---MySql学习笔记---

事务

概念

是一组操作的集合，它是一个不可分割的工作单位，事务会把所有的操作作为一个整体一起向系 统提交或撤销操作请求，即这些操作要么同时成功，要么同时失败。

查看/设置事务提交方式

```
select @@autocommit;

set @@autocommit=0;
```

提交事务

commit;

回滚事务

rollback;

开启事务

```
START TRANSACTION 或 BEGIN ;
```

事务四大特性 ,简称ACID。

- 原子性（Atomicity）：事务是不可分割的最小操作单元，要么全部成功，要么全部失败。
- 一致性（Consistency）：事务完成时，必须使所有的数据都保持一致状态。
- 隔离性（Isolation）：数据库系统提供的隔离机制，保证事务在不受外部并发操作影响的独立 环境下运行。
- 持久性（Durability）：事务一旦提交或回滚，它对数据库中的数据的改变就是永久的。

并发事务问题

- 赃读：一个事务读到另外一个事务还没有提交的数据。比如B读取到了A未提交的数据。
- 不可重复读：一个事务先后读取同一条记录，但两次读取的数据不同，称之为不可重复读。事务A两次读取同一条记录，但是读取到的数据却是不一样的

- 幻读：一个事务按照条件查询数据时，没有对应的数据行，但是在插入数据时，又发现这行数据 已经存在，好像出现了 "幻影"。

事务的隔离级别

Read uncommitted、Read committed、Repeatable Read(默认)、Serializable

- Read uncommitted：未解决赃读、不可重复读和幻读问题
- Read committed：解决赃读问题
- Repeatable Read： 解决赃读和不可重复读问题
- Serializable：解决赃读、不可重复读和幻读问题

查看事务隔离级别

```
SELECT @@TRANSACTION_ISOLATION;
```

设置事务隔离级别

```
SET [ SESSION | GLOBAL ] TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED |
READ COMMITTED | REPEATABLE READ | SERIALIZABLE }
```

事务隔离级别越高，数据越安全，但是性能越低。Serializable级别最高，Read uncommitted级别最低。

存储引擎

MySQL体系结构

- 连接层： 最上层是一些客户端和链接服务，包含本地sock 通信和大多数基于客户端/服务端工具实现的类似于 TCP/IP的通信。主要完成一些类似于连接处理、授权认证、及相关的安全方案。在该层上引入了线程 池的概念，为通过认证安全接入的客户端提供线程。同样在该层上可以实现基于SSL的安全链接。服务器也会为安全接入的每个客户端验证它所具有的操作权限。
- 服务层：第二层架构主要完成大多数的核心服务功能，如SQL接口，并完成缓存的查询，SQL的分析和优化，部 分内置函数的执行。所有跨存储引擎的功能也在这一层实现，如 过程、函数等。在该层，服务器会解 析查询并创建相应的内部解析树，并对其完成相应的优化如确定表的查询的顺序，是否利用索引等， 最后生成相应的执行操作。如果是select语句，服务器还会查询内部的缓存，如果缓存空间足够大， 这样在解决大量读操作的环境中能够很好的提升系统的性能。
- 引擎层： 存储引擎层， 存储引擎真正的负责了MySQL中数据的存储和提取，服务器通过API和存储引擎进行通 信。不同的存储引擎具有不同的功能，这样我们可以根据自己的需要，来选取合适的存储引擎。数据库 中的索引是在存储引擎层实现的。
- 存储层：数据存储层， 主要是将数据(如: redolog、undolog、数据、索引、二进制日志、错误日志、查询 日志、慢查询日志等)存储在文件系统之上，并完成与存储引擎的交互。

建表时指定存储引擎

```
CREATE TABLE 表名(
字段1 字段1类型 [ COMMENT 字段1注释 ] ,
......
字段n 字段n类型 [COMMENT 字段n注释 ]
) ENGINE = INNODB [ COMMENT 表注释 ] ;
```

查询当前数据库支持的存储引擎

```
show engines;
```

存储引擎特点

InnoDB

InnoDB是一种兼顾高可靠性和高性能的通用存储引擎，在 MySQL 5.5 之后，InnoDB是默认的 MySQL 存储引擎。

特点:

- DML操作遵循ACID模型，支持事务；
- 行级锁，提高并发访问性能；
- 支持外键FOREIGN KEY约束，保证数据的完整性和正确性；

文件:

xxx.ibd：xxx代表的是表名，innoDB引擎的每张表都会对应这样一个表空间文件，存储该表的表结 构（frm-早期的 、sdi-新版的）、数据和索引。

逻辑存储结构:

- 表空间 : InnoDB存储引擎逻辑结构的最高层，ibd文件其实就是表空间文件，在表空间中可以 包含多个Segment段。
- 段 : 表空间是由各个段组成的， 常见的段有数据段、索引段、回滚段等。InnoDB中对于段的管理，都是引擎自身完成，不需要人为对其控制，一个段中包含多个区。
- 区 : 区是表空间的单元结构，每个区的大小为1M。 默认情况下， InnoDB存储引擎页大小为 16K，即一个区中一共有64个连续的页。
- 页 : 页是组成区的最小单元，页也是InnoDB 存储引擎磁盘管理的最小单元，每个页的大小默 认为16KB。为了保证页的连续性，InnoDB 存储引擎每次从磁盘申请 4-5 个区。
- 行 : InnoDB 存储引擎是面向行的，也就是说数据是按行进行存放的，在每一行中除了定义表时 所指定的字段以外，还包含两个隐藏字段。

MyISAM

MyISAM是MySQL早期的默认存储引擎。

特点

- 不支持事务
- 不支持外键
- 支持表锁
- 不支持行锁
- 访问速度快

文件

- xxx.sdi：存储表结构信息
- xxx.MYD: 存储数据
- xxx.MYI: 存储索引

Memory

Memory引擎的表数据时存储在内存中的，由于受到硬件问题、或断电问题的影响，只能将这些表作为临时表或缓存使用。

特点

- 内存存放
- hash索引（默认）

文件

- xxx.sdi：存储表结构信息

特点

```
特点 InnoDB   MyISAM  Memory
存储限制 64TB     有   有
事务安全 支持       -  -
锁机制 行锁    表锁   表锁
B+tree索引 支持 支持 支持
Hash索引 -     -    支持
全文索引 支持(5.6版本之后) 支持 -
空间使用 高   低    N/A
内存使用 高   低    中等
批量插入速度 低      高      高
支持外键 支持  -    -
```

存储引擎选择

- InnoDB: 是Mysql的默认存储引擎，支持事务、外键。如果应用对事务的完整性有比较高的要求，在并发条件下要求数据的一致性，数据操作除了插入和查询之外，还包含很多的更新、删除操 作，那么InnoDB存储引擎是比较合适的选择。
- MyISAM ： 如果应用是以读操作和插入操作为主，只有很少的更新和删除操作，并且对事务的完整性、并发性要求不是很高，那么选择这个存储引擎是非常合适的。
- MEMORY：将所有数据保存在内存中，访问速度快，通常用于临时表及缓存。MEMORY的缺陷就是 对表的大小有限制，太大的表无法缓存在内存中，而且无法保障数据的安全性。

索引

介绍

索引（index）是帮助MySQL高效获取数据的数据结构(有序)。在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（指向）数据， 这样就可以在这些数据结构 上实现高级查找算法，这种数据结构就是索引。

特点

优势

- 提高数据检索的效率，降低数据库的IO成本
- 通过索引列对数据进行排序，降低 数据排序的成本，降低CPU的消耗。

劣势

- 索引列也是要占用空间的
- 索引大大提高了查询效率，同时却也降低更新表的速度， 如对表进行INSERT、UPDATE、DELETE时，效率降低。

索引结构

MySQL的索引是在存储引擎层实现的，不同的存储引擎有不同的索引结构

- B+Tree索引：最常见的索引类型，大部分引擎都支持 B+ 树索引
- Hash索引：底层数据结构是用哈希表实现的, 只有精确匹配索引列的查询才有效, 不支持范围查询
- R-tree(空间索 引) ： 空间索引是MyISAM引擎的一个特殊索引类型，主要用于地理空间数据类型，通常使用较少
- Full-text(全文 索引) ： 是一种通过建立倒排索引,快速匹配文档的方式。类似于 Lucene,Solr,ES

索引结构支持情况

从左到右分别为 InnoDB， MyISAM， Memory

- B+tree索引 ： 支持 支持 支持
- Hash 索引 ： 不支持 不支持 支持
- R-tree 索引 ： 不支持 支持 不支持
- Full-text： 5.6版本之后支持 支持 不支持

MySQL索引数据结构对经典的B+Tree进行了优化。在原B+Tree的基础上，增加一个指向相邻叶子节点的链表指针，就形成了带有顺序指针的B+Tree，提高区间访问的性能，利于排序。

在MySQL中，支持hash索引的是Memory存储引擎。 而InnoDB中具有自适应hash功能，hash索引是InnoDB存储引擎根据B+Tree索引在指定条件下自动构建的。

为什么InnoDB存储引擎选择使用B+tree索引结构?

- 相对于二叉树，层级更少，搜索效率高
- 对于B-tree，无论是叶子节点还是非叶子节点，都会保存数据，这样导致一页中存储 的键值减少，指针跟着减少，要同样保存大量数据，只能增加树的高度，导致性能降低
- 相对Hash索引，B+tree支持范围匹配及排序操作

索引分类

- 主键索引： 针对于表中主键创建的索引，默认自动创建, 只能 有一个
- 唯一索引：避免同一个表中某数据列中的值重复 ， 可以有多个

- 常规 索引：快速定位特定数据 ，可以有多个
- 全文 索引：全文索引查找的是文本中的关键词，而不是比较索引中的值，可以有多个

聚集索引&二级索引

- 聚集索引(Clustered Index)： 将数据存储与索引放到了一块，索引结构的叶子 节点保存了行数据：必须有,而且只 有一个
- 二级索引(Secondary Index)：将数据与索引分开存储，索引结构的叶子节点关 联的是对应的主键：可以存在多个

聚集索引选取规则

- 如果存在主键，主键索引就是聚集索引
- 如果不存在主键，将使用第一个唯一（UNIQUE）索引作为聚集索引
- 如果表没有主键，或没有合适的唯一索引，则InnoDB会自动生成一个rowid作为隐藏的聚集索 引

区别

- 聚集索引的叶子节点下挂的是这一行的数据
- 二级索引的叶子节点下挂的是该字段值对应的主键值

回表查询

先到二级索引中查找数据，找到主键值，然后再到聚集索引中根据主键值，获取 数据的方式，就称之为回表查询

索引使用

创建索引

```
CREATE [ UNIQUE | FULLTEXT ] INDEX index_name ON table_name (
index_col_name,... );
```

查看索引

```
SHOW INDEX FROM table_name;
```

删除索引

```
DROP INDEX index_name ON table_name;
```

SQL性能分析

SQL执行频率

查看当前数据库的INSERT、UPDATE、DELETE、SELECT的访问频次

语法

```
show [session|global] status
```

- session 是查看当前会话
- global 是查询全局数据

SHOW GLOBAL STATUS LIKE 'Com___';

- Com_delete: 删除次数
- Com_insert: 插入次数
- Com_select: 查询次数
- Com_update: 更新次数

通过上述指令，我们可以查看到当前数据库到底是以查询为主，还是以增删改为主，从而为数据 库优化提供参考依据。 如果是以增删改为主，我们可以考虑不对其进行索引的优化。 如果是以 查询为主，那么就要考虑对数据库的索引进行优化了。

