1. 多关联没有必要的表

这个问题很多情况下是导致performance问题的一个重要因素, 原因有以下

1. 对表结构不了解, 表关联途径不熟悉
2. 出现在报表或者list页面, 特别是报表. 写报表的时候为了方便, 将所有数据包含在一个SQL语句中, 但是大多数时候只取一部分列, 这个时候某些表其实是没有必要串到的.
3. 滥用DISTINCT

只要SQL编写得当, DISTINCT其实是比较需要用到的. 程序中经常可以看到类似的写法:

SELECT DISTINCT order.order\_no

FROM order.order, order.order\_items

WHERE order.order\_no = order\_items.order\_no

其实完全可以改为

SELECT order.order\_no

FROM order.order

WHERE EXISTS

(SELECT 1 FROM order.order\_items

WHERE order.order\_no = order\_items.order\_no)

查看执行计划可以看出二者的差别:





这个其实效果还不明显, 因为order\_no在两个表上都有索引, 而且串的表也不多.

1. 表连接顺序对排序的影响

这个是一个很有用的技巧, 对performance影响很大:

SELECT \*

FROM order.order, order.order\_items

WHERE order.order\_no = order\_items.order\_no

ORDER BY order\_items.order\_item\_id



SELECT \*

FROM order.order\_items STRAIGHT\_JOIN order.order

ON order.order\_no = order\_items.order\_no

ORDER BY order\_items.order\_item\_id



上下两个SQL的差别就在于是先访问order表还是先访问order\_items表.

如果不强制连接顺序的话, MYSQL有话引擎会优先访问比较小的表, 也就是order表, order表中的每一条记录再来访问一次order\_items表, 但是因为排序字段是在order\_items表中, 所以mysql只能在获取所有结果集以后再进行排序, 结果集太大在内存中存放不了, 就只能使用临时表, 而是用临时表是效率很差的一个操作.

但是如果强制了连接顺序, MYSQL的优化引擎check到前导表上有排序操作, 会在访问前导表的时候就进行排序, 然后直接根据这个排序访问第二个表. 而这个排序字段如果正好是前导表的索引的时候, 就可以直接使用索引进行快速排序了.

**优化的原则是排序字段尽量放在索引字段上, 并且排序操作的结果集应尽量小.**

1. 分页中记录数的获得

这个问题其实很明显, 但是修改起来比较麻烦.

在大部分list页面, 因为list中要出现某些字段, 所以要串到某些表, 但是串这些表对结果集记录数并没有影响. 所以如果在取得结果集总数的时候还是用与取数据一样的SQL语句, 就很容易有performance问题.

1. 函数的使用

因为MYSQL至于一种连接方式, 那就是nested loop, 也就是说在将a表与b表相关联的时候, 如果a表示前导表, 那么b表的访问次数就是a表的记录数.

很多时候如果只需要从b表取很有限的信息, 并且因为SCM系统中大部分时候都会有limit, 这个时候, 如果将从b表取数据改为函数获取, 在于limit进行组合的时候, 往往可以取得意想不到的效果.

1. MYSQL中limit的位置问题

我们目前使用的MYSQL版本在处理嵌套查询的时候有很严重的性能问题, 而我们现在的SCM程序中使用嵌套查询很多.

下面两个SQL性能的差别可以很容易看出问题:

Select \* from order.order limit 20

Select \* from(select \* from order.order) a limit 20;

尽量不要使用第二种, 如果程序处理实在有困难, 希望使用嵌套查询

也建议使用

Select \* from(select \* from order.order limit 20) a;

1. 索引

加索引绝大多数时候是很有效又方便的调优方式, 但是却很容易被滥用. 因为对表的增删改都涉及到对索引的操纵, 往往是这一个搜索变快了, 其他很多地方变慢了. 增加索引一定要慎重, 通常应该将索引作为调优的最后考虑方案.