1. 基于成本的优化
2. 概述

所谓查询语句的执行成本由两个方面组成，一个是I/O成本，经常使用的InnoDB和MyISAM存储引擎都是将数据和索引存储到磁盘上，当查询表中的记录时，需要先把数据或者索引加载到内存中，然后再进行操作，从磁盘到内存的时间成本为I/O成本；还有一个是CPU成本，读取记录以及检测记录是否满足对应的搜索条件等操作的时间成本为CPU成本。

InnoDB读取一个页面花费的成本默认是1.0，读取一个记录以及检测一条记录是否符合搜索条件的成本默认是0.2，这些数字称为成本常数。

1. 优化步骤

MySQL的优化器会找出所有可以用来执行该语句的方案，在对比这些方案后找出成本最低的方案，即执行计划，之后才会调用存储引擎提供的接口执行查询。

1. 根据搜索条件，找出所有可能使用的索引

根据所有的搜索条件，来判断是否可以生成合适的扫描区间，进而得到可能使用的索引。

1. 计算全表扫描的代价

对于InnoDB来说，全表扫描就是将聚簇索引中的记录都一次与给定的搜索条件进行比较，并将符合搜索条件的记录加入到结果集中。所以需要将聚簇索引对应的页面加载到内存中，然后再检测记录是否符合搜索条件。查询成本=I/O成本+CPU成本，因此计算全表扫描的代价时需要两个信息：聚簇索引占用的页面数与该表中的记录数。

通过查询表的统计信息，得到表中的记录数与表占用的存储空间字节数。

SHOW TABLE STATUS LILKE ‘表名’;

Rows表示表中的记录数，Data\_length表示表占用的字节数，对于MyISAM来说是数据文件的大小，对于InnoDB来说是聚簇索引的大小，可以除以16KB得出聚簇索引的页面数量。之后，计算I/O成本：

I/O成本 = 聚簇索引页面数量 x 1.0 + 1.1

1.0表示磁盘加载到内存的成本常数，1.1是微调值；

CPU成本 = 记录数 x 0.2 + 1.0

0.2是读取记录以及检测记录的成本常数，1.0 是微调值

1. 计算使用不同索引执行查询的代价

分析单独使用不同的索引执行查询的代价，最后还要分析是否能够使用索引合并。一般使用的都是二级索引+回表方式执行的查询，在计算这种查询的成本时依赖两个数据：扫描区间的数量与需要回表的记录数。无论某个扫描区间的二级索引占用了几个页面，查询优化器粗暴地认为I/O成本与读取一个页面的成本相同。

根据回表的记录的主键值到聚簇索引中执行回表操作，MySQL中评估回表成本时认为每次回表相当于访问一个页面，即有多少条记录就有多少次页面I/O。回表得到完整的用户记录，再检测其他搜索条件是否成立即可，检测也是用读取记录与检测记录的成本计算。

最后对不同索引查询得到的二级索引记录是否按照主键排序，来决定是否索引合并。

1. 对比各种执行方案的代价，找出成本最低的方案

对比全表，不同索引的查询代价，使用最低成本的即可。

1. 连接查询的成本
2. 条件过滤

连接查询的查询成本由两部分组成：单词查询驱动表的成本以及多次查询被驱动表的成本。将查询驱动表后得到的记录条数称为驱动表的扇出。在两种情况下计算驱动表扇出值时需要经过猜测，一种是使用全表扫描的方式执行单表查询，另一种是使用索引单表查询，猜测满则搜索条件的记录数。这种猜测过程称为条件过滤。

1. 两表连接查询成本分析

连接查询的成本 = 单次访问驱动表成本 + 驱动表扇出值 x 单次访问被驱动表成本

对于左连接和右连接，驱动表是固定的，所以只需要为驱动表和被驱动表选择成本最低的访问方法，即可得到最优的查询方案。而对于内连接，驱动表与被驱动表的顺序可以互换，因此需要考虑最优的表连接顺序，然后再选择访问方法。这样的话，就需要为每一种连接顺序进行成本计算分析了。

可以看到，连接查询的优化目标就是尽量减少驱动表的扇出，以及访问被驱动表的成本要尽量的低。实际使用各种，降低访问被驱动表的成本最有效，如在被驱动表的连接列上建立索引，这样就可以使用ref访问方法了，最好情况下，使用主键或者唯一二级索引列，使用const访问方法。

1. 多表连接查询成本分析

对于n个表的连接查询，确实有n!中连接顺序，也确实都要计算一遍成本，MySQL中有很多优化这种因计算不同连接顺序下的查询成本而带来的性能损耗的方法。

1. 提前结束某种连接顺序的成本计算

MySQL在计算各种连接顺序的成本之前，维护一个全局变量，表示当前最小的连接查询成本，只要某种连接顺序的成本超过了它，就continue退出到下一个连接顺序了。

1. 系统变量optimizer\_search\_depth

为了防止无穷尽地分析各种顺序，设置了该系统变量，相当于一个上限值，如果连接表的个数小于该值，就穷举分析每一种连接顺序，否则只穷举分析与该值相同的表。