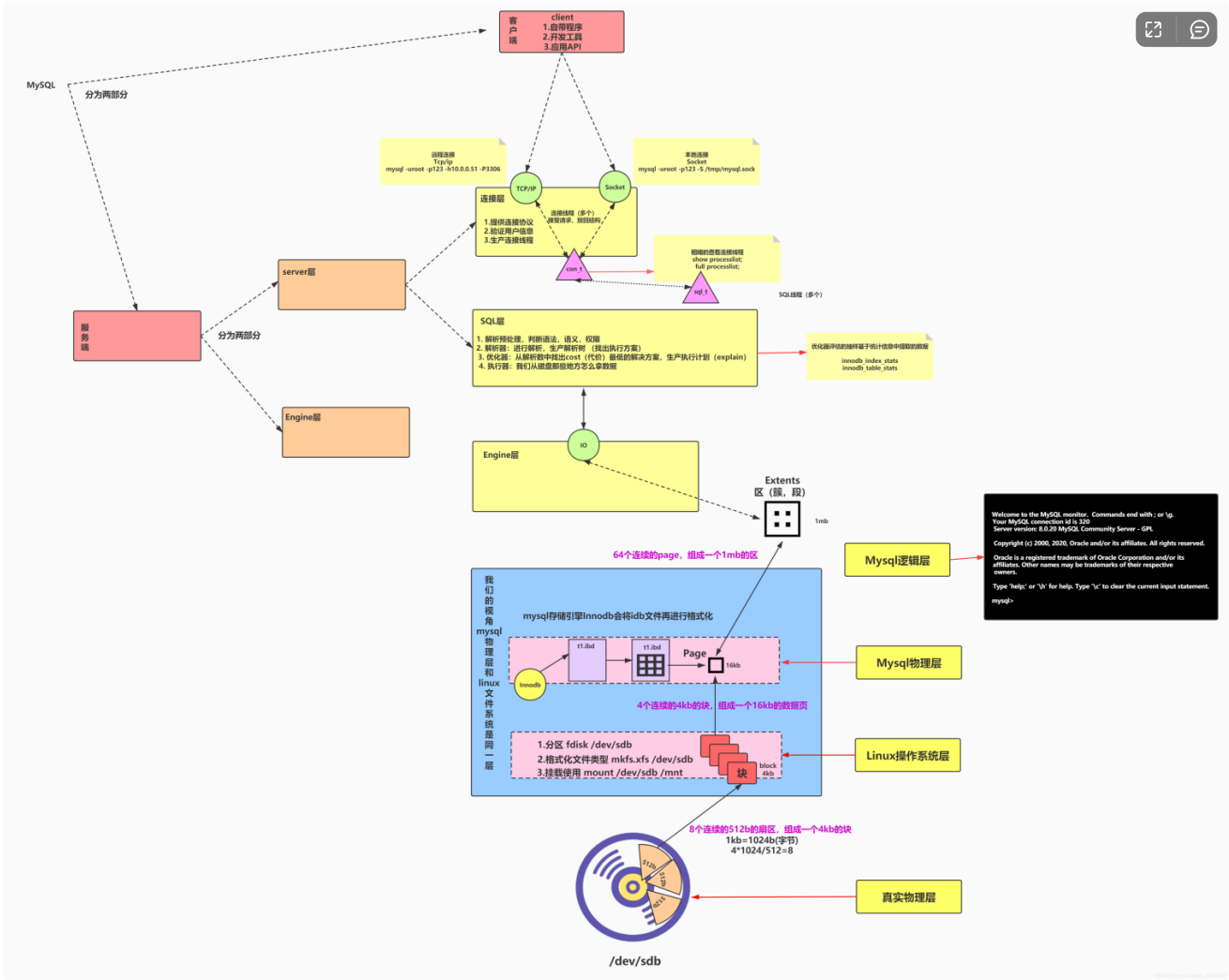
16.MySQL-执行器和优化器算法 √

1 执行计划（explian）

1.1 执行计划与优化器的关系

> 优化器：从解析数中找出cost（代价）最低的解决方案，生产执行计划（explain）



1.2 如何获取一条sql语句的执行计划呢?