慢查询日志

慢查询日志记录了所有执行时间超过指定参数（long_query_time，单位：秒，默认10秒）的所有 SQL语句的日志。

MySQL的慢查询日志默认没有开启

开启

如果要开启慢查询日志，需要在MySQL的配置文件（/etc/my.cnf）中配置如下信息：

```
# 开启MySQL慢日志查询开关
slow_query_log=1
# 设置慢日志的时间为2秒，SQL语句执行时间超过2秒，就会视为慢查询，记录慢查询日志
long_query_time=2
```

profile详情

show profiles 能够在做SQL优化时帮助我们了解时间都耗费到哪里去了。通过have_profiling 参数，能够看到当前MySQL是否支持profile操作

```
SELECT @@have_profiling ;
```

session/global级别开启profiling：

```
SET profiling = 1;
```

查看每一条SQL的耗时基本情况：

```
show profiles;
```

```
mysql> SET profiling = 1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> explain select count(*) from tb_hotel
    -> ;
+----+-------------+----------+------------+-------+---------------+---------+--
-------+------+------+----------+-------------+
| id | select_type | table    | partitions | type  | possible_keys | key     |
key_len | ref  | rows | filtered | Extra       |
+----+-------------+----------+------------+-------+---------------+---------+--
-------+------+------+----------+-------------+
|  1 | SIMPLE      | tb_hotel | NULL       | index | NULL          | PRIMARY | 8
      | NULL |  201 |   100.00 | Using index |
+----+-------------+----------+------------+-------+---------------+---------+--
-------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql>  select count(*) from tb_hotel
    -> ;
+----------+
| count(*) |
+----------+
|      201 |
+----------+
1 row in set (0.00 sec)

mysql> show profiles;
+----------+------------+---------------------------------------+
| Query_ID | Duration   | Query                                 |
+----------+------------+---------------------------------------+
|        1 | 0.00086325 | explain select count(*) from tb_hotel |
|        2 | 0.00165000 | select count(*) from tb_hotel         |
+----------+------------+---------------------------------------+
2 rows in set, 1 warning (0.00 sec)

mysql>
```

```
mysql> show profiles;
+----------+------------+-----------------------------------------------------------
------+
```

```
| Query_ID | Duration   | Query
    |
+----------+------------+-----------------------------------------------------
------+
|        1 | 0.00086325 | explain select count(*) from tb_hotel
      |
|        2 | 0.00165000 | select count(*) from tb_hotel
      |
|        3 | 0.00015850 | SELECT DATABASE()
      |
|        4 | 0.01245925 | show tables
      |
|        5 | 0.00442350 | explain select * from score
      |
|        6 | 0.00570150 | explain select * from student
      |
|        7 | 0.00038025 | explain select * from student where
student_no=202012340110 |
|        8 | 0.00062775 | explain select * from student where class_no=1002
      |
+----------+------------+-----------------------------------------------------
------+
8 rows in set, 1 warning (0.00 sec)

mysql>
```

explain

语法:

直接在select语句之前加上关键字 explain / desc

EXPLAIN SELECT 字段列表 FROM 表名 WHERE 条件 ;

```
mysql> use student1;
Database changed
mysql> show tables;
+-------------------------+
| Tables_in_student1      |
+-------------------------+
| administrators          |
| administrators_password |
| class                   |
| course                  |
| forum                   |
| login_log               |
| news                    |
| score                   |
| student                 |
| student_password        |
```

```
| teach                  |
| teacher                |
| teacher_password       |
+------------------------+
13 rows in set (0.01 sec)

mysql> explain select * from score;
+----+-------------+-------+------------+------+---------------+------+---------
+------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len |
ref  | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------+---------
+------+------+----------+-------+
|  1 | SIMPLE      | score | NULL       | ALL  | NULL          | NULL | NULL
| NULL | 7199 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------+---------
+------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)

mysql> explain select * from student;
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL
| NULL |  600 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)

mysql> explain select * from student where student_no=202012340110;
+----+-------------+---------+------------+-------+---------------+---------+---
------+-------+------+----------+-------+
| id | select_type | table   | partitions | type  | possible_keys | key     |
key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+-------+---------------+---------+---
------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | const | PRIMARY,no    | PRIMARY | 8
      | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+-------+---------------+---------+---
------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where class_no=1002;
+----+-------------+---------+------------+------+---------------+----------+---
------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key      |
key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+----------+---
------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | class_no      | class_no | 8
      | const |   60 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+----------+---
------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql>
```

说明：

- id： select查询的序列号，表示查询中执行select子句或者是操作表的顺序 (id相同，执行顺序从上到下；id不同，值越大，越先执行)。
- select_type： 表示 SELECT 的类型，常见的取值有 SIMPLE（简单表，即不使用表连接 或者子查询）、PRIMARY（主查询，即外层的查询）、 UNION（UNION 中的第二个或者后面的查询语句）、 SUBQUERY（SELECT/WHERE之后包含了子查询）等
- type ： 表示连接类型，性能由好到差的连接类型为NULL、system、const、 eq_ref、ref、range、 index、all 。
- possible_key： 显示可能应用在这张表上的索引，一个或多个。
- key： 实际使用的索引，如果为NULL，则没有使用索引。
- key_len ： 表示索引中使用的字节数， 该值为索引字段最大可能长度，并非实际使用长度，在不损失精确性的前提下， 长度越短越好 。
- rows ： MySQL认为必须要执行查询的行数，在innodb引擎的表中，是一个估计值， 可能并不总是准确的。
- filtered 表示返回结果的行数占需读取行数的百分比， filtered 的值越大越好。

索引使用

最左前缀法则

如果索引了多列（联合索引），要遵守最左前缀法则。最左前缀法则指的是查询从索引的最左列开始，并且不跳过索引中的列。如果跳跃某一列，索引将会部分失效(后面的字段索引失效)。

对于最左前缀法则指的是，查询时，最左变的列，也就是profession必须存在，否则索引全部失效。 而且中间不能跳过某一列，否则该列后面的字段索引将失效。

address、id_card、email建立索引

```
mysql> explain select * from student where address="13";
+----+-------------+---------+------------+------+---------------+--------+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key    | key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+--------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+--------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> explain select * from student where id_card="111";
+----+-------------+---------+------------+------+---------------+------+---------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+---------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL
| NULL |  600 |    10.00 | Using where |
+----+-------------+---------+------------+------+---------------+------+---------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where email="222";
+----+-------------+---------+------------+------+---------------+------+---------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+---------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL
| NULL |  600 |    10.00 | Using where |
+----+-------------+---------+------------+------+---------------+------+---------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address="1" and id_card="1";
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key    |
key_len | ref         | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 245
    | const,const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address="1" and id_card="1" and
email="111";
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------------------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key    |
key_len | ref               | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------------------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 368
    | const,const,const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------------------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address="1" and email="111";
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-----------------------+
| id | select_type | table   | partitions | type | possible_keys | key    |
key_len | ref   | rows | filtered | Extra                 |
```

```
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-----------------------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 183
    | const |    1 |    10.00 | Using index condition |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where email="2" and address="1";
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-----------------------+
| id | select_type | table   | partitions | type | possible_keys | key    |
key_len | ref   | rows | filtered | Extra                 |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-----------------------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 183
    | const |    1 |    10.00 | Using index condition |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

范围查询

联合索引中，出现范围查询(>,<)，范围查询右侧的列索引失效。

在业务允许的情况下，尽可能的使用类似于 >= 或 <= 这类的范围查询，而避免使用 > 或 <

索引失效情况

索引列运算

不要在索引列上进行运算操作， 索引将失效。

例：

```
explain select * from tb_user where substring(phone,10,2) = '15'
```

字符串不加引号

字符串类型字段使用时，不加引号，索引将失效。

```
mysql> explain select * from student where address=0;
```

```
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | index1        | NULL | NULL
| NULL |  600 |    10.00 | Using where |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
1 row in set, 3 warnings (0.00 sec)

mysql> explain select * from student where address="0";
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key    |
key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 183
    | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

模糊查询

如果仅仅是尾部模糊匹配，索引不会失效。如果是头部模糊匹配，索引失效。

在like模糊查询中，在关键字后面加%，索引可以生效。而如果在关键字 前面加了%，索引将会失效

```
mysql> explain select * from student where address="1";
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key    |
key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 183
    | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address like "%1";
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
```

```
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL
| NULL |  600 |    11.11 | Using where |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address like "1%";
+----+-------------+---------+------------+-------+---------------+--------+----
-----+------+------+----------+-----------------------+
| id | select_type | table   | partitions | type  | possible_keys | key    |
key_len | ref  | rows | filtered | Extra                 |
+----+-------------+---------+------------+-------+---------------+--------+----
-----+------+------+----------+-----------------------+
|  1 | SIMPLE      | student | NULL       | range | index1        | index1 | 183
    | NULL |    1 |   100.00 | Using index condition |
+----+-------------+---------+------------+-------+---------------+--------+----
-----+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address like "%1%";
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL
| NULL |  600 |    11.11 | Using where |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address like "1";
+----+-------------+---------+------------+-------+---------------+--------+----
-----+------+------+----------+-----------------------+
| id | select_type | table   | partitions | type  | possible_keys | key    |
key_len | ref  | rows | filtered | Extra                 |
+----+-------------+---------+------------+-------+---------------+--------+----
-----+------+------+----------+-----------------------+
|  1 | SIMPLE      | student | NULL       | range | index1        | index1 | 183
    | NULL |    1 |   100.00 | Using index condition |
+----+-------------+---------+------------+-------+---------------+--------+----
-----+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

or连接条件

用or分割开的条件， 如果or前的条件中的列有索引，而后面的列中没有索引，那么涉及的索引都不会被用到。

```
mysql> explain select * from student where address="1" and sex="1";
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key    |
key_len | ref   | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 183
    | const |    1 |    10.00 | Using where |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address="1" or sex="1";
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | index1        | NULL | NULL
| NULL |  600 |    19.00 | Using where |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address="1" or email="1";
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | index1        | NULL | NULL
| NULL |  600 |    19.00 | Using where |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address="1" or id_card="1";
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | index1        | NULL | NULL
| NULL |  600 |    19.00 | Using where |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where address="1" or class_no=1001;
+----+-------------+---------+------------+-------------+-----------------+-----
------------+---------+------+------+----------+-------------------------------
----------------+
```

```
| id | select_type | table   | partitions | type        | possible_keys   | key
          | key_len | ref  | rows | filtered | Extra
             |
+----+-------------+---------+------------+-------------+-----------------+-----
------------+---------+------+------+----------+--------------------------------
----------------+
|  1 | SIMPLE      | student | NULL       | index_merge | class_no,index1 |
index1,class_no | 183,8   | NULL |   61 |   100.00 | Using
sort_union(index1,class_no); Using where |
+----+-------------+---------+------------+-------------+-----------------+-----
------------+---------+------+------+----------+--------------------------------
----------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

数据分布影响

如果MySQL评估使用索引比全表更慢，则不使用索引。

就是因为MySQL在查询时，会评估使用索引的效率与走全表扫描的效率，如果走全表扫描更快，则放弃索引，走全表扫描。 因为索引是用来索引少量数据的，如果通过索引查询返回大批量的数据，则还不 如走全表扫描来的快，此时索引就会失效。

```
mysql> explain select * from student where class_no>=1002;
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | class_no      | NULL | NULL
| NULL |  600 |    90.00 | Using where |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student where class_no>=1009;
+----+-------------+---------+------------+-------+---------------+----------+--
-------+------+------+----------+-----------------------+
| id | select_type | table   | partitions | type  | possible_keys | key      |
key_len | ref  | rows | filtered | Extra                 |
+----+-------------+---------+------------+-------+---------------+----------+--
-------+------+------+----------+-----------------------+
|  1 | SIMPLE      | student | NULL       | range | class_no      | class_no | 8
      | NULL |  120 |   100.00 | Using index condition |
+----+-------------+---------+------------+-------+---------------+----------+--
-------+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

SQL提示

SQL提示，是优化数据库的一个重要手段，简单来说，就是在SQL语句中加入一些人为的提示来达到优化操作的目的。

- use index： 建议MySQL使用哪一个索引完成此次查询（仅仅是建议，mysql内部还会再次进 行评估）。
- ignore index： 忽略指定的索引。
- force index： 强制使用索引。

