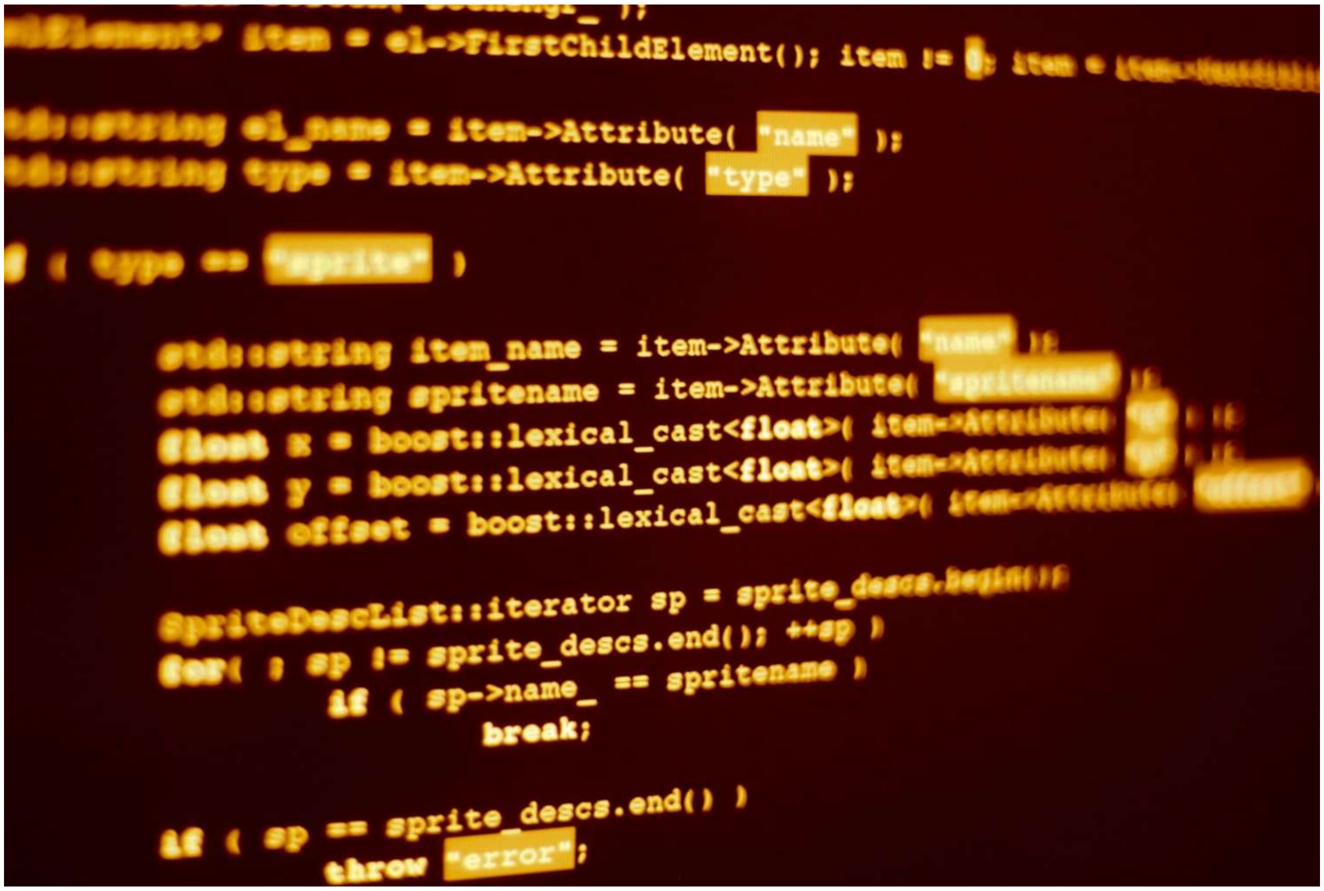
3 ways MySQL uses indexes

percona.com/blog/3-ways-mysql-uses-indexes

September 12, 2009



I often see people confuse different ways MySQL can use indexing, getting wrong ideas on what query performance they should expect. There are 3 main ways how MySQL can use the indexes for query execution, which are not mutually exclusive, in fact some queries will use indexes for all 3 purposes listed here.

