What are some best practices and “rules of thumb” for creating database indexes?

Asked 10 years, 11 months ago Active 3 months ago Viewed 17k times

16

14

I have an app, which cycles through a huge number of records in a database table and performs a number of SQL and .Net operations on records within that database (currently I am using Castle.ActiveRecord on PostgreSQL).

I added some basic btree indexes on a couple of the feilds, and as you would expect, the performance of the SQL operations increased substantially. Wanting to make the most of dbms performance I want to make some better educated choices about what I should index on all my projects.

I understand that there is a detrement to performance when doing inserts (as the database needs to update the index, as well as the data), but what suggestions and best practices should I consider with creating database indexes? How do I best select the feilds/combination of fields for a set of database indexes (rules of thumb)?

Also, how do I best select which index to use as a clustered index? And when it comes to the access method, under what conditions should I use a btree over a hash or a gist or a gin (what are they anyway?).

database database-design indexing

edited Apr 9 '17 at 9:54

Cœur
26.9k ● 14 ● 148 ● 204

asked Mar 26 '09 at 23:55

Ash
19.3k ● 31 ● 94 ● 142

3 Answers

38

Some of my rules of thumb:

- Index ALL primary keys (I think most RDBMS do this when the table is created).
- Index ALL foreign key columns.
- Create more indexes ONLY if:
  - Queries are slow.
  - You know the data volume is going to increase significantly.
- Run statistics when populating a lot of data in tables.

If a query is slow, look at the execution plan and:

- If the query for a table only uses a few columns, put all those columns into an index, then you can help the RDBMS to only use the index.
- Don't waste resources indexing tiny tables (hundreds of records).
- Index multiple columns in order from high cardinality to less. This means: first index the columns with more distinct values, followed by columns with fewer distinct values.

- If a query needs to access more than 10% of the data, a full scan is normally better than an index.

edited Nov 16 '19 at 2:28

CarenRose
994 ● 9 ● 18

answered Mar 27 '09 at 0:05

FerranB
28.7k ● 18 ● 61 ● 81

6 This answer is a good start but in some ways is a little simplistic. Indexing all foreign keys is not always right because sometimes the data is always traversed from fk to parent. Always putting highest cardinality first is useless if never queried on. And several more things. Reading this post might be useful to some readers. – ErikE Jun 5 '11 at 17:05

3

Here's a slightly simplistic overview: it's certainly true that there is an overhead to data modifications due to the presence of indexes, but you ought to consider the relative number of reads and writes to the data. In general the number of reads is far higher than the number of writes, and you should take that into account when defining an indexing strategy.

When it comes to which columns to index I'v e always felt that the designer ought to know the business well enough to be able to take a very good first pass at which columns are likely to benefit. Other then that it really comes down to feedback from the programmers, full-scale testing, and system monitoring (preferably with extensive internal metrics on performance to capture long-running operations),

answered Mar 26 '09 at 23:59

David Aldridge
46.6k ● 8 ● 52 ● 85

2

As @David Aldridge mentioned, the majority of databases perform many more reads than they do writes and in addition, appropriate indexes will often be utilised even when performing INSERTS (to determine the correct place to INSERT).

The critical indexes under an unknown production workload are often hard to guess/estimate, and a set of indexes should not be viewed as set once and forget. Indexes should be monitored and altered with changing workloads (that new killer report, for instance).

Nothing beats profiling; if you guess your indexes, you will often miss the really important ones.

As a general rule, if I have little idea how the database will be queried, then I will create indexes on all Foriegn Keys, profile under a workload (think UAT release) and remove those that are not being used, as well as creating important missing indexes.

Also, make sure that a scheduled index maintenance plan is also created.

answered Mar 27 '09 at 0:09

Mitch Wheat
268k ● 38 ● 422 ● 509

What do you mean with 'a scheduled index maintenance plan'? – tuinstoel Mar 27 '09 at 7:30

@tuinstoel: not sure how I can describe that any more accurately. All production DB's should have fragmented indexes rebuilt as part of a regular maintenance plan. – Mitch Wheat Mar 27 '09 at 7:48

1 I think that is db vendor specific. Rebuilding btree indexes is considered a bad practice in the Oracle

world. – tuinstoel Mar 27 '09 at 8:59

@tuinstoel: ah OK, I had my sql server hat on! – Mitch Wheat Mar 27 '09 at 9:00