What is a good way to denormalize a mysql database?

Asked 16 years, 4 months ago Modified 10 years, 11 months ago

Viewed 13k times

22

I have a large database of normalized order data that is becoming very slow to query for reporting. Many of the queries that I use in reports join five or six tables and are having to examine tens or hundreds of thousands of lines.

There are lots of queries and most have been optimized as much as possible to reduce server load and increase speed. I think it's time to start keeping a copy of the data in a denormalized format.

Any ideas on an approach? Should I start with a couple of my worst queries and go from there?

mysql database denormalization

Share

Improve this question

Follow

edited Jan 15, 2014 at 8:01

Sajad Karuthedath

15.7k ● 4 ● 34 ● 49

asked Aug 15, 2008 at 23:36

Eric Goodwin

2,986 ● 6 ● 25 ● 22

8 Answers

Sorted by: Highest score (default)

12

I know more about mssql that mysql, but I don't think the number of joins or number of rows you are talking about should cause you too many problems with the correct indexes in place. Have you analyzed the query plan to see if you are missing any?

http://dev.mysql.com/doc/refman/5.0/en/explain.html

That being said, once you are satisifed with your indexes and have exhausted all other avenues, de-normalization might be the right answer. If you just have one or two queries that are problems, a manual approach is probably appropriate, whereas some sort of data warehousing tool might be better for creating a platform to develop data cubes.

Here's a site I found that touches on the subject:

http://www.meansandends.com/mysql-data-warehouse/?link_body%2Fbody=%7Bincl%3AAggregation%7D

Here's a simple technique that you can use to keep denormalizing queries simple, if you're just doing a few at a time (and I'm not replacing your OLTP tables, just creating a new one for reporting purposes). Let's say you have this query in your application:

```
select a.name, b.address from tbla a
join tblb b on b.fk_a_id = a.id where a.id=1
```

You could create a denormalized table and populate with almost the same query:

```
create table tbl_ab (a_id, a_name, b_address);
-- (types elided)
```

Notice the underscores match the table aliases you use

```
insert tbl_ab select a.id, a.name, b.address from tbla
join tblb b on b.fk_a_id = a.id
-- no where clause because you want everything
```

Then to fix your app to use the new denormalized table, switch the dots for underscores.

```
select a_name as name, b_address as address
from tbl_ab where a_id = 1;
```

For huge queries this can save a lot of time and makes it clear where the data came from, and you can re-use the queries you already have.

Remember, I'm only advocating this as the last resort. I bet there's a few indexes that would help you. And when you de-normalize, don't forget to account for the extra space on your disks, and figure out when you will run the query to populate the new tables. This should probably be at night, or whenever activity is low. And the data in that table, of course, will never exactly be up to date.

[Yet another edit] Don't forget that the new tables you create need to be indexed too! The good part is that you can index to your heart's content and not worry about update lock contention, since aside from your bulk insert the table will only see selects.

Share Improve this answer Follow

edited Aug 16, 2008 at 1:55

answered Aug 16, 2008 at 1:36

Eric Z Beard

38.4k ● 27 ● 101 ● 147

2

MySQL 5 does support views, which may be helpful in this scenario. It sounds like you've already done a lot of optimizing, but if not you can use MySQL's EXPLAIN syntax to see what indexes are actually being used and what is slowing down your queries.

As far as going about normalizing data (whether you're using views or just duplicating data in a more efficient

manner), I think starting with the slowest queries and working your way through is a good approach to take.

Share Improve this answer

Follow

answered Aug 16, 2008 at 0:04

pix0r

31.3k • 18 • 87 • 102

1

I know this is a bit tangential, but have you tried seeing if there are more indexes you can add?

I don't have a lot of DB background, but I am working with databases a lot recently, and I've been finding that a lot of the queries can be improved just by adding indexes.

We are using DB2, and there is a command called db2expln and db2advis, the first will indicate whether table scans vs index scans are being used, and the second will recommend indexes you can add to improve performance. I'm sure MySQL has similar tools...

Anyways, if this is something you haven't considered yet, it has been helping a lot with me... but if you've already gone this route, then I guess it's not what you are looking for.

Another possibility is a "materialized view" (or as they call it in DB2), which lets you specify a table that is essentially built of parts from multiple tables. Thus, rather than normalizing the actual columns, you could provide this view to access the data... but I don't know if this has severe performance impacts on inserts/updates/deletes

(but if it is "materialized", then it should help with selects since the values are physically stored separately).

Share Improve this answer

Follow

answered Aug 15, 2008 at 23:58

Mike Stone

44.6k ● 30 ● 114 ● 140

1

In line with some of the other comments, i would definately have a look at your indexing.

One thing i discovered earlier this year on our MySQL databases was the power of composite indexes. For example, if you are reporting on order numbers over date ranges, a composite index on the order number and order date columns could help. I believe MySQL can only use one index for the query so if you just had separate indexes on the order number and order date it would have to decide on just one of them to use. Using the EXPLAIN command can help determine this.

To give an indication of the performance with good indexes (including numerous composite indexes), i can run queries joining 3 tables in our database and get almost instant results in most cases. For more complex reporting most of the queries run in under 10 seconds. These 3 tables have 33 million, 110 million and 140 millions rows respectively. Note that we had also already normalised these slightly to speed up our most common query on the database.

More information regarding your tables and the types of reporting queries may allow further suggestions.

Share Improve this answer Follow

edited Aug 16, 2008 at 2:41

answered Aug 16, 2008 at 2:30

Jarod Elliott

15.7k ● 3 ● 36 ● 34

1

For MySQL I like this talk: Real World Web: Performance & Scalability, MySQL Edition. This contains a lot of different pieces of advice for getting more speed out of MySQL.

Share Improve this answer Follow

answered Aug 18, 2008 at 15:03

Peter Stuifzand

5,084 ● 1 ● 25 ● 28

0

You might also want to consider selecting into a temporary table and then performing queries on that temporary table. This would avoid the need to rejoin your tables for every single query you issue (assuming that you can use the temporary table for numerous queries, of course). This basically gives you denormalized data, but if you are only doing select calls, there's no concern about data consistency.

Share Improve this answer

answered Aug 16, 2008 at 0:20

Follow

Derek Park
46.8k ● 16 ● 59 ● 76

0

Further to my previous answer, another approach we have taken in some situations is to store key reporting data in separate summary tables. There are certain reporting queries which are just going to be slow even after denormalising and optimisations and we found that creating a table and storing running totals or summary information throughout the month as it came in made the end of month reporting much quicker as well.

We found this approach easy to implement as it didn't break anything that was already working - it's just additional database inserts at certain points.

Share Improve this answer

Follow

answered Aug 16, 2008 at 2:53

Jarod Elliott
15.7k ● 3 ● 36 ● 34

0

I've been toying with composite indexes and have seen some real benefits...maybe I'll setup some tests to see if that can save me here..at least for a little longer.

Share Improve this answer

Follow

answered Aug 17, 2008 at 16:52

Eric Goodwin
2,986 ● 6 ● 25 ● 22