


一图解析MySQL执行查询全流程



华为云开“H”
已认证帐号

5 人赞同了该文章

摘要：当我们希望MySQL能够以更高的性能运行查询时，最好的办法就是弄清楚MySQL是如何优化和执行查询的。

本文分享自华为云社区《MySQL执行查询全流程解析》，作者：61X 8HT。

MySQL执行查询的过程



客户端先发送查询语句给服务器

服务器检查缓存，如果存在则返回

进行 kh 解析，生成解析树，再预处理，生成第二个解析树，最后再经过优化器，生成真正的执行计划

根据执行计划，调用存储引擎的 $2E$ 来执行查询

将结果返回给客户端。

一、客户端到服务端之间的原理

客户端和服务端之间是半双工的，即一个通道内只能一个在发一个接收，不能同时互相发互相接收

客户端只会发送一个数据包给服务端，并不会在应用层拆成 6 个数据包去发（ $h\ \acute{u}\ \acute{e}\ T\ X\ \acute{u}\ H\ \backslash \acute{a}x\ m$ 可以设置数据包最大长），这关系到 kh 语句不能太长。

服务端返回给客户端可以有多个数据包，但是客户端必须完整接收，不能接到一半停掉连接或用连接去做其他事（ $h\ A$ 界面可以操作，不同的线程）

例如 $\acute{A}\acute{t}$ ，如果没设置 $NX\acute{h}\ \acute{U}\ g\ \acute{a}\ X$ ，那么都是一次性把结果读进内存。当你使用 $H\ X\acute{o}\ \acute{e}\ g\ X\acute{r}$ 的时候，其实已经全部进来了，而不是一条条从服务端获取。 $A\ A\ A\ A$ 使用 $NX\acute{h}\ \acute{U}\ g\ \acute{a}\ X$ 边读边处理的坏处：服务端占用的资源时间变久了。

查询 $h\ \acute{u}\ kh$ 服务此时的状态

使用 $\acute{C}\acute{u}\acute{r}\ i\ L\ \acute{Q}\acute{u}\acute{a}\acute{r}\ \acute{z}\ \acute{C}\acute{u}\acute{r}$ 命令可以查看 $h\ \acute{u}\ kh$ 服务端某些线程的状态

$g\ \acute{E}\ X\ H$ 正在等待客户端发送新的请求

$\acute{e}\ \acute{o}\ X\ \acute{h}$ 正在执行查询，或者发结果发给客户端

$\acute{t}\ \backslash \acute{a}\ X$ 正在等待表锁（注意表锁是服务器层的，而行锁是存储引擎层的，行锁时状态为 $h\acute{o}\ X\ \acute{h}$ ）

$2\ \acute{e}\ \acute{E}\ V\ \acute{a}\ n\ j\ \acute{e}\ l\ k\ m\ \acute{r}\ \acute{a}\ \acute{r}\ k$ 正在生成查询的计划或者收集统计信息

$\backslash\ \acute{r}\ H\ \acute{a}\ n\ j\ \acute{r}\ \acute{h}\ H\ m\ 6\ \acute{E}$ 临时表操作，一般是正在做 $n\ H\acute{o}\ H\ 6\ \acute{u}$ 等操作

$k\ \acute{r}\ H\ \acute{a}\ n\ j\ X\ \acute{o}\ \acute{e}\ n$ 正在对结果集做排序

$k\ X\ \acute{e}\ l\ \acute{a}\ n\ j\ l\ \acute{m}$ 正在服务器线程之间传数据

二、查询缓存

缓存的查询在 kh 解析之前进行。

缓存的查找通过一个对大小写敏感的哈希表实现，即直接比对 kh 字符串。

因此只要有一个字节不同，都不会匹配中。（毕竟还没开始解析，大小写什么的他也不知道要不要区分）

第 6 章中有更详细的查询缓存。

三、查询优化处理

5 语法解析器和预处理

这里就是把SQL做解析， 变成一个解析树。解析时会做SQL语法规则验证。

语法解析器：检查关键字错误、关键字顺序、引号匹配

预处理：和元数据关联校验， 检查数据表和列是否存在， 解析名字和别名。

权限校验

6 查询优化器(重点)

MySQL可能会生成多种计划， 他会分别计算一个预测成本值， 然后选一个成本最小的计划

计算信息来自于 表的页面个数、索引分布、长度、个数、数据行长度

因为多种原因，可能不会选择到最优的计划，有偏差

静态优化和动态优化的区别：

静态优化类似编译期优化，只和语句结构有关，和具体值无关

动态优化是在运行中去优化的，需要依赖索引行数、统计值取值，执行次数可能比静态优化要多。

MySQL的优化类型

关联表（JOIN）的顺序可能会变

”JOIN-A”可能会变成内连接

优化条件表达式，例如 $a \cdot b \cdot c$ 被简化成 $a \cdot b$

优化 $a \cdot b \cdot c$ ，如果是 $a \cdot b$ (索引)，那么直接拿 $a \cdot b$ 树的第一条或者最后一条即可。

当发现某个查询或者表达式的结果是可以提前计算出来的时候，就会优化成常数

索引覆盖，如果只要返回索引列，就不会走到最底层去。

子查询优化

提前终止查询（例如 $a \cdot b$ ）

等值传播： $a \cdot b$ 中可能把左表的 a 拿给右表一起用

$a \cdot b$ 这个条件，并不是简单遍历判断，会先排序，然后用二分去判断是否存在。

7 数据和索引的统计信息

统计信息是存储引擎去计算的，不同的存储引擎有不同的统计信息

服务器层生成查询计划时，会向存储引擎获取这些信息。

8 对关联查询的执行

JOIN查询的本质其实是读取临时表做关联

例如 $a \cdot b \cdot c$ 这个条件，如果是 $a \cdot b$ (索引)，那么直接拿 $a \cdot b$ 树的第一条或者最后一条即可。