Using index to find rows The main purpose of the index is to find rows quickly – without scanning whole data set. This is most typical reason index gets added on the first place. Most popular index type in MySQL – BTREE can speed up equality and prefix range matches. So if you have index on **(A,B)** This index can be used to lookup rows for WHERE clauses like **A=5** ; **A BETWEEN 5 AND 10** ; **A=5 AND B BETWEEN 5 AND 10** it however will **NOT** be able to help lookup rows for **B BETWEEN 5 AND 10** predicate because it is not index prefix. It is important to look at **key_len** column in explain plan to see how many index parts are actually used for row lookup. Very common problem I see is multi column indexes which are used but only to their short prefix which is not very selective. A lot of this mistakes come from missing one very important MySQL limitation – once MySQL runs into the interval range it will not use any further index parts. If you have **A BETWEEN 5 AND 10 AND B=5**

for the same index MySQL will use the index… but it will only use **A** prefix for row lookups and scan whole **A BETWEEN 5 AND 10** range. It is interesting to note this limitation only applies to interval ranges – for enumerated ranges MySQL will use both key parts. Hence if you change this predicate to **A IN (5,6,7,8,9,10) AND B=5** you will quite likely see improved query performance. Beware however of large nested enumerated ranges they are very hard on the optimizer. This just describes how MySQL uses single index – there are more complex rules of how indexes will be used if you look at multiple indexes usage with "index merge"

Using Index to Sort Data Another great benefit of BTREE index is – it allows to retrieve data in sorted form hence avoiding external sort process for executing of queries which require sorting. Using index for sorting often comes together with using index to find rows, however it can also be used just for sort for example if you're just using ORDER BY without and where clauses on the table. In such case you would see "Index" type in explain which correspond to scanning (potentially) complete table in the index order. It is very important to understand in which conditions index can be used to sort data together with restricting amount of rows. Looking at the same index **(A,B)** things like **ORDER BY A** ; **ORDER BY A,B** ; **ORDER BY A DESC, B DESC** will be able to use full index for sorting (note MySQL may not select to use index for sort if you sort full table without a limit). However **ORDER BY B** or **ORDER BY A, B DESC** will not be able to use index because requested order does not line up with the order of data in BTREE. If you have both restriction and sorting things like this would work **A=5 ORDER BY B** ; **A=5 ORDER BY B DESC; A>5 ORDER BY A** ; **A>5 ORDER BY A,B** ; **A>5 ORDER BY A DESC** which again can be easily visualized as scanning a range in BTREE. Things like this however would not work **A>5 ORDER BY B** , **A>5 ORDER BY A,B DESC** or **A IN (3,4) ORDER BY B** – in these cases getting data in sorting form would require a bit more than simple range scan in the BTREE and MySQL decides to pass it on. There are some workarounds you can use though.

Using index to read data Some storage engines (MyISAM and Innodb included) can also use index to read the data, hence avoiding to read the row data itself. This is not simply savings of having 2 reads per index entry instead of one but it can save IO orders of magnitude in some cases – Indexes are sorted (at least on the page boundary) so doing index range scan you typically get many index entries from the same page but the rows itself can be scattered across many pages requiring potentially a lot of IOs. On top of that if you just need access to couple of columns
index can be simply much smaller than the data which is one of the reason covering indexes help to speed up queries even if data is in memory. If MySQL is only reading index and not accessing rows you will see "using index" in EXPLAIN output.

These are the main "core" use for indexes. You can also see others like using index for group by but I think they can be pretty much looked as one of these 3 ways described.

About the Author

Peter Zaitsev

Peter managed the High Performance Group within MySQL until 2006, when he founded Percona. Peter has a Master's Degree in Computer Science and is an expert in database kernels, computer hardware, and application scaling.