有两种方式可以获取，只是获取当前sql语句的执行计划而不是真的运行当前sql语句。
所以可以在生产环境中提前检测一下使用sql语句的执行计划是否合理

```bash
desc select * from t100w where k1='aa';
explain  select * from t100w where k1='aa';
```

1.3 执行计划各列的作用

```
mysql> explain  select * from t100w where k1='aa';
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+---
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Ex
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+---
|  1 | SIMPLE      | t100w | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 997335 |    10.00 | Us
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+---
1 row in set, 1 warning (0.00 sec)
```

id	表格查询的顺序编号。	降序查看，id相同的从上到下查查看。 id可以为null，当table为(union ,m,n)类型的时候，id为null，这个时候，id的顺序为 m跟n的后面。
select_type	查询的方式	下文详细说明。
table	表格名称	表名，别名，(union m,n) 。
partitions	分区名称	查询使用到表分区的分区名。
type	表连接的类型	下文详细说明。
possible_keys	可能使用到的索引	这里的索引只是可能会有到，实际不一定会用到。
key	使用到的索引	实际使用的索引。
key_len	使用到索引的长度	比如多列索引，只用到最左的一列，那么使用到索引的长度则为该列的长度，故该值不一定等于 key 列索引的长度。
ref	谓词的关联信息	当 join type 为 const、eq_ref 或者 ref 时，谓词的关联信息。 可能为：null（非 const \ eq_ref \ ref join type 时）、const（常量）、关联的谓词列名。
rows	扫描的行数	该表格扫描到的行数。这里注意在mysql里边是嵌套链接，所以，需要把所有rows相乘就会得到查询数据行关联的次数
filtered	实际显示行数占扫描rows的比例	实际显示的行数 = rows * filtered / 100
extra	特性使用	

①id

②select_type

③table♥

sql语句操作的表，主要用在多表查询时，排查sql对应那张表出现了问题。

④partitions

⑤type♥ 查找的类型

type的输出结果分析

1>all 全表扫描

可能会产生场景的列子

1.没有加索引或者没有加where条件查询的sql语句

```
mysql> desc select * from t100w;
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------+
|  1 | SIMPLE      | t100w | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 997335 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

2.查询没有加索引列的sql语句

```
mysql>  desc t100w;
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
| Field | Type      | Null | Key | Default           | Extra                                         |
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
| id    | int       | YES  |     | NULL              |                                               |
| num   | int       | YES  |     | NULL              |                                               |
| k1    | char(2)   | YES  |     | NULL              |                                               |
| k2    | char(4)   | YES  |     | NULL              |                                               |
| dt    | timestamp | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
5 rows in set (0.00 sec)

mysql> desc select * from t100w where id=10;
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | t100w | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 997335 |    10.00 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```


3.模糊匹配 '%a%'

```
mysql> desc t100w;
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
| Field | Type      | Null | Key | Default           | Extra                                         |
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
| id    | int       | NO   | PRI | NULL              | auto_increment                                |
| num   | int       | YES  |     | NULL              |                                               |
| k1    | char(2)   | YES  | MUL | NULL              |                                               |
| k2    | char(4)   | YES  |     | NULL              |                                               |
| dt    | timestamp | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
5 rows in set (0.00 sec)

mysql> desc select * from t100w where k1 like '%a%';
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | t100w | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 997632 |    11.11 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```


4.不等于情况下！ =

```
mysql> desc t100w;
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
| Field | Type      | Null | Key | Default           | Extra                                         |
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
| id    | int       | NO   | PRI | NULL              | auto_increment                                |
| num   | int       | YES  |     | NULL              |                                               |
| k1    | char(2)   | YES  | MUL | NULL              |                                               |
| k2    | char(4)   | YES  |     | NULL              |                                               |
| dt    | timestamp | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |
+-------+-----------+------+-----+-------------------+-----------------------------------------------+
5 rows in set (0.00 sec)

mysql> desc select * from t100w where k1 !='aa';
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | t100w | NULL       | ALL  | i_k1          | NULL | NULL    | NULL | 997632 |    84.02 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```


5.不等于情况下<>

```
mysql> desc select * from t100w where k1<>'a';
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | t100w | NULL       | ALL  | i_k1          | NULL | NULL    | NULL | 997632 |    81.74 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

6.在联合索引中不走最左列情况下

