SQL 性能：

https://www.zhihu.com/question/52273252

1. (a, b) IN ((…), (…), …)



254 -> 29 88.5%

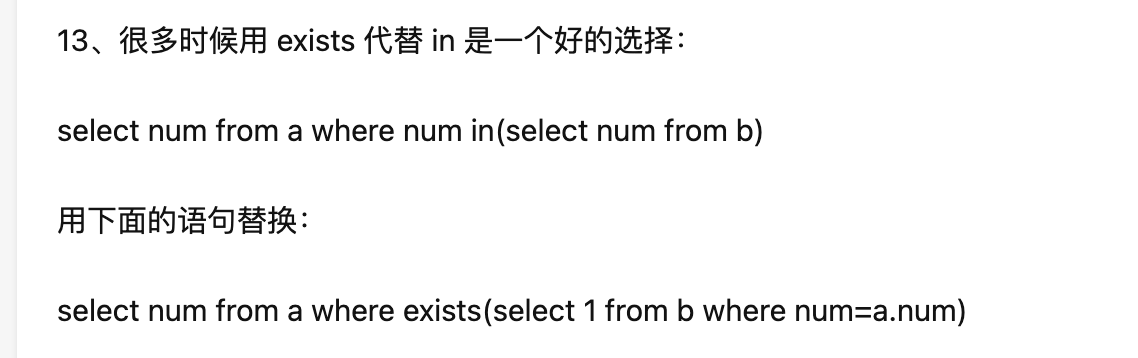
主要的原因是优化器无法识别决策到利用索引，因此通过force index 也可以实现

1. IN (…) 太多

分批查询/插入：pundimdaan caacheon aam [cond] paici

使用 join / exist / between





1. IN (select …)

在dao层分两次调用mapper查询 500 -> 50 90%

52.08%（1903ms -> 912ms）

1. 回表不够

新建索引：有哪些原则？索引是不是越多越好？

1. Pailoeng saangsing \_ip!fudaan 重复鉴权上缓存 5620 -> 2915 48.13%

**索引原则**

1、尽量选择区分度高的列作为索引，区分度的公式是count(distinct col)/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。

2、最左前缀匹配原则，非常重要的原则，索引字段使用 >, >=, =, <, <=, IF NULL和BETWEEN 将会使用索引，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。

3、=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。or 两边必须都建立索引，否则索引失效。

4、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’)。

5、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。

6、尽量避免在WHERE子句中使用in,not in <>,,is not null ，msyql函数操作符，这样数据库会进行全表扫描

7、关于LIKE 查询，使用 LIKE ‘%abc%’   
, ‘%abc’ 不能使用索引，使用 LIKE ‘abc%’ 将能够使用索引。

8、尽量避免在WHERE子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。

9、尽量避免where 进行 null值判断（where num is null）否则将引擎放弃使用索引

10、检查的行数过多，并且没有使用覆盖索引。

11、使用了多个索引，mysql一次只会采用一个索引。

12、where与order by使用了不同的索引，与11条类似

13、对索引列同时使用了ASC和DESC。 通过where语句将order by中索引列转为常量，则不会显示filesort。

14、where或者ORDER BY中索引列使用了表达式(+1，-1这种操作)，包括函数表达式。

15、where 与ORDER BY组合满足最左前缀，但where中使用了范围查询。

16、order by中加入了非索引列,且非索引列不在where中。

17、order by或者它与where组合没有满足索引最左前列。

18、当使用left join，使用右边的表字段排序。

## explain神器

**概要描述：**  
id:选择标识符  
select\_type:表示查询的类型。

SIMPLE：简单的select查询，查询中不包含子查询或者UNION

PRIMARY：查询中若包含任何复杂的子部分，最外层查询则被标记为

SUBQUERY：在SELECT或者WHERE列表中包含了子查询

DERIVED：在FROM列表中包含的子查询被标记为DERIVED（衍生）MySQL会递归执行这些子查询，把结果放在临时表里。

UNION：若第二个SELECT出现在UNION之后，则被标记为UNION;若UNION包含在FROM子句的子查询中，外层SELECT将被标记为：DERIVED

UNION RESULT：从UNION表获取结果的SELECT

table:输出结果集的表  
partitions:匹配的分区  
type:表示表的连接类型

ALL、index、range、 ref、eq\_ref、const、system、NULL（从左到右，性能从差到好），一般来说，得保证查询至少达到range级别，最好能达到ref。

possible\_keys:表示查询时，可能使用的索引  
key:表示实际使用的索引  
key\_len:索引字段的长度  
ref:列与索引的比较  
rows:扫描出的行数(估算的行数)  
filtered:按表条件过滤的行百分比  
Extra:执行情况的描述和说明

Using temporary：表示MySQL需要使用临时表来存储结果集，常见于排序和分组查询，常见 group by ; order by。必须优化。

Using filesort：当Query中包含 order by 操作，而且无法利用索引完成的排序操作称为“文件排序”。一定要优化

Using join buffer：改值强调了在获取连接条件时没有使用索引，并且需要连接缓冲区来存储中间结果。如果出现了这个值，那应该注意，根据查询的具体情况可能需要添加索引来改进能。

Impossible where：这个值强调了where语句会导致没有符合条件的行（通过收集统计信息不可能存在结果）。

Select tables optimized away：这个值意味着仅通过使用索引，优化器可能仅从聚合函数结果中返回一行

No tables used：Query语句中使用from dual 或不含任何from子句

Using where 查找

Using index 表示相应的select 操作使用了覆盖索引，避免访问了表的数据行，效率不错。如果只有 Using index，说明他没有查询到数据表，只用索引表就完成了这个查询，这个叫覆盖索引。 如果出现Using where ，表明索引被用来执行索引键值的查找。