此文章已于 18:11:46 2017/12/25 发布到 HealerJean梦想博客

sql优化

类别 java面试

##### 1、查询条件减少使用函数，避免全表扫描

##### 2、减少不必要的表连接

##### 3、有些数据操作的业务逻辑可以放到应用层进行实现

##### 4、可以使用with as

##### 5、使用“临时表”暂存中间结果

##### 6、不要把SQL语句写得太复杂

##### 7、不能循环执行查询

##### 8、用 exists 代替 in

##### 9、表关联关系不要太纠结

##### 10、查询多用索引列取查，用charindex或者like[0-9]来代替%%

##### 11、inner关联的表可以先查出来，再去关联leftjoin的表

##### 12、可以进行表关联数据拆分，即先查出核心数据，再通过核心数据查其他数据，这样会快得多

##### 13、参考SQL执行顺序进行优化

##### 14、表关联时取别名，也能提高效率

##### 15、使用视图，给视图建立索引进行优化

##### 16、使用数据仓库的形式，建立单独的表存储数据，根据时间戳定期更新数据。将多表关联的数据集中抽取存入一张表中，查询时单表查询，提高了查询效率。

# 1、合理使用索引

<https://www.cnblogs.com/Cheney222/articles/5876382.html>

<http://blog.csdn.net/tsuliuchao/article/details/4877157>

## 1、索引问题

索引是数据库优化中最常用也是最重要的手段之一,通过索引通常可以帮助用户解决大多数的 SQL 性能问题。

## 2 MySQL 如何使用索引

　　索引用于快速找出在某个列中有一特定值的行。对相关列使用索引是提高 SELECT 操作性能的最佳途径。  
　　查询要使用索引最主要的条件是查询条件中需要使用索引关键字,如果是多列索引,那么只有查询条件使用了多列关键字最左边的前缀时,才可以使用索引,否则将不能使用索引。

#### **1.使用索引**

##### (1)对于创建的多列索引,只要查询的条件中用到了最左边的列,索引一般就会被使用，

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | #首先按 company\_id,moneys 的顺序创建一个复合索引,具体如下:  mysql> create index ind\_sales2\_companyid\_moneys on sales2(company\_id,moneys);  Query OK, 1000 rows affected (0.03 sec)  Records: 1000 Duplicates: 0 Warnings: 0    #然后按 company\_id 进行表查询,具体如下:  mysql> explain select \* from sales2 where company\_id = 2006\G;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  id: 1  select\_type: SIMPLE  table: sales2  type: ref  possible\_keys: ind\_sales2\_companyid\_moneys  208key: ind\_sales2\_companyid\_moneys  key\_len: 5  ref: const  rows: 1  Extra: Using where  1 row in set (0.00 sec)    #可以发现即便 where 条件中不是用的 company\_id 与 moneys 的组合条件,索引仍然能用到,这就是索引的前缀特性。#但是如果只按 moneys 条件查询表,那么索引就不会被用到,具体如下:  mysql> explain select \* from sales2 where moneys = 1\G;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  id: 1  select\_type: SIMPLE  table: sales2  type: ALL  possible\_keys: NULL  key: NULL  key\_len: NULL  ref: NULL  rows: 1000  Extra: Using where  1 row in set (0.00 sec) |

##### (2)对于使用 like 的查询,后面如果是常量并且只有%号不在第一个字符,索引才可能会被使用：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | mysql> explain select \* from company2 where name like '%3'\G;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  id: 1  select\_type: SIMPLE  table: company2  type: ALL  possible\_keys: NULL  key: NULL  key\_len: NULL  ref: NULL  rows: 1000  Extra: Using where  1 row in set (0.00 sec)      mysql> explain select \* from company2 where name like '3%'\G;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  id: 1  select\_type: SIMPLE  table: company2  type: range  209possible\_keys: ind\_company2\_name  key: ind\_company2\_name  key\_len: 11  ref: NULL  rows: 103  Extra: Using where  1 row in set (0.00 sec)  #可以发现第一个例子没有使用索引,而第二例子就能够使用索引,  #区别就在于“%”的位置不同,前者把“%”放到第一位就不能用到索引,而后者没有放到第一位就使用了索引。  #另外,如果如果 like 后面跟的是一个列的名字,那么索引也不会被使用。 |

##### (3)如果对大的文本进行搜索,使用全文索引而不用使用 like ‘%...%’。

##### (4)如果列名是索引,使用 column\_name is null 将使用索引。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | mysql> explain select \* from company2 where name is null\G;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  id: 1  select\_type: SIMPLE  table: company2  type: ref  possible\_keys: ind\_company2\_name  key: ind\_company2\_name  key\_len: 11  ref: const  rows: 1  Extra: Using where  1 row in set (0.00 sec) |

#### **2.存在索引但不使用索引**

##### (1)如果 MySQL 估计使用索引比全表扫描更慢,则不使用索引。例如，如果列key\_part1 均匀分布在 1 和 100 之间,下列查询中使用索引就不是很好:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM table\_name where key\_part1 > 1 and key\_part1 < 90; |

##### (2)如果使用 MEMORY/HEAP 表并且 where 条件中不使用“=”进行索引列,那么不会用到索引。heap 表只有在“=”的条件下才会使用索引。

##### (3)用 or 分割开的条件,如果 or 前的条件中的列有索引,而后面的列中没有索引,那么涉及到的索引都不会被用到,例如:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | mysql> show index from sales\G;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Table: sales  Non\_unique: 1  Key\_name: ind\_sales\_year  Seq\_in\_index: 1  Column\_name: year  210Collation: A  Cardinality: NULL  Sub\_part: NULL  Packed: NULL  Null:  Index\_type: BTREE  Comment:  1 row in set (0.00 sec) |

从上面可以发现只有 year 列上面有索引,来看如下的执行计划:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | mysql> explain select \* from sales where year = 2001 or country = 'China'\G;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  id: 1  select\_type: SIMPLE  table: sales  type: ALL  possible\_keys: ind\_sales\_year  key: NULL  key\_len: NULL  ref: NULL  rows: 12  Extra: Using where  1 row in set (0.00 sec) |

 可见虽然在 year 这个列上存在索引 ind\_sales\_year,但是这个 SQL 语句并没有用到这个索引,原因就是 or 中有一个条件中的列没有索引。