```
mysql> explain select * from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys        | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address | index1
| 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student use index(index_address) where address="1";
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index_address | index_address
| 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student ignore index(index_address,index1) where
address="1";
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL
| NULL |  600 |    10.00 | Using where |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> explain select * from student ignore index(index_address) where
address="1";
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key    |
key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1        | index1 | 183
    | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+--------+-----
----+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student ignore index(index1) where address="1";
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index_address | index_address
| 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student force index(index_address) where
address="1";
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index_address | index_address
| 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+-------------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

覆盖索引

尽量使用覆盖索引，减少select *。 那么什么是覆盖索引呢？ 覆盖索引是指 查询使用了索引，并 且需要返回的列，在该索引中已经全部能够找到 。

Extra含义：

- Using where; Using Index：查找使用了索引，但是需要的数据都在索引列中能找到，所以不需要回表查询数据
- Using index condition：查找使用了索引，但是需要回表查询数据

```
mysql> explain select * from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys        | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address | index1
| 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)

mysql> explain select address,email from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys        | key
| key_len | ref   | rows | filtered | Extra       |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address | index1
| 183     | const |    1 |   100.00 | Using index |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select address,id_card from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys        | key
| key_len | ref   | rows | filtered | Extra       |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address | index1
| 183     | const |    1 |   100.00 | Using index |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select address,id_card,email from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys        | key
| key_len | ref   | rows | filtered | Extra       |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address | index1
| 183     | const |    1 |   100.00 | Using index |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> explain select address,id_card,email,sex from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys        | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address | index1
| 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select sex from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys        | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address | index1
| 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select student_no from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
--------+---------+-------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys        | key
      | key_len | ref   | rows | filtered | Extra       |
+----+-------------+---------+------------+------+----------------------+-------
--------+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address |
index_address | 183     | const |    1 |   100.00 | Using index |
+----+-------------+---------+------------+------+----------------------+-------
--------+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select student_no,class_no from student where address="1";
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys        | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ref  | index1,index_address | index1
| 183     | const |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+----------------------+-------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

因为，在student表中有一个联合索引index1，

该索引关联了三个字段 address、id_card、email，

而这个索引也是一个二级索引，所以叶子节点下面挂的是这一行的主 键id。

所以当我们查询返回的数据在 id、 address、id_card、email 之中，则直接走二级索引直接返回数据了。

如果超出这个范围，就需要拿到主键id，再去扫描聚集索引，这个过程就是回表。

而我们如果一直使用select * 查询返回所有字段值，很容易就会造成回表 查询（除非是根据主键查询，此时只会扫描聚集索引）。

前缀索引

当字段类型为字符串（varchar，text，longtext等）时，有时候需要索引很长的字符串，这会让 索引变得很大，查询时，浪费大量的磁盘IO， 影响查询效率。此时可以只将字符串的一部分前缀，建 立索引，这样可以大大节约索引空间，从而提高索引效率。

语法：

```
create index xxxx on table_name(column(n)) ;
```

```
mysql> create index index_email on student(email(5));
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from student;
+---------+------------+---------------+--------------+-------------+-----------
+-------------+----------+--------+------+------------+---------+---------------
+---------+------------+
| Table   | Non_unique | Key_name      | Seq_in_index | Column_name | Collation |
Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
Visible | Expression |
+---------+------------+---------------+--------------+-------------+-----------
+-------------+----------+--------+------+------------+---------+---------------
+---------+------------+
| student |          0 | PRIMARY       |            1 | student_no  | A
|         600 |     NULL |   NULL |      | BTREE      |         |
| YES     | NULL       |
| student |          1 | class_no      |            1 | class_no    | A
|          10 |     NULL |   NULL |      | BTREE      |         |
| YES     | NULL       |
| student |          1 | index1        |            1 | address     | A
|         600 |     NULL |   NULL | YES  | BTREE      |         |
| YES     | NULL       |
| student |          1 | index1        |            2 | id_card     | A
|         600 |     NULL |   NULL |      | BTREE      |         |
| YES     | NULL       |
```

```
| student |          1 | index1        |            3 | email       | A
|         600 |     NULL |   NULL | YES  | BTREE      |         |
| YES     | NULL       |
| student |          1 | index_address |            1 | address     | A
|         600 |     NULL |   NULL | YES  | BTREE      |         |
| YES     | NULL       |
| student |          1 | index_email   |            1 | email       | A
|         600 |        5 |   NULL | YES  | BTREE      |         |
| YES     | NULL       |
+---------+------------+---------------+--------------+-------------+-----------
+-------------+----------+--------+------+------------+---------+--------------
+---------+------------+
7 rows in set (0.00 sec)

mysql>
```

前缀长度：

可以根据索引的选择性来决定，而选择性是指不重复的索引值（基数）和数据表的记录总数的比值， 索引选择性越高则查询效率越高， 唯一索引的选择性是1，这是最好的索引选择性，性能也是最好的。

```
select count(distinct email) / count(*) from student ;
select count(distinct substring(email,1,5)) / count(*) from student ;
```

```
mysql> select count(distinct email) / count(*) from student ;
+-----------------------------------+
| count(distinct email) / count(*) |
+-----------------------------------+
|                            1.0000 |
+-----------------------------------+
1 row in set (0.00 sec)

mysql> select count(distinct substring(email,1,5)) / count(*) from student ;
+-------------------------------------------------+
| count(distinct substring(email,1,5)) / count(*) |
+-------------------------------------------------+
|                                          1.0000 |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> select count(distinct substring(email,1,4)) / count(*) from student ;
+-------------------------------------------------+
| count(distinct substring(email,1,4)) / count(*) |
+-------------------------------------------------+
|                                          1.0000 |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> select count(distinct substring(email,1,3)) / count(*) from student ;
```

```
+-------------------------------------------------+
| count(distinct substring(email,1,3)) / count(*) |
+-------------------------------------------------+
|                                          0.9933 |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> select count(distinct substring(email,1,2)) / count(*) from student ;
+-------------------------------------------------+
| count(distinct substring(email,1,2)) / count(*) |
+-------------------------------------------------+
|                                          0.8000 |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> select count(distinct substring(email,1,1)) / count(*) from student ;
+-------------------------------------------------+
| count(distinct substring(email,1,1)) / count(*) |
+-------------------------------------------------+
|                                          0.0600 |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

单列索引与联合索引

- 单列索引：即一个索引只包含单个列。
- 联合索引：即一个索引包含了多个列。

在业务场景中，如果存在多个查询条件，考虑针对于查询字段建立索引时，建议建立联合索引， 而非单列索引。

索引设计原则

- 针对于数据量较大，且查询比较频繁的表建立索引。
- 针对于常作为查询条件（where）、排序（order by）、分组（group by）操作的字段建立索引。
- 尽量选择区分度高的列作为索引，尽量建立唯一索引，区分度越高，使用索引的效率越高。
- 如果是字符串类型的字段，字段的长度较长，可以针对于字段的特点，建立前缀索引。
- 尽量使用联合索引，减少单列索引，查询时，联合索引很多时候可以覆盖索引，节省存储空间， 避免回表，提高查询效率。
- 要控制索引的数量，索引并不是多多益善，索引越多，维护索引结构的代价也就越大，会影响增删改的效率。
- 如果索引列不能存储NULL值，请在创建表时使用NOT NULL约束它。当优化器知道每列是否包含NULL值时，它可以更好地确定哪个索引最有效地用于查询。

SQL优化

插入数据

insert

如果我们需要一次性往数据库表中插入多条记录，可以从以下三个方面进行优化。

1. 批量插入数据

```
Insert into tb_test values(1,'Tom'),(2,'Cat'),(3,'Jerry');
```

2. 手动控制事务

```
start transaction;
insert into tb_test values(1,'Tom'),(2,'Cat'),(3,'Jerry');
insert into tb_test values(4,'Tom'),(5,'Cat'),(6,'Jerry');
insert into tb_test values(7,'Tom'),(8,'Cat'),(9,'Jerry');
commit;
```

3. 主键顺序插入，性能要高于乱序插入

- 主键乱序插入 : 8 1 9 21 88 2 4 15 89 5 7 3
- 主键顺序插入 : 1 2 3 4 5 7 8 9 15 21 88 89

大批量插入数据

如果一次性需要插入大批量数据(比如: 几百万的记录)，使用insert语句插入性能较低，此时可以使 用MySQL数据库提供的load指令进行插入。操作如下：

-- 客户端连接服务端时，加上参数 --local-infile

```
mysql --local-infile -u root -p
```

设置全局参数local_infile为1，开启从本地加载文件导入数据的开关

```
set global local_infile = 1;
```

执行load指令将准备好的数据，加载到表结构中

```
load data local infile '/root/sql1.log' into table tb_user fields
terminated by ',' lines terminated by '\n' ;
```

在load时，主键顺序插入性能高于乱序插入

主键优化

数据组织方式

在InnoDB存储引擎中，表数据都是根据主键顺序组织存放的，这种存储方式的表称为索引组织表 (index organized table IOT)。

行数据，都是存储在聚集索引的叶子节点上的。

在InnoDB引擎中，数据行是记录在逻辑结构 page 页中的，而每一个页的大小是固定的，默认16K。 那也就意味着， 一个页中所存储的行也是有限的，如果插入的数据行row在该页存储不小，将会存储 到下一个页中，页与页之间会通过指针连接。

页分裂

页可以为空，也可以填充一半，也可以填充100%。每个页包含了2-N行数据(如果一行数据过大，会行溢出)，根据主键排列。

主键顺序插入效果：

- 从磁盘中申请页， 主键顺序插入
- 第一个页没有满，继续往第一页插入
- 当第一个也写满之后，再写入第二个页，页与页之间会通过指针连接
- 当第二页写满了，再往第三页写入

主键乱序插入效果：

- 假如1,2页都已经写满了，存放了数据，第一页为1、5、9、23、47，第二页为55、67、89、101、107
- 此时再插入id为50的记录
- 按照顺序，应该存储在47之后
- 但是47所在的1页，已经写满了，存储不了50对应的数据了。 那么此时会开辟一个新的页 3。
- 但是并不会直接将50存入3页，而是会将1页后一半的数据，移动到3页，然后在3页，插入50。
- 移动数据，并插入id为50的数据之后，那么此时，这三个页之间的数据顺序是有问题的。 1的下一个 页，应该是3， 3#的下一个页是2。 所以，此时，需要重新设置链表指针。
- 上述的这种现象，称之为 "页分裂"，是比较耗费性能的操作。

页合并

现有3页数据：

第一页为1到8

第二页为9到16

第三页为17到19

当我们对已有数据进行删除时，具体的效果如下:

- 当删除一行记录时，实际上记录并没有被物理删除，只是记录被标记（flaged）为删除并且它的空间 变得允许被其他记录声明使用。
- 当我们继续删除2号页的数据记录
- 当页中删除的记录达到 MERGE_THRESHOLD（默认为页的50%），InnoDB会开始寻找最靠近的页（前 或后）看看是否可以将两个页合并以优化空间使用。
- 二号页的数据只剩9到12，三号的数据为17到19，三号页的所有数据到二号页的所有数据的后面
- 再次插入新的数据21，则直接插入3号页
- 个里面所发生的合并页的这个现象，就称之为 "页合并"。