```
mysql> show  index from t100w;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+-------
----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Express
ion |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+-------
----+
| t100w |          0 | PRIMARY  |            1 | id          | A         |      997632 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL
    |
| t100w |          1 | i_k1     |            1 | k1          | A         |        1397 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL
    |
| t100w |          1 | i_n_k1   |            1 | num         | A         |      576352 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL
    |
| t100w |          1 | i_n_k1   |            2 | k1          | A         |      997632 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL
    |
| t100w |          1 | i_n_k2   |            1 | num         | A         |      571255 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL
    |
| t100w |          1 | i_n_k2   |            2 | k2          | A         |      997550 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL
    |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+-------
----+
6 rows in set (0.00 sec)

mysql> desc select * from t100w where k2='0189';
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | t100w | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 997632 |    10.00 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+--------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```


2>index 全索引扫描

如果是主键列的全索引扫描也就等同于全表扫描

下图是name列和k2列的一个联合索引的全索引扫描

```
mysql> mysql> desc select num,k2 from t100w;
+----+-------------+-------+------------+-------+---------------+--------+---------+------+--------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key    | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+--------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | t100w | NULL       | index | NULL          | i_n_k2 | 22      | NULL | 997632 |   100.00 | Using index |
+----+-------------+-------+------------+-------+---------------+--------+---------+------+--------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

3>range 索引范围扫描♥

sql语句中用到（>,<,>=,<=,like,in,or）

```
mysql> +mysql> desc select * from t100w where id<10;
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | t100w | NULL       | range | PRIMARY       | PRIMARY | 4       | NULL |    9 |   100.00 | Using where |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

4>ref 辅助索引的等值查询♥

在range使用in与ref 对比实验

**我们在使用range的索引范围查询中如果用到in，查询两个值不是顺序io就利用不到b+tree底层的双向指针。

这里选用的方案按照真实的数据进行选择，进行测压。（如果基数大的话从range改写为ref效果会更加明显）**

5>eq_ref 多表连接时,非驱动表的连接条件是主键或唯一键

6>const(system)主键或唯一键的等值查询.

⑥possible_keys♥可能会用的索引

⑦key♥最终选择的索引

⑧key_len♥索引覆盖长度.联合索引有意义

key_len的算法

1.首先确定字符集

```
utf8 ,字符串列(char(N)) , N*3
utf8mb4 ,字符串列(char(10)) , N*4
gbk N*2
```

2.非空

```
如果设置not null 忽略不需要考虑。
如果没有设置not null 需要+1
```

3.判断什么数据类型

```
数字
tinyint  1
int      4
字符串
char(10)
varchar(10)   +2
时间
timestamp  4
datetime   8
```

⑩rows估算值,要扫描的行数

⑾filtered

⑿Extra 额外信息

filesort:查询中出现了额外的排序（order by ,group by)要进行优化

1.4 联合索引应用细节

```
idx(a,b,c)
a. 完全覆盖
where a= and b= and c=
where  b= and c=  and   a=
mysql> desc select * from student where stuname='aa' and stuage  in (10,20)  and gender='M';
where a= and b= order  c=
where  b= and c=  and   a=
where a= and b= and c>
b. 部分覆盖
mysql> desc select * from student where stuname='aa' and stuage>10 and gender='M';

where a= and b=
where a=
where a= and c=

c. 不覆盖
b= and c=
b=
c=

d. 最左原则
1. 选择基数大的列作为最左列
2. 查询条件中必须包含最左列条件.
```

2.扩展内容

2.1 8.0在索引中的新特性

```
help alter table 查看修改表的选项
```

```
index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | {VISIBLE | INVISIBLE}
```

a.不可见索引

删除修改索引时设置，默认是可见（visible）。我们不适用那个索引可以修改为不可见模式(invisible)

Bash | Copy

```
alter table t1 alter index idx_name visible;
alter table t1 alter index idx_name invisible;
```

b.倒序索引

在8.0版本之前索引默认的排序都是从小到大排序
业务需求 我们有a b c 三列，要求先过滤出a列信息，在排序b列 ， c列要求排倒序
我们之前的做法是做abc的联合索引 `index(a,b,c)`
可是如果这种要求我们经常用到，我们在创建索引的时候可以对c列创建出倒序索引

Bash | Copy

```
index(a,b,c desc)
```

2.2 索引自优化 ♥(5.6版本之后)

