Is 20 SQL Queries per page load really considered a lot? [closed]

Asked 16 years ago Modified 11 years, 8 months ago Viewed 21k times

39

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 10 years ago.

Improve this question

I was reading Jeff Atwood's blog on Behold WordPress, Destroyer of CPUs and saw that many people there considered 20 SQL Queries per page load to be a lot. What is the average amount of queries per page nowadays for a highly dynamic page with auto suggest, auto refreshing of data, customized pages, and the kitchen sink?

For a simple example, Amazon.com practically customizes my homepage with stuff they think I will buy. To me, that doesn't look it just uses 5 or less queries for the front page.

I'm still a newbie with databases so please tell me if I'm missing something obvious.

sql optimization

Share

Improve this question

Follow

edited Dec 5, 2008 at 0:54

Lance Roberts

22.8k ● 32 ● 114 ● 132

asked Dec 5, 2008 at 0:13

danmine

11.5k ● 18 ● 57 ● 75

3 Amazon is a bit of a special case - never mind SQL queries, how many web service requests does it do per hit on the home page? I've heard 100+ but I can't find a link. The significant metric is load time, not number of requests, and Amazon has considerably more silicon than you...
– Steve Jessop Dec 5, 2008 at 2:13

9 Answers

Sorted by: Highest score (default)

29

You can usually bring all the data in two or three big queries instead of on twenty small ones. Minimizing the amount of queries is as important as, if not most important than, writing optimal queries to maximize performace.

Of course you should always analyze the query plans and aim towards optimal queries, be them small or big.

The thing is that badly designed webpages do many queries, one per each tiny little task, which could easily be grouped in a single query.

For example, a **badly designed stackoverflow could** do a query to get all the question ids it will show on the main page, then do one query per each question to get the summary and the votes. Then you have easily 20 useless queries. A well designed will do a single query getting all the information about all the questions it'll display.

Of course the impact of this all is reduced with good caching, which is what all big sites do, that way you actually can do a lot of queries and still get decent performance.

Share Improve this answer Follow

edited Dec 5, 2008 at 0:25

answered Dec 5, 2008 at 0:20

Vinko Vrsalovic

340k ● 55 ● 340 ● 373

8 problem with combing lots of queries is you need to be careful how you combine code. Don't combine unrelated things otherwise you might get coupled code that becomes difficult to break apart. – JoshBerke Dec 5, 2008 at 0:59

I do fully agree with the principle you are stating. I'd add that it depends on how unrelated things are, and on how big (and how needed) a performance gain can be achieved. You can

always break things apart on code instead of on SQL if it's really necessary. – Vinko Vrsalovic Dec 5, 2008 at 9:31

12

It's more about caching.

If you're getting a high number of concurrent page views, and each page view does a lot of queries, it doesn't make a lot of sense to hit the database **every. single. time.** Especially when a lot of the data coming back will by semi-dynamic reference data that only changes every now and then (as opposed to session or real time data which is always changing).

You may as well cache those database results using memcached or something similar. You don't necessarily need to cache the whole page (although that is what most Wordpress caching plugins do), as this kills interactivity, but you can cache on a data-by-data basis.

There's also the issue of optimising the queries. Especially avoiding the dreaded N+1 situation where you do one query for a parent record, then an extra query for *each* of its children. The latency of the round trip back and forth to the database alone will kill your page rendering performance, not to mention cause grief on the DB itself.

Share Improve this answer

Follow

answered Dec 5, 2008 at 0:36

madlep

49.5k ● 7 ● 44 ● 53

11

I'm always late comer in a party, this is kind of 5 years late...

But to-the-point answer to this question would be that NUMBER OF QUERIES MATTERS LESS THAN TOTAL TIME TAKEN BY THE QUERIES.

If a large query with multiple joins and sub queries takes 20 secs to execute, then (I think) 20 small queries which take .20 secs in all are much better.

I find manage smaller queries much easier, well mainly because I cache every query, and I can reuse the data from that individual query again and again and again.....

Share Improve this answer

Follow

answered Apr 23, 2013 at 5:48

Amit Kriplani

682 ● 5 ● 12

> But, there is a cost to sending the request to mysql for the query, mysql parsing the query. That cost would be repeated once per query. So even if the queries are small, depending the distance between mysql and the webserver, the sheer amount of these queries will cause slowness.
> – Joe Yahchouchi Mar 30, 2015 at 7:07

5

The answer really depends on a few key things: - The amount of traffic of your site - The IT budget for your support - The complexity of the site and the resources required to optimize

If you have a website that gets a few hits a day, then who cares about 20 queries. On the flip side, if you are Amazon then you are going to offer the needed content at a large infrastructure cost.

