MySQL: Advisable number of rows

Asked 16 years, 3 months ago Modified 15 years ago Viewed 5k times

1

Consider an indexed MySQL table with 7 columns, being constantly queried and written to. What is the advisable number of rows that this table should be allowed to contain before the performance would be improved by splitting the data off into other tables?

mysql performance database-design optimization

Share
Improve this question
Follow

edited Dec 10, 2009 at 20:05
Peter Mortensen
31.6k ● 22 ● 109 ● 133

asked Sep 20, 2008 at 15:34
tags2k
84.2k ● 31 ● 80 ● 106

8 Answers

Sorted by: Highest score (default)

11

Whether or not you would get a performance gain by partitioning the data depends on the data and the queries you will run on it. You can store many millions of rows in a table and with good indexes and well-designed queries it will still be super-fast. Only consider partitioning if you are already confident that your indexes and queries are as good as they can be, as it can be more trouble than its worth.

Share Improve this answer Follow

answered Sep 20, 2008 at 15:42

Matt Howells
41.3k ● 20 ● 85 ● 104

1 I've accepted this answer because it seems the most pragmatic. Other people have given factors to look out for which is useful, but you can't always be around when things go wrong, so I feel that the best literal answer to my question is "a few million". – tags2k Sep 21, 2008 at 7:50

3

There's no magic number, but there's a few things that affect performance in particular:

- Index Cardinality: don't bother indexing a row that has 2 or 3 values (like an ENUM). On a large table, the query optimizer will ignore these.
- There's a trade off between writes and indexes. The more indexes you have, the longer writes take. Don't just index every column. Analyze your queries and see which columns need to be indexed for your app.
- Disk IO and a memory play an important role. If you can fit your whole table into memory, you take disk IO out of the equation (once the table is cached, anyway). My guess is that you'll see a big performance change when your table is too big to buffer in memory.

- Consider partitioning your servers based on use. If your transactional system is reading/writing single rows, you can probably buy yourself some time by replicating the data to a read only server for aggregate reporting.

As you probably know, table performance changes based on the data size. Keep an eye on your table/queries. You'll know when it's time for a change.

Share Improve this answer Follow

answered Sep 20, 2008 at 17:19
Gary Richardson
16.4k ● 10 ● 54 ● 48

2

MySQL 5 has partitioning built in and is very nice. What's nice is you can define how your table should be split up. For instance, if you query mostly based on a userid you can partition your tables based on userid, or if you're querying by dates do it by date. What's nice about this is that MySQL will know exactly which partition table to search through to find your values. The downside is if you're search on a field that isn't defining your partition its going to scan through each table, which could possibly decrease performance.

Share Improve this answer Follow

answered Sep 22, 2008 at 14:07
Jacob
10.7k ● 5 ● 24 ● 11

0

While after the fact you could point to the table size at which performance became a problem, I don't think you can predict it, and certainly not from the information given on a web site such as this!

Some questions you might usefully ask yourself:

- Is performance currently acceptable?
- How is performance measured - is there a metric?
- How do we recognise unacceptable performance?
- Do we measure performance in any way that might allow us to forecast a problem?
- Are all our queries using an efficient index?
- Have we simulated extreme loads and volumes on the system?

Share Improve this answer Follow

answered Sep 20, 2008 at 16:31

Mike Woodhouse
52.3k ● 12 ● 89 ● 127

0

Using the MyISAM engine, you'll run into a 2GB hard limit on table size unless you change the default.

Share Improve this answer Follow

answered Sep 20, 2008 at 16:34

Ian P
13k ● 6 ● 50 ● 70

0

Don't ever apply an optimisation if you don't think it's needed. Ideally this should be determined by testing (as others have alluded).

Horizontal or vertical partitioning can improve performance but also complicate you application. Don't do it unless you're sure that you need it AND it will definitely help.

The 2G data MyISAM file size is only a default and can be changed at table creation time (or later by an ALTER, but it needs to rebuild the table). It doesn't apply to other engines (e.g. InnoDB).

Share Improve this answer Follow

answered Sep 20, 2008 at 20:26

MarkR
63.5k ● 15 ● 119 ● 154

"Don't ever apply an optimisation if you don't think it's needed" - Perhaps in the context of splitting off a database table and being unsure as to what it will do, but in general

surely this is terrible advice? Wait for something to go wrong/slow before you then spend a few hours/days fixing it?
– tags2k Sep 21, 2008 at 7:46

I'm not advocating "waiting for something to go wrong", but rather doing some performance testing to evaluate whether further optimisation is needed. In many cases people apply unnecessary optimisations, which add code complexity (reducing maintainability and increasing the chance of bugs) .
– MarkR Sep 21, 2008 at 8:57

0

Actually this is a good question for performance. Have you read Jay Pipes? There isn't a specific number of rows but there is a specific page size for reads and there can be good reasons for vertical partitioning.

Check out his kung fu presentation and have a look through his posts. I'm sure you'll find that he's written some useful advice on this.

Share Improve this answer Follow

answered Sep 20, 2008 at 20:46

Polsonby
22.9k ● 19 ● 60 ● 74

0

Are you using MyISAM? Are you planning to store more than a couple of gigabytes? Watch out for MAX_ROWS and AVG_ROW_LENGTH.

Jeremy Zawodny has an excellent write-up on how to solve this problem.

Share Improve this answer

Follow

answered Sep 20, 2008 at 20:54

mbac32768

11.6k ● 9 ● 37 ● 41