MERGE_THRESHOLD：合并页的阈值，可以自己设置，在创建表或者创建索引时指定。

索引设计原则

- 满足业务需求的情况下，尽量降低主键的长度。
- 插入数据时，尽量选择顺序插入，选择使用AUTO_INCREMENT自增主键。
- 尽量不要使用UUID做主键或者是其他自然主键，如身份证号。
- 业务操作时，避免对主键的修改。

order by优化

MySQL的排序，有两种方式：

- Using filesort : 通过表的索引或全表扫描，读取满足条件的数据行，然后在排序缓冲区sort buffer中完成排序操作，所有不是通过索引直接返回排序结果的排序都叫 FileSort 排序。
- Using index : 通过有序索引顺序扫描直接返回有序数据，这种情况即为 using index，不需要 额外排序，操作效率高。

对于以上的两种排序方式，Using index的性能高，而Using filesort的性能低，我们在优化排序 操作时，尽量要优化为 Using index。

Extra中出现了 Backward index scan，这个代表反向扫描索 引，因为在MySQL中我们创建的索引，默认索引的叶子节点是从小到大排序的，而此时我们查询排序 时，是从大到小，所以，在扫描时，就是反向扫描，就会出现 Backward index scan。 在 MySQL8版本中，支持降序索引，我们也可以创建降序索引。

排序时,也需要满足最左前缀法则,否则也会出现 filesort。

order by优化原则:

- 根据排序字段建立合适的索引，多字段排序时，也遵循最左前缀法则。
- 尽量使用覆盖索引。
- 多字段排序, 一个升序一个降序，此时需要注意联合索引在创建时的规则（ASC/DESC）。
- 如果不可避免的出现filesort，大数据量排序时，可以适当增大排序缓冲区大小 sort_buffer_size(默认256k)。

group by优化

在分组操作中，我们需要通过以下两点进行优化，以提升性能:

- 在分组操作时，可以通过索引来提高效率。
- 分组操作时，索引的使用也是满足最左前缀法则的。

limit优化

在数据量比较大时，如果进行limit分页查询，在查询时，越往后，分页查询效率越低。

因为，当在进行分页查询时，如果执行 limit 2000000,10 ，此时需要MySQL排序前2000010 记 录，仅仅返回 2000000 - 2000010 的记录，其他记录丢弃，查询排序的代价非常大 。

优化思路: 一般分页查询时，通过创建 覆盖索引 能够比较好地提高性能，可以通过覆盖索引加子查 询形式进行优化。

```
explain select * from tb_sku t , (select id from tb_sku order by id
limit 2000000,10) a where t.id = a.id;
```

```
 explain select * from student limit 550,10;
```

转换成:

```
explain select * from student s , (select student_no from student order by
student_no limit 550,10) a where s.student_no=a.student_no;
```

```
mysql> explain select * from student limit 550,10;
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len
| ref  | rows | filtered | Extra |
```

```
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL
| NULL |  600 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+------+-------
--+------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from student s , (select student_no from student order by
student_no limit 550,10) a where s.student_no=a.student_no;
+----+-------------+------------+------------+--------+---------------+---------
+---------+--------------+------+----------+-------------+
| id | select_type | table      | partitions | type   | possible_keys | key     |
key_len | ref          | rows | filtered | Extra       |
+----+-------------+------------+------------+--------+---------------+---------
+---------+--------------+------+----------+-------------+
|  1 | PRIMARY     | <derived2> | NULL       | ALL    | NULL          | NULL
| NULL    | NULL         |  560 |   100.00 | NULL        |
|  1 | PRIMARY     | s          | NULL       | eq_ref | PRIMARY       | PRIMARY
| 8       | a.student_no |    1 |   100.00 | NULL        |
|  2 | DERIVED     | student    | NULL       | index  | NULL          | PRIMARY
| 8       | NULL         |  560 |   100.00 | Using index |
+----+-------------+------------+------------+--------+---------------+---------
+---------+--------------+------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)

mysql>
```

```
mysql>  select * from student limit 550,10;
+--------------+--------+-----+-----------------+-------------------------+----
--------+----------------------+--------------------+-------------------+-------
----------+----------+-------+---------+
| student_no   | name   | sex | telephone_number | family_telephone_number |
birthday   | address              | id_card            | email             |
dormitory_number | class_no | state | remarks |
+--------------+--------+-----+-----------------+-------------------------+----
--------+----------------------+--------------------+-------------------+-------
----------+----------+-------+---------+
| 202012341011 | 慕青   | 女  | 13907538010      | 13105786404             | 2003-
11-09 | 邹平路138号-11-7     | 433099200311098637 | e3sloc2ub@263.net | 2栋208
      |     1010 | 在读  |         |
| 202012341012 | 皇欢   | 女  | 15901101707      | 13405035307             | 2001-
06-23 | 瞿塘峡街140号-6-10   | 421799200106236569 | pxk3bu@yahoo.com  | 1栋304
      |     1010 | 在读  |         |
| 202012341013 | 苍群贵 | 男  | 13908394412      | 13104336454             | 2003-
09-09 | 江苏广场120号-8-2    | 434806200309096637 | ffidoik@yahoo.com | 9栋305
      |     1010 | 在读  |         |
| 202012341014 | 郗磊信 | 男  | 15800733895      | 15201011062             | 2003-
09-17 | 巫峡大厦35号-18-2    | 429097200309174160 | y4of3u00@163.com  | 9栋305
      |     1010 | 在读  |         |
| 202012341015 | 邵凤婕 | 女  | 15102684762      | 15203383107             | 2003-
10-07 | 寿康路105号-4-4      | 431055200310078167 | ewlmjj2@yahoo.com | 6栋217
      |     1010 | 在读  |         |
```

```
| 202012341016 | 哈荣    | 女   | 13708197402      | 13107044497             | 1999-
07-06 | 洞庭湖广场109号-17-3 | 430250199907062813 | bup6c8nt8@ask.com | 6栋315
      |     1010 | 在读   |          |
| 202012341017 | 雍彩华 | 女   | 15506433243      | 13907411224             | 1999-
11-10 | 宁阳广场68号-13-8    | 434344199911106287 | cfxe3am5@163.com  | 4栋412
      |     1010 | 在读   |          |
| 202012341018 | 漆新若 | 男   | 15305892524      | 15804621277             | 2001-
10-17 | 兴安支街102号-16-4   | 429258200110176194 | r546w7z68@163.net | 7栋125
      |     1010 | 在读   |          |
| 202012341019 | 史心军 | 男   | 15906726131      | 15207121427             | 1999-
05-20 | 扬州街73号-5-6       | 440299199905202057 | fitg7hzw@0355.net | 11栋325
       |     1010 | 在读   |          |
| 202012341020 | 续青    | 女   | 15201436477      | 15900795237             | 2002-
09-12 | 祚山路100号-10-2     | 423416200209122512 | dv5a2a@yahoo.com  | 2栋315
      |     1010 | 在读   |          |
+--------------+--------+-----+------------------+-------------------------+----
--------+----------------------+--------------------+-------------------+-------
----------+----------+-------+---------+
10 rows in set (0.01 sec)

mysql>  select * from student s , (select student_no from student order by
student_no limit 550,10) a where s.student_no=a.student_no;
+--------------+--------+-----+------------------+-------------------------+----
--------+----------------------+--------------------+-------------------+-------
----------+----------+-------+---------+--------------+
| student_no   | name   | sex | telephone_number | family_telephone_number |
birthday   | address              | id_card            | email             |
dormitory_number | class_no | state | remarks | student_no   |
+--------------+--------+-----+------------------+-------------------------+----
--------+----------------------+--------------------+-------------------+-------
----------+----------+-------+---------+--------------+
| 202012341011 | 慕青    | 女   | 13907538010      | 13105786404             | 2003-
11-09 | 邹平路138号-11-7     | 433099200311098637 | e3sloc2ub@263.net | 2栋208
      |     1010 | 在读   |          | 202012341011 |
| 202012341012 | 皇欢    | 女   | 15901101707      | 13405035307             | 2001-
06-23 | 瞿塘峡街140号-6-10   | 421799200106236569 | pxk3bu@yahoo.com  | 1栋304
      |     1010 | 在读   |          | 202012341012 |
| 202012341013 | 苍群贵 | 男   | 13908394412      | 13104336454             | 2003-
09-09 | 江苏广场120号-8-2    | 434806200309096637 | ffidoik@yahoo.com | 9栋305
      |     1010 | 在读   |          | 202012341013 |
| 202012341014 | 郗磊信 | 男   | 15800733895      | 15201011062             | 2003-
09-17 | 巫峡大厦35号-18-2    | 429097200309174160 | y4of3u00@163.com  | 9栋305
      |     1010 | 在读   |          | 202012341014 |
| 202012341015 | 邵凤婕 | 女   | 15102684762      | 15203383107             | 2003-
10-07 | 寿康路105号-4-4      | 431055200310078167 | ewlmjj2@yahoo.com | 6栋217
      |     1010 | 在读   |          | 202012341015 |
| 202012341016 | 哈荣    | 女   | 13708197402      | 13107044497             | 1999-
07-06 | 洞庭湖广场109号-17-3 | 430250199907062813 | bup6c8nt8@ask.com | 6栋315
      |     1010 | 在读   |          | 202012341016 |
| 202012341017 | 雍彩华 | 女   | 15506433243      | 13907411224             | 1999-
11-10 | 宁阳广场68号-13-8    | 434344199911106287 | cfxe3am5@163.com  | 4栋412
      |     1010 | 在读   |          | 202012341017 |
| 202012341018 | 漆新若 | 男   | 15305892524      | 15804621277             | 2001-
10-17 | 兴安支街102号-16-4   | 429258200110176194 | r546w7z68@163.net | 7栋125
      |     1010 | 在读   |          | 202012341018 |
| 202012341019 | 史心军 | 男   | 15906726131      | 15207121427             | 1999-
05-20 | 扬州街73号-5-6       | 440299199905202057 | fitg7hzw@0355.net | 11栋325
       |     1010 | 在读   |          | 202012341019 |
```

```
| 202012341020 | 续青    | 女   | 15201436477      | 15900795237             | 2002-
09-12 | 祚山路100号-10-2      | 423416200209122512 | dv5a2a@yahoo.com   | 2栋315
      |     1010 | 在读   |          | 202012341020 |
+--------------+--------+-----+------------------+-------------------------+----
--------+-----------------------+--------------------+--------------------+-------
-----------+----------+-------+---------+--------------+
10 rows in set (0.00 sec)

mysql> show profiles;
+----------+------------+-------------------------------------------------------
-------------------------------------------------------------------------------
+
| Query_ID | Duration   | Query
                                                                                |
+----------+------------+-------------------------------------------------------
-------------------------------------------------------------------------------
+
|        1 | 0.00025250 | explain select * from student limit 550,10
                                                                                |
|        2 | 0.00026125 | explain select * from student limit 550,10
                                                                                |
|        3 | 0.00078550 | explain select * from student s , (select student_no
from student order by id limit 550,10) a where s.student_no=a.student_no
|
|        4 | 0.00141850 | explain select * from student s , (select student_no
from student order by student_no limit 550,10) a where s.student_no=a.student_no
|
|        5 | 0.00818275 | select * from student limit 550,10
                                                                                |
|        6 | 0.00094250 | select * from student s , (select student_no from
student order by student_no limit 550,10) a where s.student_no=a.student_no
    |
+----------+------------+-------------------------------------------------------
-------------------------------------------------------------------------------
+
6 rows in set, 1 warning (0.00 sec)

mysql>
```

count优化

如果数据量很大，在执行count操作时，是非常耗时的。

- MyISAM 引擎把一个表的总行数存在了磁盘上，因此执行 count(*) 的时候会直接返回这个 数，效率很高； 但是如果是带条件的count，MyISAM也慢。
- InnoDB 引擎就麻烦了，它执行 count(*) 的时候，需要把数据一行一行地从引擎里面读出 来，然后累积计数。

