1. 查询优化器基于规则的优化
2. 概述

查询优化器针对查询语句根据相应的规则进行优化，转换为某种可以高效执行的形式，这个过程称为查询重写。

1. 条件化简
2. 移除不必要的括号

将表达式中多余的括号去掉。

1. 常量传递

若某个列和某个常量已经进行了等值匹配，当使用AND操作符将这个表达式和其他涉及该列的表达式连接起来的时候，可以将该列转换为常量。

如：

a = 5 AND b > a

a = 5 AND b > 5

1. 移除已知的条件

对于明显为TRUE或者FALSE的表达式，优化器会将它们移除。

1. 表达式计算

在查询执行之前，如果表达式中只包含常量的话，它的值会被先计算出来。

1. 常量表检测

对于记录数不大于1的表或者使用主键，唯一二级索引列等值匹配的表，由于这两种表查询花费时间很少，称为常量表。查询优化器在分析一个查询语句时，首先执行常量表检测，看看有没有常量表，然后将查询中涉及该表的条件全部替换为常数，再分析其他表的查询成本。

1. 外连接消除

为什么要消除外连接呢？因为内连接能够通过优化器评估表的不同连接顺序的成本，选出成本最低的连接顺序来执行查询。

知道了原因后，来看看怎么消除。凡是不符合WHERE子句条件的记录都不会参与连接，只要我们在WHERE子句的搜索条件中指定被驱动表的列不为NULL（这种条件称为空值拒绝），那么那些在外连接被驱动表中匹配不到ON子句条件的驱动表记录就不会出现在最终的结果集中，此时的外连接与内连接是一样的，也就达到了消除外连接的目的。

1. 子查询优化
2. 子查询
3. 概述

子查询就是在查询语句中嵌套一个查询，不论是在FROM子句，WHERE子句，ON子句等等中都可以嵌套子查询，外面的查询称为外层查询。放到FROM子句中的子查询称为派生表，因为相当于查询一个表。

1. 按照返回的结果集分类
2. 标量子查询

只返回一个单一值的子查询。

1. 行子查询

返回一条记录的子查询，该记录需要包含多个列。

1. 列子查询

查询出一个列的数据，需要包含多条记录。

1. 表子查询

子查询的结果既包含多条记录，也包含多个列。

1. 按照与外层查询的关系分类
2. 不相关子查询

子查询单独运行出结果，不依赖外层查询的值。

1. 相关子查询

子查询的执行需要依赖于外层查询的值。

1. 子查询在布尔表达式中的使用
2. 与比较操作符一起使用
3. IN/NOT IN

操作数 [NOT] IN (子查询)

是否（不）存在于查询结果集组成的集合中。

1. ANY/SOME

操作数 比较操作符 ANY/SOME(子查询)

只要在子查询的结果集中存在一个值，某个指定的操作数与该值通过比较操作符进行比较时结果为TRUE，则整个表达式为TRUE。

1. ALL

操作数 比较操作符 ALL(子查询)

只有某个指定的操作数与子查询的结果集中所有的值通过比较操作符进行比较时结果为TRUE，整个表达式才会为TRUE。

1. EXISTS

[NOT] EXISTS (子查询)

只需要判断子查询的结果集中是否有记录，而不在乎记录具体是什么。

1. 子查询注意事项
2. 子查询必须用小括号
3. SELECT子句中的子查询必须是标量子查询
4. 不允许在一条语句中增删改某个表的记录时，同时对该表子查询
5. 子查询在MySQL中如何执行的
6. 标量子查询，行子查询
7. 不相关子查询

单独执行子查询，然后将子查询得到的结果当做外层查询的参数，再执行外层查询，即多个单表查询。

1. 相关子查询

先从外层查询中获取一条记录，然后从这条记录中找出子查询涉及的值执行子查询，最后根据子查询的查询结果来检测外层查询中的条件是否成立，如果成立就把外层查询的那条记录加入到结果集中，否则丢弃。

1. IN子查询优化
2. 物化表

对于不相关的IN子查询，如果子查询的结果集太大可能会导致性能问题，如内存放不下，无法有效使用索引。因此不直接将结果集当做外层查询的参数，而是将结果集写入一个临时表中，对写入的记录进行去重以节省空间。临时表是基于内存的使用MEMORY存储引擎的，还会为该表建立哈希索引，如果结果集特别大，会将临时表转换为基于磁盘的存储引擎来保存结果集，索引也会变为B+树索引。这种创建临时表的过程称为物化，临时表称为物化表。

既然生成了新的表，就可以将该物化表与其他的表进行内连接操作了，也就可以通过计算成本来找出成本最低的执行方式了。

1. 半连接

物化表有创建临时表的成本，MySQL提出了半连接，直接将子查询转换为连接。将s1表和s2表进行半连接的意思就是对于s1表中的记录来说，只关心在s2表中是否存在与之匹配的记录，而不关心具体有多少条记录与之匹配，最终的结果集中只保留s1表的记录。

共有5中半连接的执行策略，且半连接只是MySQL内部的优化策略， 不是向用户开放的接口。

1. [NOT] EXISTS子查询

如果是不相关子查询，先执行子查询，得出结果是TURE还是FALSE后，重写原先的查询语句。如果是相关子查询，只能按照之前标量子查询中相关子查询的方式查询，捕获EXISTS子查询可以使用索引加快速度。

1. 派生表的优化

放在FROM子句中的子查询就是派生表，MySQL提供了两种执行策略。一种是将派生表物化，写入到一个临时表中，将这个物化表当做普通表一样来参与查询，MySQL中还有一种延迟物化的策略，在查询时真正用到派生表时才会物化派生表。还有一种就是将派生表与外层查询合并，写成没有派生表的形式，将派生表提取到FROM子句中，搜索条件合并到WHERE子句中。

MySQL优先尝试派生表和外层查询的合并，如果不行，就物化派生表，执行查询。