



BRENT OZAR
UNLIMITED®

When to Invest in Application Technology

You know... caching.

Your first server...



After a while,
there's load.
And you decide to
buy a new server.



3

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

And you got a bigger server...



4

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

When do you stop looking for a bigger server?



5

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Why do we need a bigger server?

Execute queries faster.

Support more users.

Run multiple complex searches at once.



6

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

The fastest query is the one you never run

What is cache?

A cache should transparently store
data so future requests are served
faster.

Latency is why we need cache

Table 2.2 Example Time Scale of System Latencies

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-state disk I/O (flash memory)	50–150 µs	2–6 days
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1–3 s	105–317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia



Data from Systems Performance by Brendan Gregg

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

9

Examples of Cache

CPU L1 cache

CPU L2 cache

Main memory (RAM)

OS file system cache

SQL Server's buffer pool



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

10

These cache **raw**
data.

These are out of our
direct control.

These don't cache query results.

Why add another cache?

Control

- Cache duration
- Data format – data pages vs query results

Why add another cache?

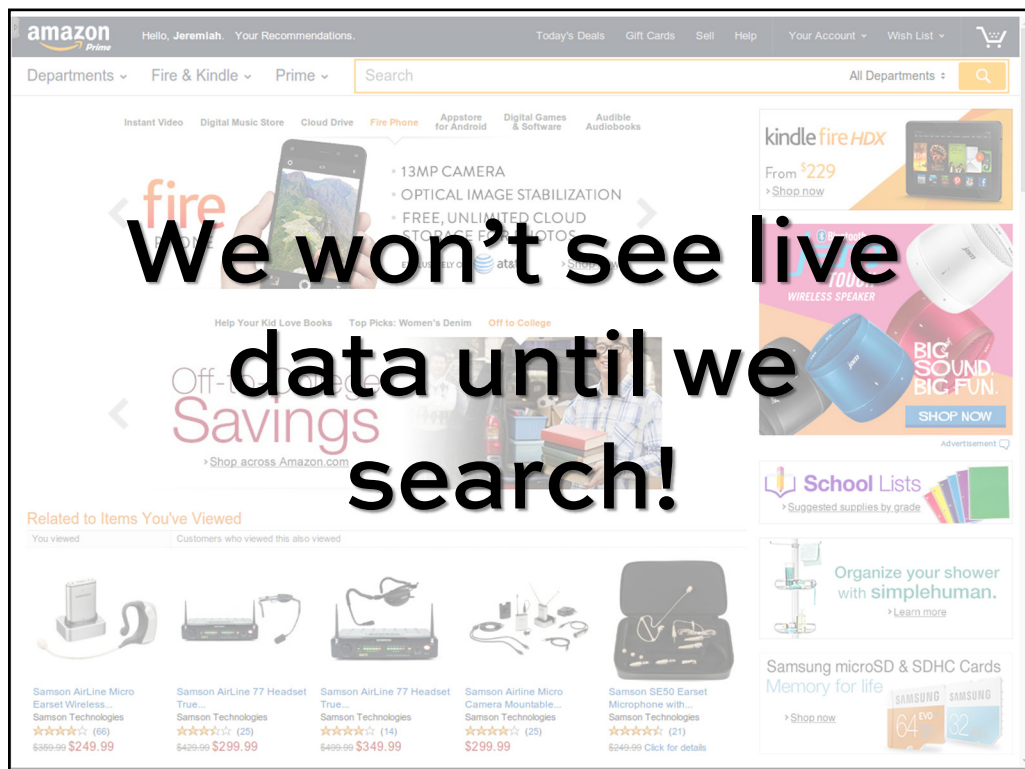
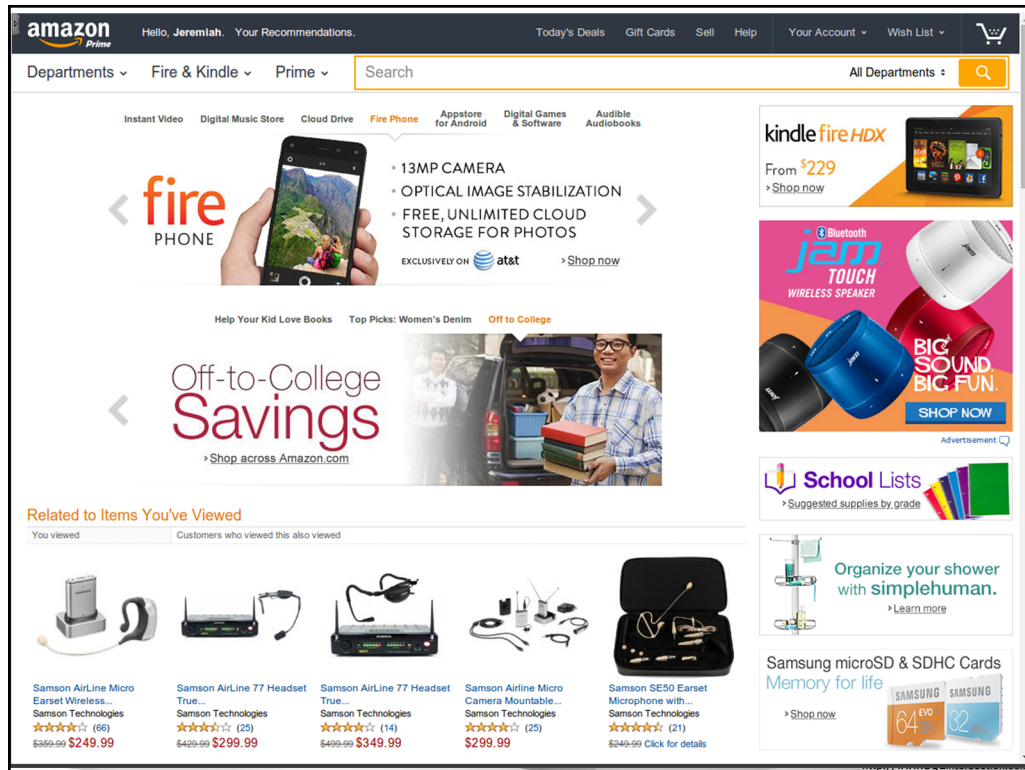
Control

