UPDATE dbo.Marbles
SET Color='White'
WHERE MarbleID=2;
```

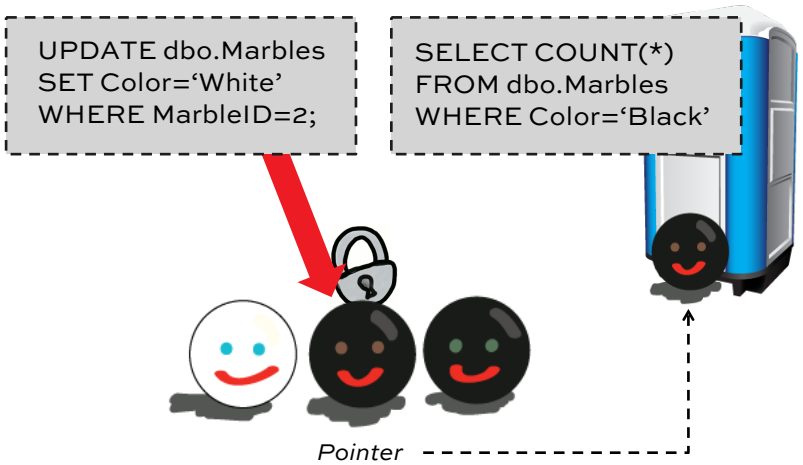


It creates a version for the change in tempdb

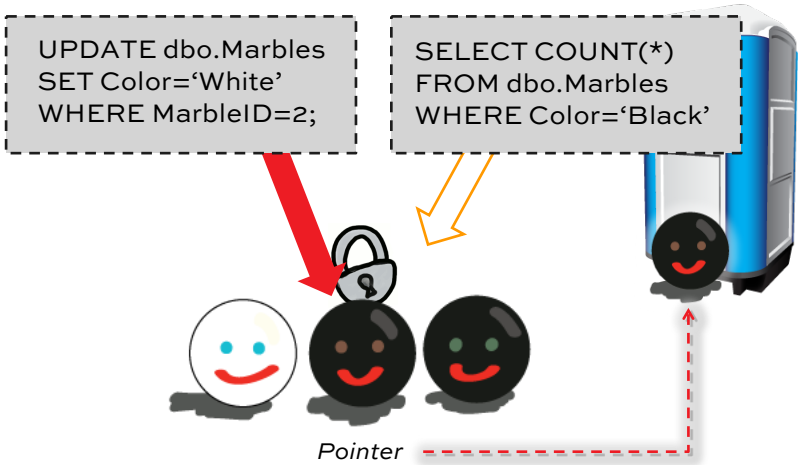
```
UPDATE dbo.Marbles
SET Color='White'
WHERE MarbleID=2;
```



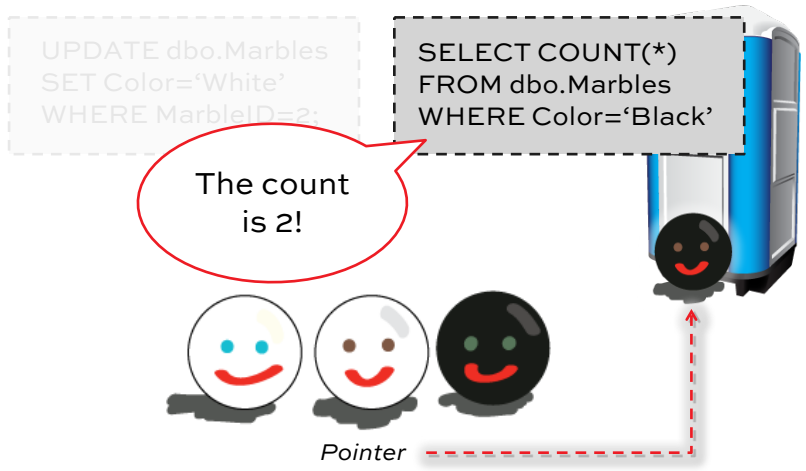
Before it completes, a read comes in...



It follows the version pointer!



They complete at the same time



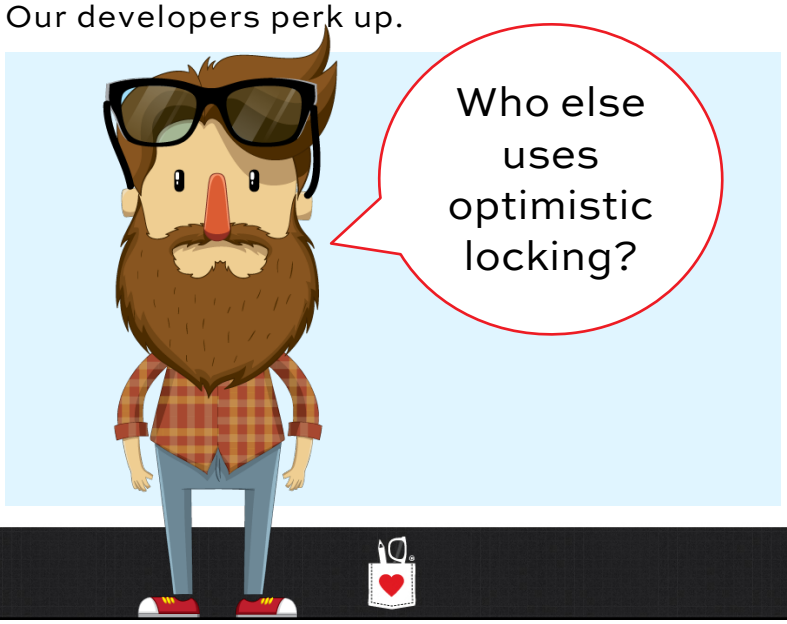
Things changed

- The read completed faster using optimistic locking
- The 'read' query saw different results depending on whether it had to wait:
  - Pessimistic locking: 1 black marble
  - Optimistic locking: 2 black marbles

Two ways to implement

	Read Committed Snapshot Isolation	Snapshot Isolation
Commonly called	"RCSI"	"Snapshot"
What does it do?	Changes the default isolation level for an entire database to optimistic locking	Allows individual transactions to use optimistic locking
What version does a statement read?	The most recent committed version	The committed version consistent with when the transaction began
Can be used for modifications?	NO	YES

Our developers perk up.



Optimistic locking

Windows Azure SQL Database

- It's on by default
- You can't turn it off!

AlwaysOn Availability Group Readable Secondaries

MS recommends optimistic locking if you use:

- Change Tracking
- Change Data Capture

RCSI is fantastic, but....

Most people don't have time to test their whole app

One exception: vendor tools where the app was written for Oracle and/or PostgreSQL too

For other existing applications, SNAPSHOT can be a better fit because of reduced testing

Recap

2 ways to fix LCK* waits

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.

2. Use the right isolation level for your app's needs.

More info: BrentOzar.com/go/lock