久

杨

珍

精

悟尽归原 几近于道。

create by Yangcl

2018-03-15

1 Mysql基础篇

2 Mysql高级特性

2.1数据库索引与高性能查询

2.1.1 MySQL索引5大原则

**A：最左前缀匹配原则(重要的原则)**

mysql会一直向右匹配直到遇到范围查询（>、<、between、like）就停止匹配。比如a = 1 and b = 2 and c > 3 and d = 4，如果建立（a,b,c,d）顺序的索引，d是用不到索引的，如果建立(a,b,d,c)的索引则都可以用到，a,b,d的顺序可以任意调整。 一个查询中，如果第一个查询条件为索引，同时因为使用了>、<、between或like导致他失效了，那么索引整体失效。

**B：=和in可以乱序**

比如a = 1 and b = 2 and c = 3 建立（a,b,c）索引可以任意顺序，mysql的查询优化器会帮你优化成索引可以识别的形式。

**C：尽量选择区分度高的列作为索引**

区分度的公式是count(distinct column)/count(\*)。表示字段不重复的比例，比例越大我们扫描的记录数越少，唯一键的区分度是1，而一些状态、性别字段可能在大数据面前区分度就是0，那可能有人会问，这个比例有什么经验值吗？使用场景不同，这个值也很难确定，一般需要join的字段我们都要求是0.1以上，即平均1条扫描10条记录。

**D：索引列不能参与计算，保持列“干净”**

比如from\_unixtime(create\_time) = ’2014-05-29’就不能使用到索引，原因很简单，b+树中存的都是数据表中的字段值，但进行检索时，需要把所有元素都应用函数才能比较，显然成本太大。所以语句应该写成create\_time = unix\_timestamp(‘2014-05-29’);

**E：尽量的扩展索引，不要新建索引**

比如表中已经有a的索引，现在要加(a,b)的索引，那么只需要修改原来的索引即可。

2.1.2 索引使用的典型场景(使用索引优化查询)

A：匹配全值

对索引中所有列都指定具体值，即对索引中的所有列都有等值匹配的条件。

#设置组合索引（rental\_date,inventory\_id,customer\_id）为唯一索引。

EXPLAIN

SELECT \* FROM

rental

WHERE rental\_date = '2005-05-25 17:22:10'

AND inventory\_id = 373

AND customer\_id = 343 ;



B：匹配值的范围查询

对索引值进行范围查找，这种情况最常见。

#设置索引idx\_fk\_customer\_id(customer\_id)

EXPLAIN

SELECT

\*

FROM

rental

WHERE customer\_id >= 373

AND customer\_id < 400 ;



**C：匹配最左前缀**

仅仅使用索引中的最左边列进行查询。比如组合索引（col1,col2,col3）能够被col1，col1+col2，col1+col2+col3的等值查询利用到的。

#创建索引idx\_payment\_date(payment\_date,amount,last\_update);

EXPLAIN

SELECT \* FROM

payment

WHERE payment\_date = '2006-02-14 15:16:03'

AND last\_update = '2006-02-15 22:12:32' ;



从结果可以看出利用了索引，但又row为182行，所有只使用了部分索引。换一种方式，如下脚本，将不适用索引，请注意标红色的amount条件

EXPLAIN

SELECT \* FROM

payment

WHERE **amount** = 3.98

AND last\_update = '2006-02-15' ;



从结果看出，这次查询没有利用索引，进行了全表查找。因为我们所创建的索引中没有包含amount参数。

**D：所查询列全为索引**

当查询列都在索引字段中。即select中的列都在索引中。

#创建索引idx\_payment\_date(payment\_date,amount,last\_update);

EXPLAIN

SELECT

last\_update

FROM

payment

WHERE payment\_date = '2005-08-19 21:21:47'

AND amount = 4.99 ;



Extra内容显示为Using index,说明不需要通过索引回表，Using index就是平时说的覆盖索引扫描（即找到索引，就找到了要查询的结果，不用再回表查找了）。

**E：匹配列前缀**

仅仅使用索引的第一列，并且只包含索引第1列的开头部分进行查找。

#创建索引idx\_title\_desc\_part(title(10)，description(20));