- Cache duration
- Data format – data pages vs query results

Scalability

- Cache should scale separately

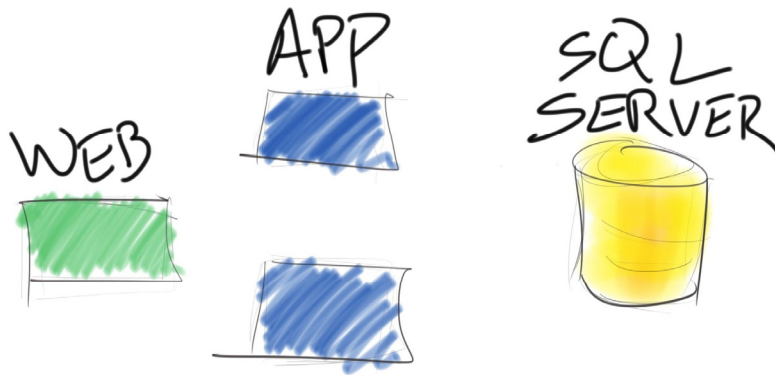
An example of caching



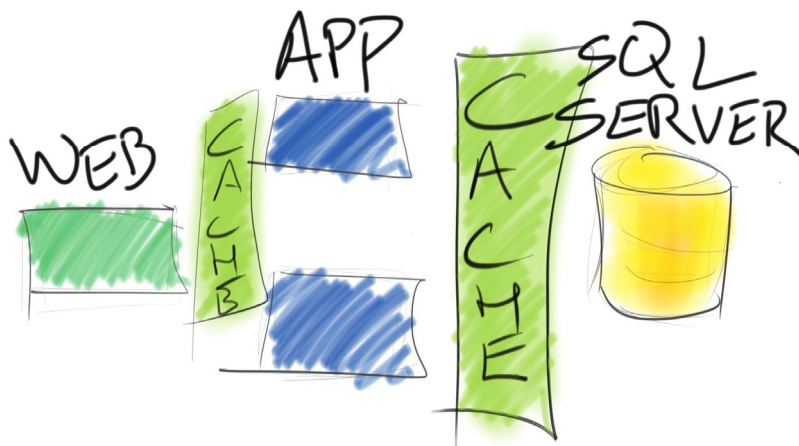
The screenshot shows the Stack Overflow homepage. At the top, there's a navigation bar with 'StackExchange', a reputation indicator '+100', a user profile icon, and statistics '4,808 • 2 • 18 • 47'. There are links for 'review' and 'help', and a search bar. Below the navigation bar, the 'stackoverflow' logo is followed by tabs for 'Questions', 'Tags', 'Users', 'Badges', and 'Unanswered'. An 'Ask Question' button is on the right. The main content area is titled 'Top Questions' and has a filter set to 'interesting' with a count of '389'. It lists several questions with their respective vote counts, answer counts, view counts, tags, and timestamps. The sidebar on the right contains a 'Blog' section with a post about editing enhancements, 'Hot Meta Posts' with a list of recent posts, 'Favorite Tags' with a list of popular tags, and a 'Jobs near you' section with several job listings. At the bottom, there's a footer with the 'SQL intersection' logo, a page number '19', and a copyright notice for 'SQLIntersection, All rights reserved.' with a website URL.