遍历 的每一行（此时 表本质上是 $k \times \bar{X} \bar{L} \bar{m} \bar{N} \bar{J} \bar{h}$ $\bar{T} \bar{U} \bar{X} \bar{J}$ $\bar{h} \bar{U} \bar{i} \bar{u}$ ）

在那行中 的 \bar{a} 被定下来，那么就会去获取一个临时表，临时表为（ $k \times \bar{X} \bar{L} \bar{m} \bar{N} \bar{J} \bar{h}$ $\bar{6} \bar{T} \bar{U} \bar{X} \bar{J}$ $\bar{h} \bar{U} \bar{i} \bar{u}$ \bar{a} ）

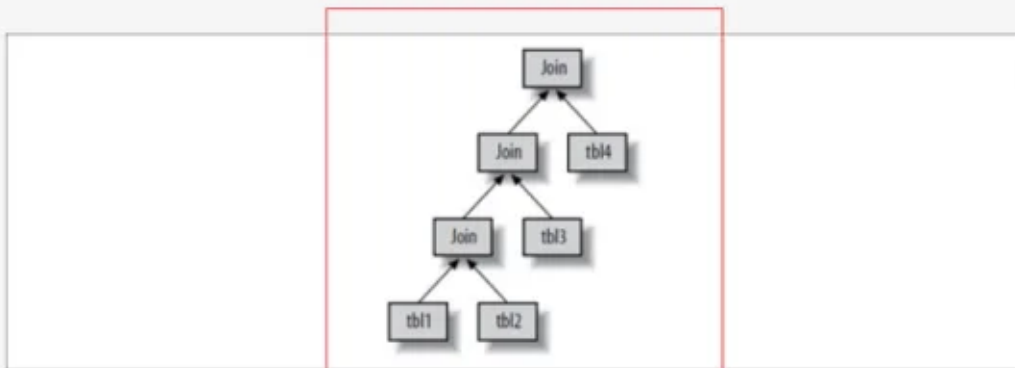
接着用这个临时表和 那一行拼接，输出多行。

然后再用这里的结果作为临时表，给更上层的关联去用（嵌套查询的含义）。

如果是 $\bar{X} \bar{N} \bar{L} \bar{A} \bar{h}$ ，则就是临时表如果为空，则给 那一行拼接一个 $\bar{e} \bar{o} \bar{E}$

+ \bar{h} 执行计划中的 $\bar{A} \bar{h}$ 树

在计算机科学中，这被称为一颗平衡树。但是，这并不是MySQL执行查询的方式。正如我们前面章节介绍的，MySQL总是从一个表开始一直嵌套循环、回溯完成所有表关联。所以，MySQL的执行计划总是如图6-4所示，是一棵左测深度优先的树。



- \bar{h} 关联查询优化器

$\bar{A} \bar{h}$ 实际执行的顺序会关系到性能

例如 $\bar{a} \bar{q}$ 三个表关联，可能先让 \bar{a} 和 $\bar{6}$ 关联得到的临时表里的记录只有 5^0 条，而如果让 \bar{a} 和 \bar{L} 先关联，会有 $5^0 \ 0 \ 0 \ 0$ 条，那么后面的效率就会截然不同

$\bar{t} \ \bar{S} \ \bar{E} \ \bar{t} \ \bar{2} \ \bar{A} \bar{h} \ \bar{t} \ \bar{S} \ \bar{K} \bar{t} \ \bar{O} \bar{k} \bar{t} \bar{k}$ 可以展示关联的顺序

$\bar{g} \bar{K} \bar{E} \bar{2} \bar{A} \bar{h} \bar{S} \bar{K} \bar{U} \bar{h}$ $\bar{A} \bar{h}$ 可以手动指定关联顺序

$\bar{h} \ \bar{u} \bar{k} \bar{h}$ 自己会评估搜索一个最优的顺序，但如果 $\bar{A} \bar{h}$ 表太多，则无法搜完所有结果（ $\bar{h} \bar{e} \bar{L}$ ，那时候就会采用贪心。是否使用贪心算法的边界值可以根据 $\bar{H} \bar{h} \bar{h} \ \bar{A} \bar{X} \bar{h} \bar{k} \bar{X} \ \bar{h} \bar{h} \bar{U} \bar{U} \ \bar{X} \bar{h} \bar{h}$ 去指定。

（ \bar{h} 排序优化

如果排序的量小，就用内存快速排序；如果排序的量，就用文件排序

$\bar{h} \ \bar{u} \bar{k} \bar{h}$ 有 6 种取排序数据的方式：

两次传输排序：先取要排序的字段加行序号，按照字段排序好之后，再根据行索引一条条取读
优点：排序时占用内存小。

缺点：排序之后读的过程会很慢，根据行序号取读不是很方便

单次传输排序：直接把行读出来（行里只有需要用的列，不一定是整行），然后排序

优点：把全部行读出来相当于顺序读，读取速度快

缺点：可能会很大导致需要文件排序

关联查询”的执行的注意事项

如果”的列 都来自关联的 第一张表，则直接第一张表读的时候就排序了。

除此之外！！都是全部读完，再排序！就算用了索引也是全部读完排序后，再读的！

四、查询执行计划

执行计划是一个数据结构

五、返回结果给客户端

用行封装包并逐步传送，而不是全部准备好再发送。