如果说要大幅度提升InnoDB表的count效率，主要的优化思路：自己计数(可以借助于redis这样的数 据库进行,但是如果是带条件的count又比较麻烦了)。

count() 是一个聚合函数，对于返回的结果集，一行行地判断，如果 count 函数的参数不是 NULL，累计值就加 1，否则不加，最后返回累计值。

主要有四种用法：count（*）、count（主键）、count（字段）、count（数字）

- count（*）：InnoDB引擎并不会把全部字段取出来，而是专门做了优化，不取值，服务层直接按行进行累加。
- count（主键）：InnoDB 引擎会遍历整张表，把每一行的 主键id值都取出来，返回给服务层。 服务层拿到主键后，直接按行进行累加(主键不可能为null)
- count（字段）：没有not null 约束 : InnoDB 引擎会遍历整张表把每一行的字段值都取出 来，返回给服务层，服务层判断是否为null，不为null，计数累加。 有not null 约束：InnoDB 引擎会遍历整张表把每一行的字段值都取出来，返 回给服务层，直接按行进行累加。
- count（数字）：InnoDB 引擎遍历整张表，但不取值。服务层对于返回的每一行，放一个数字“1”进去，直接按行进行累加。

count(字段) < count(主键 id) < count(1) ≈ count(*)，所以尽 量使用 count(*)。

```
mysql> select count(*) from student;
+----------+
| count(*) |
+----------+
|      600 |
+----------+
1 row in set (0.00 sec)

mysql> select count(1)from student;
+----------+
| count(1) |
+----------+
|      600 |
+----------+
1 row in set (0.00 sec)

mysql> select count(student_no) from student;
+-------------------+
| count(student_no) |
+-------------------+
|               600 |
+-------------------+
1 row in set (0.00 sec)

mysql> select count(email) from student;
+--------------+
| count(email) |
+--------------+
|          600 |
+--------------+
1 row in set (0.00 sec)
```

```
mysql> show profiles;
+----------+------------+------------------------------------------------------------------------------------------------------------------------------------+
| Query_ID | Duration   | Query                                                                                                                              |
+----------+------------+------------------------------------------------------------------------------------------------------------------------------------+
|        9 | 0.00256450 | select count(*) from student                                                                                                       |
|       10 | 0.00087100 | select count(1)from student                                                                                                        |
|       11 | 0.00081750 | select count(student_no) from student                                                                                              |
|       12 | 0.00182125 | select count(email) from student                                                                                                   |
|       13 | 0.00085800 | select count(*) from student                                                                                                       |
+----------+------------+------------------------------------------------------------------------------------------------------------------------------------+
13 rows in set, 1 warning (0.00 sec)
```

update优化

当执行这条sql语句时

```
update student set name="" where student_no=1;
```

会锁定id为1这一行的数据，然后事务提交之后，行锁释放

是当我们在执行如下SQL时

```
update student set name="" where name="";
```

行锁升级为了表锁。 导致该update语句的性能 大大降低。

InnoDB的行锁是针对索引加的锁，不是针对记录加的锁 ,并且该索引不能失效，否则会从行锁 升级为表锁

视图

视图（View）是一种虚拟存在的表。视图中的数据并不在数据库中实际存在，行和列数据来自定义视 图的查询中使用的表，并且是在使用视图时动态生成的。 通俗的讲，视图只保存了查询的SQL逻辑，不保存查询结果。所以我们在创建视图的时候，主要的工作 就落在创建这条SQL查询语句上。

语法

创建

```
CREATE [OR REPLACE] VIEW 视图名称[(列名列表)] AS SELECT语句 [ WITH [
CASCADED | LOCAL ] CHECK OPTION ]
```

```
mysql> create view student_names as select student_no , name from student;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

查询

查看创建视图语句：

```
SHOW CREATE VIEW 视图名称;
```

查看视图数据：

```
SELECT * FROM 视图名称 ..... ;
```

```
mysql> show create view student_names;
+---------------+-----------------------------------------------------------------
-------------------------------------------------------------------------------
---------------------------------------------+----------------------+------------
----------+
| View          | Create View

                                        | character_set_client |
collation_connection |
+---------------+-----------------------------------------------------------------
-------------------------------------------------------------------------------
---------------------------------------------+----------------------+------------
----------+
| student_names | CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL
SECURITY DEFINER VIEW `student_names` AS select `student`.`student_no` AS
`student_no`,`student`.`name` AS `name` from `student` | gbk                  |
gbk_chinese_ci       |
+---------------+-----------------------------------------------------------------
-------------------------------------------------------------------------------
---------------------------------------------+----------------------+------------
----------+
1 row in set (0.00 sec)
```

```
mysql> select * from student_names;
+--------------+--------+
| student_no   | name   |
+--------------+--------+
| 202012340101 | 关亮   |
| 202012340102 | 幸茗羽 |
| 202012340103 | 五中言 |
| 202012340104 | 邱月   |
| 202012340105 | 寿泽敬 |
| 202012340106 | 明壮冠 |
| 202012340107 | 利慧瑞 |
| 202012340108 | 良爽婵 |
| 202012340109 | 阳婉菊 |
| 202012340110 | 璩璧   |
| 202012340111 | 洪成健 |
| 202012340112 | 阳颖   |
| 202012340113 | 夔琳   |
| 202012340114 | 蒋馨凝 |
| 202012340115 | 怀秀妍 |
| 202012340116 | 邬琼   |
| 202012340117 | 寇盛   |
| 202012340118 | 谷顺山 |
| 202012340119 | 利玉雁 |
| 202012340120 | 毛昭   |
| 202012340121 | 秦翔栋 |
| 202012340122 | 夏生宁 |
| 202012340123 | 凌成振 |
| 202012340124 | 巩鸣固 |
| 202012340125 | 蓝辉   |
| 202012340126 | 湛芝   |
| 202012340127 | 爱飞   |
| 202012340128 | 言妹   |
| 202012340129 | 暴瑗   |
| 202012340130 | 门妹静 |
| 202012340131 | 东以   |
| 202012340132 | 贡瑾澜 |
...
...
...
| 202012341002 | 曹婉君 |
| 202012341003 | 柯琰勤 |
| 202012341004 | 鲁巧   |
| 202012341005 | 孙莎颖 |
| 202012341006 | 谯婵姬 |
| 202012341007 | 田秋   |
| 202012341008 | 宋宁斌 |
| 202012341009 | 徒竹   |
| 202012341010 | 甄岚   |
| 202012341011 | 慕青   |
| 202012341012 | 皇欢   |
| 202012341013 | 苍群贵 |
| 202012341014 | 郗磊信 |
| 202012341015 | 邵凤婕 |
| 202012341016 | 哈荣   |
| 202012341017 | 雍彩华 |
```

```
| 202012341018 | 漆新若 |
| 202012341019 | 史心军 |
| 202012341020 | 续青   |
| 202012341021 | 柏亨   |
| 202012341022 | 江以   |
| 202012341023 | 爱翔学 |
| 202012341024 | 贝风   |
| 202012341025 | 傅莲   |
| 202012341026 | 殳柔   |
| 202012341027 | 里康   |
| 202012341028 | 温翰   |
| 202012341029 | 胡炎兴 |
| 202012341030 | 长行   |
| 202012341031 | 言栋   |
| 202012341032 | 支雪彩 |
| 202012341033 | 孙瑞   |
| 202012341034 | 驷岩   |
| 202012341035 | 胡诚胜 |
| 202012341036 | 年宏刚 |
| 202012341037 | 潘薇月 |
| 202012341038 | 淳家   |
| 202012341039 | 年凝   |
| 202012341040 | 孟琛山 |
| 202012341041 | 卜舒叶 |
| 202012341042 | 段馥   |
| 202012341043 | 吕坚岩 |
| 202012341044 | 东宜   |
| 202012341045 | 涂群   |
| 202012341046 | 马元   |
| 202012341047 | 申成裕 |
| 202012341048 | 陈青娟 |
| 202012341049 | 程澜   |
| 202012341050 | 苏伊艺 |
| 202012341051 | 双芳   |
| 202012341052 | 南香聪 |
| 202012341053 | 辕辉宁 |
| 202012341054 | 黎昭   |
| 202012341055 | 夏飞壮 |
| 202012341056 | 俞庆   |
| 202012341057 | 元艺艳 |
| 202012341058 | 干竹   |
| 202012341059 | 姚之鸣 |
| 202012341060 | 金富   |
+--------------+--------+
600 rows in set (0.00 sec)

mysql>
```

修改

方式一：

```
CREATE [OR REPLACE] VIEW 视图名称[(列名列表)] AS SELECT语句 [ WITH [ CASCADED |
LOCAL ] CHECK OPTION ]
```

方式二：

```
ALTER VIEW 视图名称[(列名列表)] AS SELECT语句 [ WITH [ CASCADED |LOCAL ] CHECK
OPTION ]
```

```
mysql> alter view student_names as select student_no,name,sex from student;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from student_names;
+--------------+--------+-----+
| student_no   | name   | sex |
+--------------+--------+-----+
| 202012340101 | 关亮   | 男  |
| 202012340102 | 幸茗羽 | 女  |
| 202012340103 | 五中言 | 男  |
| 202012340104 | 邱月   | 女  |
| 202012340105 | 寿泽敬 | 男  |
| 202012340106 | 明壮冠 | 男  |
| 202012340107 | 利慧瑞 | 女  |
| 202012340108 | 良爽婵 | 女  |
| 202012340109 | 阳婉菊 | 女  |
| 202012340110 | 璩璧   | 女  |
| 202012340111 | 洪成健 | 男  |
| 202012340112 | 阳颖   | 女  |
| 202012340113 | 夔琳   | 女  |
| 202012340114 | 蒋馨凝 | 女  |
| 202012340115 | 怀秀妍 | 女  |
| 202012340116 | 邬琼   | 女  |
| 202012340117 | 寇盛   | 男  |
| 202012340118 | 谷顺山 | 男  |
| 202012340119 | 利玉雁 | 女  |
| 202012340120 | 毛昭   | 女  |
| 202012340121 | 秦翔栋 | 男  |
| 202012340122 | 夏生宁 | 男  |
| 202012340123 | 凌成振 | 男  |
| 202012340124 | 巩鸣固 | 男  |
| 202012340125 | 蓝辉   | 男  |
| 202012340126 | 湛芝   | 女  |
| 202012340127 | 爱飞   | 男  |
...
...
...
| 202012341015 | 邵凤婕 | 女  |
| 202012341016 | 哈荣   | 女  |
| 202012341017 | 雍彩华 | 女  |
| 202012341018 | 漆新若 | 男  |
| 202012341019 | 史心军 | 男  |
```