Bash | Copy

```
AHI（adaptive hash index）：自适应的hash索引（索引的索引）
a.自适应：自动去评估我们内存buffer pool中的热点索引页，自动生成hash索引表，把这些热点索引页组成到AHI的缓冲区中，
b.当用户想去使用热点索引页是先去扫描AHI的缓冲区里的hash列表，能够快速锁定热点索引页所在的内存的位置
```

```
change buffer ：
当我们从索引页中间进行数据的插入，可能会造成页分裂（重组所有数据结构，时间长）或者修改指针的情况。
影响：
聚簇索引如果是自增列的情况下，插入数据就是追加的方式，如果数据页（叶子节点）有空间直接添加到该数据页（叶子节点），如果没有会追加新的数
辅助索引是根据聚簇索引中的列进行创建的，如果在辅助索引的叶子节点中间进行数据插入默认数据页是有余留的空间的，那么只是枝节点指针的变化。
```

change buffer 针对辅助索引的变更

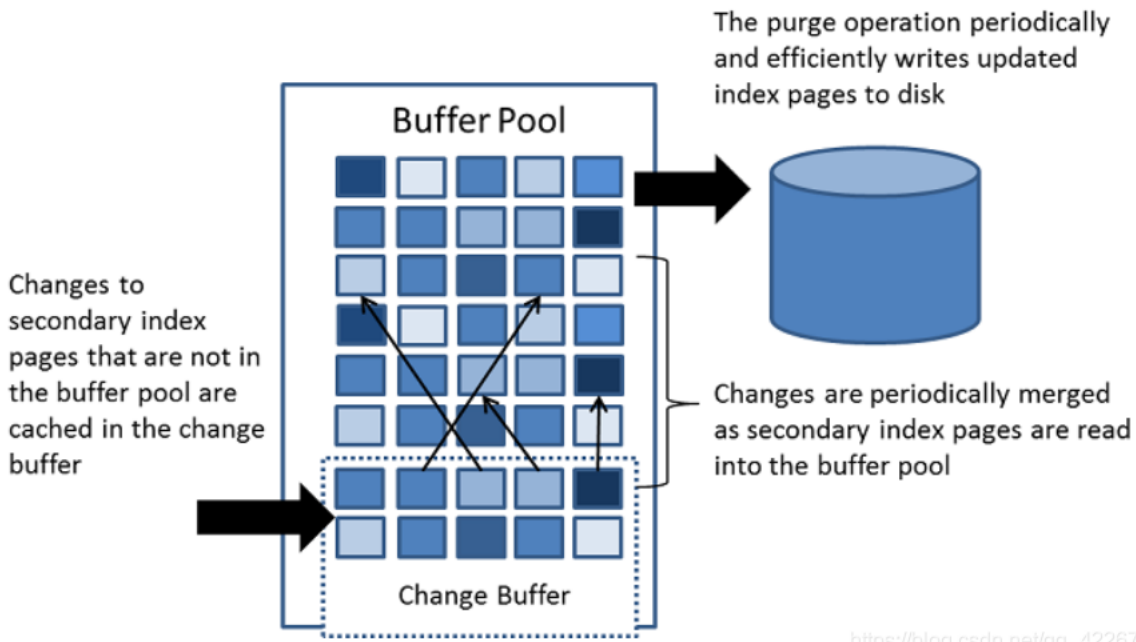
配置更改缓冲区最大大小

该innodb_change_buffer_max_size 变量允许将更改缓冲区的最大大小配置为缓冲池总大小的百分比。默认情况下， innodb_change_buffer_max_size设置为25。最大设置为50。

考虑innodb_change_buffer_max_size在具有大量插入，更新和删除活动的MySQL服务器上进行增加， 在这种情况下，更改缓冲区合并不能跟上新的更改缓冲区条目，从而导致更改缓冲区达到其最大大小限制。

考虑innodb_change_buffer_max_size在使用静态数据进行报告的MySQL服务器上减少 存储空间，或者更改缓冲区消耗的缓冲池共享的内存空间过多，从而导致页面比预期的更快地退出缓冲池。

使用代表性的工作负载测试不同的设置，以确定最佳配置。该 innodb_change_buffer_max_size 设置是动态的，它允许修改设置而无需重新启动服务器。



2.3 优化器算法

5.6版本之后mysql放开了优化器算法

①查看优化器算法

```bash
select @@optimizer_switch;
index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection=on,engine_condition_pus
```

②修改优化器算法

1.全局修改
需要重新进入才能生效

```bash
mysql> set global optimizer_switch='index_condition_pushdown=off';
```