This screenshot is identical to the one above, showing the Stack Overflow homepage. However, a large, bold, black text overlay is centered over the main content area, reading 'What's being cached now?'. The text is positioned over the 'Top Questions' list, partially obscuring the details of the questions. The rest of the page, including the navigation bar, sidebar, and footer, remains the same as in the previous screenshot.

Where should you cache?



Cache Everywhere

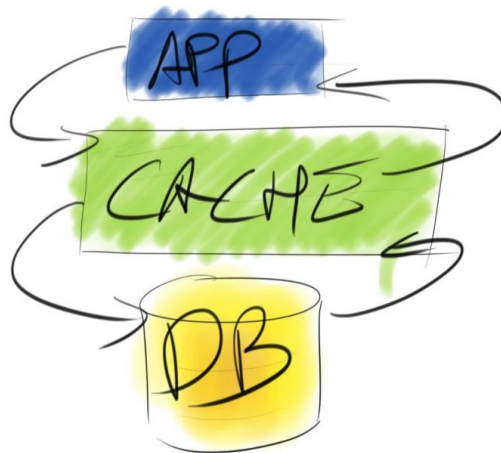


There are two forms of cache

Write Through Cache

Write Aside Cache

Write Through Cache



Write Through Cache: Benefits

Cache coherence

Fast updates

Simple code

Write Through Cache: Drawbacks

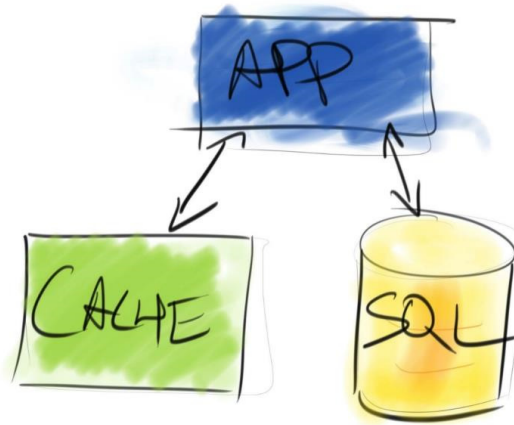
Cache now needs fault tolerance

Strict dependency on language or database

Cache coherence is a myth

May not have complete control over what is cached

Cache Aside



Cache Aside: Benefits

- Application retains full control
- Result sets can be easily cached
- Intermediate work can be stored in cache
- Easy to get started

Cache Aside: Drawbacks

We're responsible for the hard work

- Cache invalidation
- Determining what to cache

Changes are required to app code



29

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Problems with Both Methods

Cache consistency

Cache invalidation

Performance during cache start up



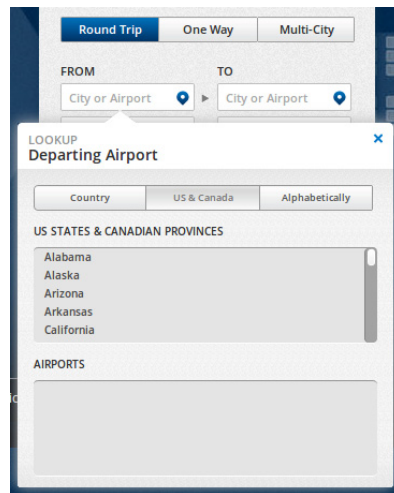
30

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Either Option Reduces Work Done

What Should You Cache?