```
| 202012341020 | 续青   | 女   |
| 202012341021 | 柏亨   | 男   |
| 202012341022 | 江以   | 男   |
| 202012341023 | 爱翔学 | 男   |
| 202012341024 | 贝风   | 男   |
| 202012341025 | 傅莲   | 女   |
| 202012341026 | 殳柔   | 女   |
| 202012341027 | 里康   | 男   |
| 202012341028 | 温翰   | 男   |
| 202012341029 | 胡炎兴 | 男   |
| 202012341030 | 长行   | 男   |
| 202012341031 | 言栋   | 男   |
| 202012341032 | 支雪彩 | 女   |
| 202012341033 | 孙瑞   | 女   |
| 202012341034 | 驷岩   | 男   |
| 202012341035 | 胡诚胜 | 男   |
| 202012341036 | 年宏刚 | 男   |
| 202012341037 | 潘薇月 | 女   |
| 202012341038 | 淳家   | 男   |
| 202012341039 | 年凝   | 女   |
| 202012341040 | 孟琛山 | 男   |
| 202012341041 | 卜舒叶 | 女   |
| 202012341042 | 段馥   | 女   |
| 202012341043 | 吕坚岩 | 男   |
| 202012341044 | 东宜   | 女   |
| 202012341045 | 涂群   | 男   |
| 202012341046 | 马元   | 男   |
| 202012341047 | 申成裕 | 男   |
| 202012341048 | 陈青娟 | 女   |
| 202012341049 | 程澜   | 女   |
| 202012341050 | 苏伊艺 | 女   |
| 202012341051 | 双芳   | 女   |
| 202012341052 | 南香聪 | 女   |
| 202012341053 | 辕辉宁 | 男   |
| 202012341054 | 黎昭   | 女   |
| 202012341055 | 夏飞壮 | 男   |
| 202012341056 | 俞庆   | 男   |
| 202012341057 | 元艺艳 | 女   |
| 202012341058 | 干竹   | 女   |
| 202012341059 | 姚之鸣 | 男   |
| 202012341060 | 金富   | 男   |
+--------------+--------+-----+
600 rows in set (0.00 sec)

mysql>
```

删除

```
DROP VIEW [IF EXISTS] 视图名称 [,视图名称] ...
```

```
mysql> drop view if exists student_names;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from student_names;
ERROR 1146 (42S02): Table 'student1.student_names' doesn't exist
mysql>
```

检查选项

当使用WITH CHECK OPTION子句创建视图时，MySQL会通过视图检查正在更改的每个行，例如 插入，更新，删除，以使其符合视图的定义。 MySQL允许基于另一个视图创建视图，它还会检查依赖视图中的规则以保持一致性。为了确定检查的范围，mysql提供了两个选项： CASCADED 和 LOCAL ，默认值为 CASCADED 。

CASCADED

级联。 比如，v2视图是基于v1视图的，如果在v2视图创建的时候指定了检查选项为 cascaded，但是v1视图 创建时未指定检查选项。 则在执行检查时，不仅会检查v2，还会级联检查v2的关联视图v1。

LOCAL

本地。 比如，v2视图是基于v1视图的，如果在v2视图创建的时候指定了检查选项为 local ，但是v1视图创 建时未指定检查选项。 则在执行检查时，知会检查v2，不会检查v2的关联视图v1。

视图的更新

要使视图可更新，视图中的行与基础表中的行之间必须存在一对一的关系。如果视图包含以下任何一项，则该视图不可更新：

- 聚合函数或窗口函数（SUM()、 MIN()、 MAX()、 COUNT()等）
- DISTINCT
- GROUP BY
- HAVING
- UNION 或者 UNION ALL

视图作用

- 简单：视图不仅可以简化用户对数据的理解，也可以简化他们的操作。那些被经常使用的查询可以被定义为视图，从而使得用户不必为以后的操作每次指定全部的条件。
- 安全：数据库可以授权，但不能授权到数据库特定行和特定的列上。通过视图用户只能查询和修改他们所能见到的数据
- 数据独立：视图可帮助用户屏蔽真实表结构变化带来的影响。

存储过程

存储过程是事先经过编译并存储在数据库中的一段 SQL 语句的集合，调用存储过程可以简化应用开发人员的很多工作，减少数据在数据库和应用服务器之间的传输，对于提高数据处理的效率是有好处的。存储过程思想上很简单，就是数据库 SQL 语言层面的代码封装与重用。

特点

- 封装，复用：可以把某一业务SQL封装在存储过程中，需要用到 的时候直接调用即可。
- 可以接收参数，也可以返回数据 ： 再存储过程中，可以传递参数，也可以接收返回值。
- 减少网络交互，效率提升：如果涉及到多条SQL，每执行一次都是一次网络传 输。 而如果封装在存储过程中，我们只需要网络交互一次可能就可以了。

语法

创建

```
CREATE PROCEDURE 存储过程名称 ([ 参数列表 ])
BEGIN
-- SQL语句
END ;
```

```
CREATE PROCEDURE student_count()
BEGIN
select count(*) from student;
SELECT count(*) AS count_class_no_is_1001 from student WHERE class_no=1001;
END;
```

调用

```
CALL 名称 ([ 参数 ]);
```

```
mysql> call student_count();
+----------+
| count(*) |
+----------+
|      600 |
```

```
+----------+
1 row in set (0.00 sec)

+------------------------+
| count_class_no_is_1001 |
+------------------------+
|                     60 |
+------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

查看

查询指定数据库的存储过程及状态信息

```
SELECT * FROM INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_SCHEMA = 'xxx';
```

查询某个存储过程的定义

```
SHOW CREATE PROCEDURE 存储过程名称 ;
```

```
mysql> SHOW CREATE PROCEDURE student_count;
+---------------+-------------------------------------------------+------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---+----------------------+----------------------+--------------------+
| Procedure     | sql_mode                                | Create Procedure


| character_set_client | collation_connection | Database Collation |
+---------------+-------------------------------------------------+------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---+----------------------+----------------------+--------------------+
| student_count | STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION | CREATE
DEFINER=`root`@`localhost` PROCEDURE `student_count`()
BEGIN
select count(*)  from student;
SELECT count(*) AS count_class_no_is_1001 from student WHERE class_no=1001;
END | utf8mb4              | utf8mb4_0900_ai_ci   | utf8_general_ci    |
+---------------+-------------------------------------------------+------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---+----------------------+----------------------+--------------------+
1 row in set (0.00 sec)

mysql>
```

删除

```
DROP PROCEDURE [ IF EXISTS ] 存储过程名称;
```

```
mysql> drop procedure if exists student_count;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW CREATE PROCEDURE student_count;
ERROR 1305 (42000): PROCEDURE student_count does not exist
mysql>
```

在命令行中，执行创建存储过程的SQL时，需要通过关键字 delimiter 指定SQL语句的结束符。

变量

在MySQL中变量分为三种类型: 系统变量、用户定义变量、局部变量。

系统变量

系统变量是MySQL服务器提供，不是用户定义的，属于服务器层面。分为全局变量（GLOBAL）、会话变量（SESSION）。

查看所有系统变量

```
SHOW [ SESSION | GLOBAL ] VARIABLES ;
```

```
show session variables;
```

设置系统变量

```
SET [ SESSION | GLOBAL ] 系统变量名 = 值;
```

```
SET @@[SESSION | GLOBAL]系统变量名 = 值;
```

用户定义变量

用户定义变量是用户根据需要自己定义的变量，用户变量不用提前声明，在用的时候直接用 "@变量 名" 使用就可以。其作用域为当前连接。

赋值:

```
SET @var_name = expr [, @var_name = expr] ... ;
```

```
SET @var_name := expr [, @var_name := expr] ... ;
```

赋值时，可以使用 = ，也可以使用 := 。

```
SELECT @var_name := expr [, @var_name := expr] ... ;
```

```
SELECT 字段名 INTO @var_name FROM 表名;
```

```
mysql> set @a=134;
Query OK, 0 rows affected (0.00 sec)

mysql> set @b=8765;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

使用:

```
SELECT @var_name ;
```

```
mysql> select @a;
+------+
| @a   |
+------+
|  134 |
+------+
1 row in set (0.00 sec)

mysql> select @b;
+------+
| @b   |
+------+
| 8765 |
+------+
1 row in set (0.00 sec)

mysql>
```

用户定义的变量无需对其进行声明或初始化，只不过获取到的值为NULL。

局部变量

局部变量 是根据需要定义的在局部生效的变量，访问之前，需要DECLARE声明。可用作存储过程内的局部变量和输入参数，局部变量的范围是在其内声明的BEGIN ... END块。

声明：

```
DECLARE 变量名 变量类型 [DEFAULT ... ] ;
```

变量类型就是数据库字段类型：INT、BIGINT、CHAR、VARCHAR、DATE、TIME等。

赋值：

```
SET 变量名 = 值 ;
```

```
SET 变量名 := 值 ;
```

```
SELECT 字段名 INTO 变量名 FROM 表名 ... ;
```

if

if 用于做条件判断，具体的语法结构为：

```
IF 条件1 THEN
.....
ELSEIF 条件2 THEN -- 可选
.....
ELSE -- 可选
.....
END IF;
```

在if条件判断的结构中，ELSE IF 结构可以有多个，也可以没有。 ELSE结构可以有，也可以没有。

根据定义的分数score变量，判定当前分数对应的分数等级。

- score >= 85分，等级为优秀。
- score >= 60分 且 score < 85分，等级为及格。
- score < 60分，等级为不及格。

创建：

```
CREATE PROCEDURE p3 () BEGIN
    DECLARE
        score INT DEFAULT 58;
    DECLARE
        result VARCHAR ( 10 );
    IF
        score >= 85 THEN

            SET result := '优秀';

        ELSEIF score >= 60 THEN

        SET result := '及格';
        ELSE
            SET result := '不及格';

    END IF;
    SELECT
    result;
END;
```

运行：

```
call p3();
```

```
mysql> call p3();
+--------+
| result |
+--------+
| 不及格 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

参数

参数的类型，主要分为以下三种：IN、OUT、INOUT。 具体的含义如下：

- IN：该类参数作为输入，也就是需要调用时传入值
- OUT：该类参数作为输出，也就是该参数可以作为返回值
- INOUT：既可以作为输入参数，也可以作为输出参数

语法：

```
CREATE PROCEDURE 存储过程名称 ([ IN/OUT/INOUT 参数名 参数类型 ])
BEGIN
-- SQL语句
END ;
```

根据传入参数score，判定当前分数对应的分数等级，并返回。

- score >= 85分，等级为优秀。
- score >= 60分 且 score < 85分，等级为及格。
- score < 60分，等级为不及格。

```
CREATE PROCEDURE p4 (
    IN score INT,
    OUT result VARCHAR ( 10 )) BEGIN
    IF
        score >= 85 THEN

            SET result := '优秀';

        ELSEIF score >= 60 THEN

        SET result := '及格';
        ELSE
            SET result := '不及格';

    END IF;
END;
```

运行：

```
call p4(18, @result);
```

```
select @result;
```

```
mysql> call p4(18,@result);
Query OK, 0 rows affected (0.00 sec)

mysql> select @result;
+---------+
| @result |
+---------+
| 不及格  |
+---------+
1 row in set (0.00 sec)

mysql> call p4(86,@result);
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select @result;
+---------+
| @result |
+---------+
| 优秀    |
+---------+
1 row in set (0.00 sec)

mysql> call p4(69,@result);
Query OK, 0 rows affected (0.00 sec)

mysql> select @result;
+---------+
| @result |
+---------+
| 及格    |
+---------+
1 row in set (0.00 sec)

mysql>
```

case

当case_value的值为 when_value1时，执行statement_list1，当值为 when_value2时， 执行statement_list2， 否则就执行 statement_list

```
CASE case_value
    WHEN when_value1 THEN statement_list1
    [ WHEN when_value2 THEN statement_list2] ...
    [ ELSE statement_list ]
END CASE;
```

当条件search_condition1成立时，执行statement_list1，当条件search_condition2成 立时，执行statement_list2， 否则就执行 statement_list

```
CASE
    WHEN search_condition1 THEN statement_list1
    [WHEN search_condition2 THEN statement_list2] ...
    [ELSE statement_list]
END CASE;
```

while

while 循环是有条件的循环控制语句。满足条件后，再执行循环体中的SQL语句。具体语法为：

```
WHILE 条件 DO
    SQL逻辑...
END WHILE;
```

计算从1累加到n的值，n为传入的参数值。

```
CREATE PROCEDURE p7 ( IN n INT ) BEGIN
    DECLARE
        total INT DEFAULT 0;
    WHILE
            n > 0 DO

            SET total := total + n;

        SET n := n - 1;

    END WHILE;
    SELECT
        total;

END;
```