Just about everyone else in the world is somewhere between those two extremes and has to balance based on their own resources.

The only other thing I'll say is caching is your friend.

Share Improve this answer

Follow

answered Dec 5, 2008 at 0:26

Darian Miller
7,948 ● 3 ● 48 ● 63

3

It depends on the type of application you are building, the complexity of the queries and the things your database engine and server lets you do.

If your database service only allows you to make simple SQL queries, less than 20 queries would be fine for a small, common webpage, but if it's the webpage for your university or a decision taking support application, 60 may not be enough.

If you have the privileges and your DBMS is capable (Oracle and such, compared to older versions of MySql for example), more than 20 queries asks that you start creating stored procedures, functions and triggers for the heavy tasks. In many cases you can't, so the number of

queries naturally grows and you start using cache to ease the pressure on the server.

Some heavy tasks can be achieved in less queries using subqueries for example, but they are really heavy on the database engine. They aren't really recomended in some cases and should be used with care if they involve thousands of records.

The example up there from Vinko may be true for small, 1 week development "projects", but if you ask about Amazon, they don't use your common PHP / MySQL development package; behind the frontdoor lies a complex system of distributed computing and data-mining algorytms. If you're a newbie you shouldn't take big brothers like that for a reference...

Share Improve this answer

Follow

answered Dec 11, 2010 at 14:26

DanyAlejandro

1,460 ● 13 ● 25

1

If you have to do 20 queries, then so be it, but it would make me a little nervous if it were a front page.

Combining queries where possible can help, but thinking about the caching is the most important part.

I am currently upgrading a site where data that changes 5 or 6 times a year is queried thousands of times a day, using some very nasty SQL to make it into a tree, but can be held as a tree structure in about 200k of RAM. (700k

of viewstate on front page too, but that's another story...) These are the kind of things that cripple web sites for no good reason.

So, there is no magic number as to how many queries you should or should not do, but think about every one of them, even if you cache some of them for only 5 minutes, that will make a huge difference if ever you hit the front page of digg.

5 minutes of caching on just 1 query could remove thousands of DB hits when your site is under stress.

Share Improve this answer

Follow

answered Dec 5, 2008 at 2:02

seanb

6,954 ● 2 ● 34 ● 34

1

Given that, short of using Ajax, each page is atomic, I haven't found it that difficult to generate quite complex pages in 3 or less queries. Conceptually, a typical page-set involves:

1. Context info (related to session and other global state);
2. Header (and related 1:0-1 joins);
3. Detail (1:M from 2).

It takes some planning ahead; but on the other hand it's an easy refactoring exercise in most cases.

Share Improve this answer

Follow

answered Dec 5, 2008 at 2:13

dkretz

37.6k ● 13 ● 83 ● 140

0

My rule of thumb is keep the front pages down to under 5-7 if possible, depending on the type of the site.

Interior pages, depending on what they need might have more, but I do what I can to keep it under 20.

However, at the same time, depending on what you are trying to do AND what types of caching you are doing with that information 20 may not be bad if 15 of them are heavily cached...

Share Improve this answer

Follow

answered Dec 5, 2008 at 0:20

Mitchel Sellers

63.1k ● 15 ● 114 ● 174

0

The number of queries isn't so important all the time. It's really how you handle connections. If you have connection pooling then it really doesn't matter and the physical location of the servers matters. If your servers are next to eachother in a data center setting up a connection is probably really fast. Most of the time your website spends loading if it's a database driven site is going to be spent waiting for connections to open and for the data to be fetched. Figure to open a connection it takes 100 - 300ms. So if you have to open 20

connections for each database access, that's 4 - 6 seconds just opening and closing connections.

Since Jeff Atwood is using LINQ, I'm assuming he's only opening a single connection, executing his 20 queries and then closing the connection. It all probably happens pretty quick.

Also, Jeff's database runs on the same physical machine and uses internal machine communication to communicate with the database and not a network so there really isn't any delay you'd associate with the TCP type connection opening. (He talked about this on the Hanselminutes podcast a few weeks ago.)

I have a similar configuration for one of my sites using LINQ and with the database on the same box. When I run the site on my local machine hitting the database on the server in another state, it takes up to 6 seconds to load a couple of the data heavy pages. When I run the site on the server, the page loads in less than a second because everything is local to the server.

edited Jun 18, 2010 at 20:19

answered Dec 5, 2008 at 3:51

Paul Mendoza
5,787 12 56 85