Using boolean or enum columns in indices?

Asked 16 years, 1 month ago Modified 13 years, 5 months ago

Viewed 3k times

11

I`ve read that columns that are chosen for indices should discriminate well among the rows, i.e. index columns should not contain a large number of rows with the same value. This would suggest that booleans or an enum such as gender would be a bad choice for an index.

But say I want to find users by gender and in my particular database, only 2% of the users are female, then in that case it seems like the gender column would be a useful index when getting the female users, but not when getting all the male users.

So would it generally be a good idea to put an index on such a column?

database indexing

Share

Improve this question

Follow

edited Jul 23, 2011 at 12:58

Jonathan Leffler

752k ● 145 ● 946 ● 1.3k

asked Nov 20, 2008 at 4:17

Dónal
187k ● 176 ● 581 ● 843

3 Answers

Sorted by: Highest score (default)

3

Indexing a low-cardinality column to improve search performance is common in my world. Oracle supports a "bitmapped index" which is designed for these situations. See this article for a short overview.

Most of my experience is with Oracle, but I assume that other RDBMS' support something similar.

Share Improve this answer

Follow

answered Nov 20, 2008 at 4:40

JPLemme
4,484 ● 6 ● 32 ● 35

2

Don't forget, though, that you'll probably only be selecting for females about 2% of the time. The rest of the time, you'll be searching for males. And for that, a straight table scan (rather than an index scan plus accessing the data from the table) is going to be quicker.

You can also, sometimes, use a compound index, with a low cardinality column (enum, boolean) coupled with a higher cardinality column (birth date, perhaps). This depends very much on the full data, and the queries you'll really use.

My experience is that an index on male/female is seldom going to be truly useful. And the general advice is valid. One more point to remember - indexes have to be maintained when you add or remove (or update) rows. The more indexes, the more work each modify operation has to do, slowing the system down.

There are whole books on index design.

Share Improve this answer

Follow

answered Nov 20, 2008 at 4:43

Jonathan Leffler

752k ● 145 ● 946 ● 1.3k

Your answer is good but consider, instead of gender, we are storing major cities or states, which are only 100 in number, distributed amongs 1 million users, so probably 10 thousand users will have same value, and if we are looking only for one particular city, then I do not want DB to iterate 1 million rows, and normal b+index will be very bad for this purpose, so what will be your suggestion in this case? – Akash Kava Jul 23, 2011 at 9:29

@Akash: see the other two answers - a bitmap index might be appropriate, but it depends on what your queries are. Are you looking to return all 10,000 users for the single city? Or are you doing some statistics on the users from that ciry? Or ... – Jonathan Leffler Jul 23, 2011 at 13:02

thanks for your answer, am looking for search within single city, but problem I am having is SQL server does not have bitmap index, I am not sure I havent seen new features in latest SQL, I hope it's there. – Akash Kava Jul 23, 2011 at 19:44

1

This is a case where I would let the server statistics inform me of when to create the index. Unless you know that this query is going to predominate or that running such a query would not meet your performance goals a priori, then creating the index prematurely may just cost you performance rather than increase it. Also, you may want to think about how you would actually use the query. In this case, my guess would be that you'd typically be doing some sort of aggregation based on this column rather than simply selecting the users who meet the criteria. In that event, you'll be doing the table scan anyway and the index won't buy you anything.

Share Improve this answer

Follow

answered Nov 20, 2008 at 4:24

tvanfosson

532k ● 102 ● 699 ● 798