Dropdown lists
Commonly queried data
Paged reports
Work in progress



The screenshot shows a flight booking form with tabs for 'Round Trip', 'One Way', and 'Multi-City'. Below the tabs are 'FROM' and 'TO' fields, each with a dropdown menu labeled 'City or Airport'. A 'LOOKUP' dialog box is open over the 'FROM' dropdown, titled 'Departing Airport'. It has three tabs: 'Country', 'US & Canada', and 'Alphabetically'. The 'US & Canada' tab is selected, showing a list of 'US STATES & CANADIAN PROVINCES' (Alabama, Alaska, Arizona, Arkansas, California) and an 'AIRPORTS' section below it.

Getting Started with Caching

Identify frequently run queries

Categorize these by:

- Risk
- Frequency
- Frequency of Updates
- Data Longevity

Finding Queries to Cache

SQL Server is already tracking this!

The plan cache can get cleared by many things:
Not all queries will be present.

<http://www.brentozar.com/go/plans/>

Which Queries Should We Cache?

# Execution	Avg CPU	% CPU	Avg Duration	% Duration	Avg Reads	Exec / Min
1	19,055,745,180	32	20,733,815	24	750,443,342	1
1	12,837,730,475	22	7,316,989	8	4,467,732,165	1
201,152,295	22	7	0	5	2	193,190,291
1	3,142,895,504	5	3,266,072	4	2,456,252	1
91,556,589	18	3	0	2	2	88,859,883
80,806,339	19	3	0	2	2	79,444,929
1	1,099,366,224	2	2,153,089	2	167,224,435	1
1	1,014,948,232	2	2,406,363	3	418,868,695	1
1	918,130,859	2	985,988	1	566,702,305	1

Which Queries Should We Cache?

# Execution	Avg CPU	% CPU	Avg Duration	% Duration	Avg Reads	Exec / Min
1	19,055,745,180	32	20,733,815	24	750,443,342	1
1	12,837,730,475	22	7,316,989	8	4,467,732,165	1
201,152,295	22	7	0	5	2	193,190,291
1	3,142,895,504	5	3,266,072	4	2,456,252	1
91,556,589	18	3	0	2	2	88,859,883
80,806,339	19	3	0	2	2	79,444,929
1	1,099,366,224	2	2,153,089	2	167,224,435	1
1	1,014,948,232	2	2,406,363	3	418,868,695	1
1	918,130,859	2	985,988	1	566,702,305	1

Why?

High executions/minute

- Probably a common lookup – e.g. list of states
- Even parameterized queries can be cached

High reads/CPU

- Ability to cache depends on use.
- Requires further investigation and business discussion.

Disk as a Cache

Remember our definition of cache?

A cache should transparently store
data so future requests are served
faster.

SQL Server's Disk as a Cache

Indexed Views

Filtered Indexes

Indexed Views

Very strict implementation rules.

Materialize a view to disk.

Best used when:

- Query patterns are well known.
- Aggregations are common.
- Read heavy workload.
- Storage is abundant.

Filtered Indexes

Move the predicate to the index definition.

Indexes are smaller and more selective.

Strict implementation rules.

Best used with:

- Common searches (e.g. IsActive = 1).
- Careful planning.

References

Picking Your Cache

<http://www.brentozar.com/archive/2012/05/picking-your-cache/>

The Fastest Query is the One You Never Make

<http://www.brentozar.com/archive/2013/02/the-fastest-query-is-the-one-you-never-make/>

Caching in the Distributed Environment

<http://msdn.microsoft.com/en-us/library/dd129907.aspx>

Working with 154mm Records in Azure Table Storage

www.troyhunt.com/2013/12/working-with-154-million-records-on.html

Creating a Flat File database with Amazon S3

<http://seroter.wordpress.com/2013/05/06/creating-a-flat-file-shared-database-with-amazon-s3-and-node-js/>



43

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>