运行:

```
call p7 (100);
```

```
mysql> call p7 (100);
+-------+
| total |
+-------+
|  5050 |
+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call p7 (500);
+--------+
| total  |
+--------+
| 125250 |
+--------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call p7 (5000);
+----------+
| total    |
+----------+
```

```
| 12502500 |
+----------+
1 row in set (0.11 sec)

Query OK, 0 rows affected (0.11 sec)

mysql> call p7 (50000);
+------------+
| total      |
+------------+
| 1250025000 |
+------------+
1 row in set (0.96 sec)

Query OK, 0 rows affected (0.96 sec)

mysql>
```

repeat

repeat是有条件的循环控制语句, 当满足until声明的条件的时候，则退出循环 。具体语法为：

```
REPEAT
    SQL逻辑...
    UNTIL 条件
END REPEAT;
```

loop

LOOP 实现简单的循环，如果不在SQL逻辑中增加退出循环的条件，可以用其来实现简单的死循环。
LOOP可以配合一下两个语句使用：

- LEAVE ： 配合循环使用，退出循环。
- ITERATE：必须用在循环中，作用是跳过当前循环剩下的语句，直接进入下一次循环。

```
[begin_label:] LOOP
    SQL逻辑...
END LOOP [end_label];
```

退出指定标记的循环体：

```
LEAVE label;
```

直接进入下一次循环：

```
ITERATE label;
```

计算从1累加到n的值，n为传入的参数值。

```
CREATE PROCEDURE p9 ( IN n INT ) BEGIN
    DECLARE
        total INT DEFAULT 0;
    sum :
    LOOP
        IF
            n <= 0 THEN
                LEAVE sum;

        END IF;

        SET total := total + n;

        SET n := n - 1;

    END LOOP sum;
    SELECT
    total;
END;
```

运行:

```
call p9(100);
```

```
mysql> call p9(100);
+-------+
| total |
+-------+
|  5050 |
+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call p9(500);
+--------+
| total  |
+--------+
| 125250 |
+--------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call p9(5000);
+----------+
```

```
| total    |
+----------+
| 12502500 |
+----------+
1 row in set (0.10 sec)

Query OK, 0 rows affected (0.10 sec)

mysql> call p9(50000);
+------------+
| total      |
+------------+
| 1250025000 |
+------------+
1 row in set (1.00 sec)

Query OK, 0 rows affected (1.00 sec)

mysql>
```

游标

游标（CURSOR）是用来存储查询结果集的数据类型，在存储过程和函数中可以使用游标对结果集进 行循环的处理。游标的使用包括游标的声明、OPEN、FETCH 和 CLOSE

声明游标

```
DECLARE 游标名称 CURSOR FOR 查询语句 ;
```

打开游标

```
OPEN 游标名称 ;
```

获取游标记录

```
FETCH 游标名称 INTO 变量 [, 变量 ] ;
```

关闭游标

```
CLOSE 游标名称 ;
```

条件处理程序

条件处理程序（Handler）可以用来定义在流程控制结构执行过程中遇到问题时相应的处理步骤。

语法：

```
DECLARE handler_action HANDLER FOR condition_value [, condition_value]
... statement ;
handler_action 的取值:
    CONTINUE: 继续执行当前程序
    EXIT: 终止执行当前程序
condition_value 的取值:
    SQLSTATE sqlstate_value: 状态码，如 02000
    SQLWARNING: 所有以01开头的SQLSTATE代码的简写
    NOT FOUND: 所有以02开头的SQLSTATE代码的简写
    SQLEXCEPTION: 所有没有被SQLWARNING 或 NOT FOUND捕获的SQLSTATE代码的简写
```

存储函数

存储函数是有返回值的存储过程，存储函数的参数只能是IN类型的。

语法：

```
CREATE FUNCTION 存储函数名称 ([ 参数列表 ])
RETURNS type [characteristic ...]
BEGIN
    -- SQL语句
    RETURN ...;
END ;
```

characteristic说明：

- DETERMINISTIC：相同的输入参数总是产生相同的结果
- NO SQL ：不包含 SQL 语句。
- READS SQL DATA：包含读取数据的语句，但不包含写入数据的语句。

计算从1累加到n的值，n为传入的参数值。

```
CREATE FUNCTION fun1 ( n INT ) RETURNS INT DETERMINISTIC BEGIN
    DECLARE
        total INT DEFAULT 0;
    WHILE
            n > 0 DO

            SET total := total + n;

        SET n := n - 1;

    END WHILE;
RETURN total;
END;
```

运行：

```
select fun1(100);
```

```
mysql> select fun1(100);
+-----------+
| fun1(100) |
+-----------+
|      5050 |
+-----------+
1 row in set (0.00 sec)

mysql> select fun1(500);
+-----------+
| fun1(500) |
+-----------+
|    125250 |
+-----------+
1 row in set (0.01 sec)

mysql> select fun1(5000);
+------------+
| fun1(5000) |
+------------+
|   12502500 |
+------------+
1 row in set (0.07 sec)

mysql> select fun1(50000);
+-------------+
| fun1(50000) |
+-------------+
|  1250025000 |
+-------------+
1 row in set (0.76 sec)

mysql>
```

触发器

触发器是与表有关的数据库对象，指在insert/update/delete之前(BEFORE)或之后(AFTER)，触 发并执行触发器中定义的SQL语句集合。触发器的这种特性可以协助应用在数据库端确保数据的完整性，日志记录，数据校验等操作 。

使用别名OLD和NEW来引用触发器中发生变化的记录内容，这与其他的数据库是相似的。现在触发器还只支持行级触发，不支持语句级触发。

- INSERT 型触发器：NEW 表示将要或者已经新增的数据
- UPDATE 型触发器 ： OLD 表示修改之前的数据 , NEW 表示将要或已经修改后的数据
- DELETE 型触发器：OLD 表示将要或者已经删除的数据

语法

创建

```
CREATE TRIGGER trigger_name
BEFORE/AFTER INSERT/UPDATE/DELETE
ON tbl_name FOR EACH ROW -- 行级触发器
BEGIN
    trigger_stmt ;
END;
```

查看

```
SHOW TRIGGERS ;
```

删除

```
DROP TRIGGER [schema_name.]trigger_name ;
```

锁

锁是计算机协调多个进程或线程并发访问某一资源的机制。在数据库中，除传统的计算资源（CPU、RAM、I/O）的争用以外，数据也是一种供许多用户共享的资源。如何保证数据并发访问的一致性、有效性是所有数据库必须解决的一个问题，锁冲突也是影响数据库并发访问性能的一个重要因素。从这个角度来说，锁对数据库而言显得尤其重要，也更加复杂。

MySQL中的锁，按照锁的粒度分，分为以下三类：

- 全局锁：锁定数据库中的所有表。
- 表级锁：每次操作锁住整张表。
- 行级锁：每次操作锁住对应的行数据。

全局锁

全局锁就是对整个数据库实例加锁，加锁后整个实例就处于只读状态，后续的DML的写语句，DDL语句，已经更新操作的事务提交语句都将被阻塞。

其典型的使用场景是做全库的逻辑备份，对所有的表进行锁定，从而获取一致性视图，保证数据的完整性。

假设在数据库中存在这样三张表: tb_stock 库存表，tb_order 订单表，tb_orderlog 订单日志表。

- 在进行数据备份时，先备份了tb_stock库存表。
- 然后接下来，在业务系统中，执行了下单操作，扣减库存，生成订单（更新tb_stock表，插入tb_order表）。
- 然后再执行备份 tb_order表的逻辑。
- 业务中执行插入订单日志操作。
- 最后，又备份了tb_orderlog表。
- 此时备份出来的数据，是存在问题的。因为备份出来的数据，tb_stock表与tb_order表的数据不一致(有最新操作的订单信息,但是库存数没减)。

对数据库进行进行逻辑备份之前，先对整个数据库加上全局锁，一旦加了全局锁之后，其他的DDL、DML全部都处于阻塞状态，但是可以执行DQL语句，也就是处于只读状态，而数据备份就是查询操作。那么数据在进行逻辑备份的过程中，数据库中的数据就是不会发生变化的，这样就保证了数据的一致性和完整性

语法

加全局锁：

```
flush tables with read lock ;
```

```
mysql> use student1;
Database changed
mysql> show tables;
+-------------------------+
| Tables_in_student1      |
+-------------------------+
| administrators          |
| administrators_password |
| class                   |
| course                  |
| forum                   |
| login_log               |
| news                    |
| score                   |
| student                 |
| student_password        |
| teach                   |
| teacher                 |
| teacher_password        |
+-------------------------+
13 rows in set (0.02 sec)

mysql> flush tables with read lock;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

数据备份：

```
mysqldump -uroot -pxxx student1 > student1.sql
```

释放锁：

```
unlock tables;
```

特点

数据库中加全局锁，是一个比较重的操作，存在以下问题：

- 如果在主库上备份，那么在备份期间都不能执行更新，业务基本上就得停摆。
- 如果在从库上备份，那么在备份期间从库不能执行主库同步过来的二进制日志（binlog），会导 致主从延迟。

表级锁

表级锁，每次操作锁住整张表。锁定粒度大，发生锁冲突的概率最高，并发度最低。应用在MyISAM、InnoDB、BDB等存储引擎中。

对于表级锁，主要分为以下三类：

- 表锁
- 元数据锁（meta data lock，MDL）
- 意向锁

表锁

对于表锁，分为两类：

- 表共享读锁（read lock）
- 表独占写锁（write lock）

语法：

- 加锁：lock tables 表名... read/write。
- 释放锁：unlock tables / 客户端断开连接 。

```
mysql> lock tables student read;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

- 读锁：客户端一，对指定表加了读锁，不会影响客户端二的读，但是会阻塞客户端二的写。
- 写锁：客户端一，对指定表加了写锁，会阻塞客户端二的读和写。

读锁不会阻塞其他客户端的读，但是会阻塞写。写锁既会阻塞其他客户端的读，又会阻塞其他客户端的写。

元数据锁

meta data lock , 元数据锁，简写MDL。

MDL加锁过程是系统自动控制，无需显式使用，在访问一张表的时候会自动加上。MDL锁主要作用是维护表元数据的数据一致性，在表上有活动事务的时候，不可以对元数据进行写入操作。为了避免DML与DDL冲突，保证读写的正确性。

对应SQL	锁类型	说明
lock tables xxx read / write	SHARED_READ_ONLY / SHARED_NO_READ_WRITE	
select 、select ... lock in share mode	SHARED_READ	与SHARED_READ、SHARED_WRITE兼容，与EXCLUSIVE互斥
insert 、update、delete、select ... for update	SHARED_WRITE	与SHARED_READ、SHARED_WRITE兼容，与EXCLUSIVE互斥
alter table ...	EXCLUSIVE	与其他的MDL都互斥

意向锁

为了避免DML在执行时，加的行锁与表锁的冲突，在InnoDB中引入了意向锁，使得表锁不用检查每行数据是否加锁，使用意向锁来减少表锁的检查

- 意向共享锁(IS): 由语句select ... lock in share mode添加 。 与 表锁共享锁 (read)兼容，与表锁排他锁(write)互斥。
- 意向排他锁(IX): 由insert、update、delete、select...for update添加 。与表锁共 享锁(read)及排他锁(write)都互斥，意向锁之间不会互斥。

一旦事务提交了，意向共享锁、意向排他锁，都会自动释放。

可以通过以下SQL，查看意向锁及行锁的加锁情况：

```
select object_schema,object_name,index_name,lock_type,lock_mode,lock_data from
performance_schema.data_locks;
```