2.hits方式（针对单条语句生效） 官方文档 <https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html>

```bash
例子:
SELECT /*+ NO_RANGE_OPTIMIZATION(t3 PRIMARY, f2_idx) */ f1
  FROM t3 WHERE f1 > 30 AND f1 < 33;
SELECT /*+ BKA(t1) NO_BKA(t2) */ * FROM t1 INNER JOIN t2 WHERE ...;
SELECT /*+ NO_ICP(t1, t2) */ * FROM t1 INNER JOIN t2 WHERE ...;
SELECT /*+ SEMIJOIN(FIRSTMATCH, LOOSESCAN) */ * FROM t1 ...;
EXPLAIN SELECT /*+ NO_ICP(t1) */ * FROM t1 WHERE ...;
SELECT /*+ MERGE(dt) */ * FROM (SELECT * FROM t1) AS dt;
INSERT /*+ SET_VAR(foreign_key_checks=OFF) */ INTO t2 VALUES(2);
```

3.配置文件my.cnf

③优化器算法本身

1.ICP（ Index Condition Pushdown）

原理

> 假设我们默认情况有联合索引
> 我们一条语句先把联合索引叶子节点中的数据页拿到sql层进行过滤，再下推到引擎层再进行一次条件查询的过滤，使我们回表的结果更加精简，较少磁盘读写io数量

**mysql5.6版本之后默认添加的。
最终选择优化器本身算法的优化还是索引有关的最左原则优化，是通过测压的方式最终决定的**

测压语句

Bash | Copy

```
mysqlslap --defaults-file=/etc/my.cnf \
--concurrency=100 --iterations=1 --create-schema='test' \
--query=" 语句" engine=innodb \
--number-of-queries=2000 -uroot -p123 -verbose
```

测压实验

1.关闭默认开启的icp，不做索引优化

Bash | Copy

```
mysql> set global optimizer_switch='index_condition_pushdown=off';
```

```
[root@db01 ~]# mysqlslap --defaults-file=/etc/my.cnf \
> --concurrency=100 --iterations=1 --create-schema='test' \
> --query="  select * from test.t100w where num>913759 and k1='ej' and k2='effg'" engine=innodb \
> --number-of-queries=2000 -uroot -p123 -verbose
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine rbose
        Average number of seconds to run all queries: 519.677 seconds
        Minimum number of seconds to run all queries: 519.677 seconds
        Maximum number of seconds to run all queries: 519.677 seconds
        Number of clients running queries: 100
        Average number of queries per client: 20
```


2.开启icp，不做索引优化

```
[root@db01 ~]# mysqlslap --defaults-file=/etc/my.cnf \
> --concurrency=100 --iterations=1 --create-schema='test' \
> --query=" select * from test.t100w where num>913759 and k1='ej' and k2='effg'" engine=innodb \
> --number-of-queries=2000 -uroot -p123 -verbose
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine rbose
        Average number of seconds to run all queries: 482.698 seconds
        Minimum number of seconds to run all queries: 482.698 seconds
        Maximum number of seconds to run all queries: 482.698 seconds
        Number of clients running queries: 100
        Average number of queries per client: 20

[root@db01 ~]#
```


3.我们现在根据最左原则，建立num,k1,k2的联合索引，关闭icp

```
[root@db01 ~]# mysqlslap --defaults-file=/etc/my.cnf \
> --concurrency=100 --iterations=1 --create-schema='test' \
> --query=" select * from test.t100w where num>913759 and k1='ej' and k2='effg'" engine=innodb \
> --number-of-queries=2000 -uroot -p123 -verbose
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine rbose
        Average number of seconds to run all queries: 0.404 seconds
        Minimum number of seconds to run all queries: 0.404 seconds
        Maximum number of seconds to run all queries: 0.404 seconds
        Number of clients running queries: 100
        Average number of queries per client: 20
```


2.MRR（multi range read）

辅助索引条件查询时,先扫描辅助索引,获得ID值,放在read_rnd_buffer中,由MRR进行排序后,回表查询.

**优化器本身会自动去做选择使用什么优化器算法

我们这里涉及到一个优化器选项 `mrr_cost_based=on` ，强行执行MRR优化器算法。

mrr_cost_based=on默认是开启的，如果强行执行MRR优化器算法,具体使用不适用我们可以通过测压的方式来进行最终的选择。**

Bash | Copy

```
1  set global optimizer_switch='mrr_cost_based=off'
```

%E6%89%A7%E8%A1%8C%E5%99%A8%E5%92%8C%E4%BC%98%E5%8C%96%E5%99%A8%E7%AE%97%E6%B3%95%20%E2%88%9A%20%7C%201%20%E6%89%A7%E8%A1%8C%E8%AE%A1%E5%