EXPLAIN

SELECT

title

FROM

film\_text

WHERE title LIKE 'african%' ;



type=range，范围扫描，基于索引做范围扫描，为诸如BETWEEN，IN，>=，LIKE类操作提供支持。

为提高like的查询效率，请注意创建索引的代码中红色的部分，title(10)大意为匹配前10个字符。大表查询中极少使用LIKE这个关键字。

**F：索引部分等值匹配，部分范围匹配**

EXPLAIN

SELECT

inventory\_id

FROM

rental

WHERE rental\_date = '2006-02-14 15:16:03'

AND customer\_id >= 300

AND customer\_id <= 400 ;



type=ref，说明使用了索引。

**G：列名是索引，column\_name is null，使用索引**

EXPLAIN

SELECT

\*

FROM

payment

WHERE rental\_id IS NULL ;



2.1.3索引存在但不能使用索引的典型场景

**A：以%开头的like查询**

EXPLAIN

SELECT

\*

FROM

actor

WHERE last\_name LIKE '%NI%' ; **#百分号“%”只能在最右的情况索引才不会失效！**



**B：数据类型出现隐式转化，不会使用索引**

EXPLAIN

SELECT

\*

FROM

actor

WHERE last\_name = 1 ;



last\_name为varchar类型存储，但匹配值为int类型的数字，此时索引失效；正确如下：WHERE last\_name = '1' ;

**C：组合索引，不满足最左原则，不使用符合索引**

详见2.1.1 A中的说明。

**D：用索引比全表扫描还慢，则不要使用索引**

如查询以“S”开头的标题的电影，返回记录比例比较大，mysql预估索引扫描还不如全表扫描。

**E：用or分割条件，若or前后只要有一个查询条件的列没有索引，就都不会用索引**

EXPLAIN

SELECT

\*

FROM

payment

WHERE customer\_id = 203

OR amount = 3.96 ;



2.1.4 mysql explain结果中type的含义

|  |  |
| --- | --- |
| type值 | 含义 |
| system | CONST的特例，当表上只有一条元组匹配 |
| const | WHERE条件筛选后表上至多有一条元组匹配时，比如WHERE ID = 2 （ID是主键，值为2的要么有一条要么没有） |
| eq\_ref | 参与连接运算的表是内表（在代码实现的算法中，两表连接时作为循环中的内循环遍历的对象，这样的表称为内表）。  基于索引（连接字段上存在唯一索引或者主键索引，且操作符必须是“=”谓词，索引值不能为NULL）做扫描，使得对外表的一条元组，内表只有唯一一条元组与之对应。 |
| ref | 可以用于单表扫描或者连接。参与连接运算的表，是内表。基于索引（连接字段上的索引是非唯一索引，操作符必须是“=”谓词，连接字段值不可为NULL）做扫描，使得对外表的一条元组，内表可有若干条元组与之对应。 |
| ref\_or\_null | 类似REF，只是搜索条件包括：连接字段的值可以为NULL的情况，比如 where col = 2 or col is null |
| range | 范围扫描，基于索引做范围扫描，为诸如BETWEEN，IN，>=，LIKE类操作提供支持 |
| index\_scan | 索引做扫描，是基于索引在索引的叶子节点上找满足条件的数据（不需要访问数据文件） |
| all | 全表扫描或者范围扫描：**不使用索引**，顺序扫描，直接读取表上的数据（访问数据文件） |
| unique\_subquery | 在子查询中，基于唯一索引进行扫描，类似于EQ\_REF |
| index\_subquery | 在子查询中，基于除唯一索引之外的索引进行扫描 |
| index\_merge | 多重范围扫描。两表连接的每个表的连接字段上均有索引存在且索引有序，结果合并在一起。适用于作集合的并、交操作。 |
| ft | FULL TEXT，全文检索 |

你定义的索引在查询时是否生效取决于这里展示的type的值。

2.2数据库索引深层原理

http://blog.codinglabs.org/articles/index-condition-pushdown.html

Java篇

Html/Css/Javascript 篇

Servlet/J2EE篇

实用代码篇

* 1. Java代码段

1.1 十六进制和八进制

* 1. Javascript代码段
  2. Mysql代码段

代码样式

|  |  |  |
| --- | --- | --- |
| 方法名 | 返回值 | 功能描述 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| 方法名 | 功能描述 |
|  |  |
|  |  |