行级锁

行级锁，每次操作锁住对应的行数据。锁定粒度最小，发生锁冲突的概率最低，并发度最高。应用在InnoDB存储引擎中。

InnoDB的数据是基于索引组织的，行锁是通过对索引上的索引项加锁来实现的，而不是对记录加的锁。对于行级锁，主要分为以下三类：

- 行锁（Record Lock）：锁定单个行记录的锁，防止其他事务对此行进行update和delete。在RC、RR隔离级别下都支持。
- 间隙锁（Gap Lock）：锁定索引记录间隙（不含该记录），确保索引记录间隙不变，防止其他事务在这个间隙进行insert，产生幻读。在RR隔离级别下都支持。
- 临键锁（Next-Key Lock）：行锁和间隙锁组合，同时锁住数据，并锁住数据前面的间隙Gap。 在RR隔离级别下支持。

行锁

InnoDB实现了以下两种类型的行锁：

共享锁（S）：允许一个事务去读一行，阻止其他事务获得相同数据集的排它锁。

排他锁（X）：允许获取排他锁的事务更新数据，阻止其他事务获得相同数据集的共享锁和排他锁。

SQL	行锁类型	说明
INSERT ...	排他锁	自动加锁
UPDATE ...	排他锁	自动加锁
DELETE ...	排他锁	自动加锁
SELECT（正常）	不加任何锁	
SELECT ... LOCK IN SHARE MODE	共享锁	需要手动在SELECT之后加LOCK IN SHARE MODE
SELECT ... FOR UPDATE	排他锁	需要手动在SELECT之后加FOR UPDATE

默认情况下，InnoDB在 REPEATABLE READ事务隔离级别运行，InnoDB使用 next-key 锁进行搜 索和索引扫描，以防止幻读。

- 针对唯一索引进行检索时，对已存在的记录进行等值匹配时，将会自动优化为行锁。
- InnoDB的行锁是针对于索引加的锁，不通过索引条件检索数据，那么InnoDB将对表中的所有记 录加锁，此时就会升级为表锁。

可以通过以下SQL，查看意向锁及行锁的加锁情况：

```
select object_schema,object_name,index_name,lock_type,lock_mode,lock_data from
performance_schema.data_locks;
```

间隙锁和临键锁

默认情况下，InnoDB在 REPEATABLE READ事务隔离级别运行，InnoDB使用 next-key 锁进行搜索和索引扫描，以防止幻读。

- 索引上的等值查询(唯一索引)，给不存在的记录加锁时, 优化为间隙锁 。
- 索引上的等值查询(非唯一普通索引)，向右遍历时最后一个值不满足查询需求时，next-key lock 退化为间隙锁。
- 索引上的范围查询(唯一索引)会访问到不满足条件的第一个值为止。

间隙锁唯一目的是防止其他事务插入间隙。间隙锁可以共存，一个事务采用的间隙锁不会 阻止另一个事务在同一间隙上采用间隙锁。

InnoDB引擎

逻辑存储结构

表空间

表空间是InnoDB存储引擎逻辑结构的最高层， 如果用户启用了参数 innodb_file_per_table(在 8.0版本中默认开启)，则每张表都会有一个表空间（xxx.ibd），一个mysql实例可以对应多个表空 间，用于存储记录、索引等数据。

段

段，分为数据段（Leaf node segment）、索引段（Non-leaf node segment）、回滚段 （Rollback segment），InnoDB是索引组织表，数据段就是B+树的叶子节点， 索引段即为B+树的 非叶子节点。段用来管理多个Extent（区）。

区

区，表空间的单元结构，每个区的大小为1M。 默认情况下， InnoDB存储引擎页大小为16K， 即一 个区中一共有64个连续的页。

页

页，是InnoDB 存储引擎磁盘管理的最小单元，每个页的大小默认为 16KB。为了保证页的连续性，InnoDB 存储引擎每次从磁盘申请 4-5 个区。

行

行，InnoDB 存储引擎数据是按行进行存放的。

在行中，默认有两个隐藏字段：

- Trx_id：每次对某条记录进行改动时，都会把对应的事务id赋值给trx_id隐藏列。
- Roll_pointer：每次对某条引记录进行改动时，都会把旧的版本写入到undo日志中，然后这个 隐藏列就相当于一个指针，可以通过它来找到该记录修改前的信息。

架构

内存结构

内存四大块：Buffer Pool、Change Buffer、Adaptive Hash Index、Log Buffer。

Buffer Pool：

InnoDB存储引擎基于磁盘文件存储，访问物理硬盘和在内存中进行访问，速度相差很大，为了尽可能弥补这两者之间的I/O效率的差值，就需要把经常使用的数据加载到缓冲池中，避免每次访问都进行磁盘I/O。 在InnoDB的缓冲池中不仅缓存了索引页和数据页，还包含了undo页、插入缓存、自适应哈希索引以及 InnoDB的锁信息等等。

缓冲池 Buffer Pool，是主内存中的一个区域，里面可以缓存磁盘上经常操作的真实数据，在执行增 删改查操作时，先操作缓冲池中的数据（若缓冲池没有数据，则从磁盘加载并缓存），然后再以一定频 率刷新到磁盘，从而减少磁盘IO，加快处理速度。

缓冲池以Page页为单位，底层采用链表数据结构管理Page。根据状态，将Page分为三种类型：

- free page：空闲page，未被使用。
- clean page：被使用page，数据没有被修改过。
- dirty page：脏页，被使用page，数据被修改过，也中数据与磁盘的数据产生了不一致。

Change Buffer：

Change Buffer，更改缓冲区（针对于非唯一二级索引页），在执行DML语句时，如果这些数据Page 没有在Buffer Pool中，不会直接操作磁盘，而会将数据变更存在更改缓冲区 Change Buffer 中，在未来数据被读取时，再将数据合并恢复到Buffer Pool中，再将合并后的数据刷新到磁盘中。

与聚集索引不同，二级索引通常是非唯一的，并且以相对随机的顺序插入二级索引。同样，删除和更新可能会影响索引树中不相邻的二级索引页，如果每一次都操作磁盘，会造成大量的磁盘IO。有了ChangeBuffer之后，我们可以在缓冲池中进行合并处理，减少磁盘IO。

Adaptive Hash Index：

自适应hash索引，用于优化对Buffer Pool数据的查询。MySQL的innoDB引擎中虽然没有直接支持hash索引，但是给我们提供了一个功能就是这个自适应hash索引。因为前面我们讲到过，hash索引在进行等值匹配时，一般性能是要高于B+树的，因为hash索引一般只需要一次IO即可，而B+树，可能需要几次匹配，所以hash索引的效率要高，但是hash索引又不适合做范围查询、模糊匹配等。

InnoDB存储引擎会监控对表上各索引页的查询，如果观察到在特定的条件下hash索引可以提升速度，则建立hash索引，称之为自适应hash索引。 自适应哈希索引，无需人工干预，是系统根据情况自动完成。

Log Buffer：

Log Buffer：日志缓冲区，用来保存要写入到磁盘中的log日志数据（redo log 、undo log）， 默认大小为 16MB，日志缓冲区的日志会定期刷新到磁盘中。如果需要更新、插入或删除许多行的事 务，增加日志缓冲区的大小可以节省磁盘 I/O。

innodb_log_buffer_size：缓冲区大小

innodb_flush_log_at_trx_commit：日志刷新到磁盘时机，取值主要包含以下三个：

- 0: 每秒将日志写入并刷新到磁盘一次。
- 1: 日志在每次事务提交时写入并刷新到磁盘，默认值。
- 2: 日志在每次事务提交后写入，并每秒刷新到磁盘一次。

磁盘结构

System Tablespace：

系统表空间是更改缓冲区的存储区域。如果表是在系统表空间而不是每个表文件或通用表空间中创建的，它也可能包含表和索引数据。

系统表空间，默认的文件名叫 ibdata1。

File-Per-Table Tablespaces：

如果开启了innodb_file_per_table开关 ，则每个表的文件表空间包含单个InnoDB表的数据和索 引 ，并存储在文件系统上的单个数据文件中。

开关参数：innodb_file_per_table

General Tablespaces：

通用表空间，需要通过 CREATE TABLESPACE 语法创建通用表空间，在创建表时，可以指定该表空 间。

创建表空间：

```
CREATE TABLESPACE ts_name ADD DATAFILE 'file_name' ENGINE = engine_name;
```

Undo Tablespaces：

撤销表空间，MySQL实例在初始化时会自动创建两个默认的undo表空间（初始大小16M），用于存储undo log日志。

Temporary Tablespaces：

InnoDB 使用会话临时表空间和全局临时表空间。存储用户创建的临时表等数据。

Doublewrite Buffer Files：

双写缓冲区，innoDB引擎将数据页从Buffer Pool刷新到磁盘前，先将数据页写入双写缓冲区文件中，便于系统异常时恢复数据。

Redo Log：

重做日志，是用来实现事务的持久性。该日志文件由两部分组成：重做日志缓冲（redo log buffer）以及重做日志文件（redo log）,前者是在内存中，后者在磁盘中。当事务提交之后会把所 有修改信息都会存到该日志中, 用于在刷新脏页到磁盘时,发生错误时, 进行数据恢复使用。

后台线程

在InnoDB的后台线程中，分为4类，分别是：Master Thread 、IO Thread、Purge Thread、 Page Cleaner Thread。

Master Thread：

核心后台线程，负责调度其他线程，还负责将缓冲池中的数据异步刷新到磁盘中, 保持数据的一致性，还包括脏页的刷新、合并插入缓存、undo页的回收 。

IO Thread：

在InnoDB存储引擎中大量使用了AIO来处理IO请求, 这样可以极大地提高数据库的性能，而IO Thread主要负责这些IO请求的回调。

线程类型	默认个数	职责
Read thread	4	负责读操作
Write thread	4	负责写操作
Log thread	1	负责将日志缓冲区刷新到磁盘
Insert buffer thread	1	负责将写缓冲区内容刷新到磁盘

Purge Thread：

主要用于回收事务已经提交了的undo log，在事务提交之后，undo log可能不用了，就用它来回收。

Page Cleaner Thread：

协助 Master Thread 刷新脏页到磁盘的线程，它可以减轻 Master Thread 的工作压力，减少阻塞。

事务原理

事务 是一组操作的集合，它是一个不可分割的工作单位，事务会把所有的操作作为一个整体一起向系 统提交或撤销操作请求，即这些操作要么同时成功，要么同时失败。

- 原子性（Atomicity）：事务是不可分割的最小操作单元，要么全部成功，要么全部失败。
- 一致性（Consistency）：事务完成时，必须使所有的数据都保持一致状态。
- 隔离性（Isolation）：数据库系统提供的隔离机制，保证事务在不受外部并发操作影响的独立环 境下运行。
- 持久性（Durability）：事务一旦提交或回滚，它对数据库中的数据的改变就是永久的。

而对于这四大特性，实际上分为两个部分。 其中的原子性、一致性、持久性，实际上是由InnoDB中的两份日志来保证的，一份是redo log日志，一份是undo log日志。 而隔离性是通过数据库的锁， 加上MVCC来保证的。

redo log

重做日志，记录的是事务提交时数据页的物理修改，是用来实现事务的持久性。 该日志文件由两部分组成：

- 重做日志缓冲（redo log buffer）
- 重做日志文件（redo log file）

前者是在内存中，后者在磁盘中。

当事务提交之后会把所有修改信息都存到该日志文件中, 用于在刷新脏页到磁盘,发生错误时, 进行数据恢复使用。

当对缓冲区的数据进行增删改之后，会首先将操作的数据页的变化，记录在redo log buffer中。在事务提交时，会将redo log buffer中的数据刷新到redo log磁盘文件中。 过一段时间之后，如果刷新缓冲区的脏页到磁盘时，发生错误，此时就可以借助于redo log进行数据 恢复，这样就保证了事务的持久性。而如果脏页成功刷新到磁盘，或者涉及到的数据已经落盘，此 时redolog就没有作用了，就可以删除了，所以存在的两个redolog文件是循环写的。

undo log

回滚日志，用于记录数据被修改前的信息 , 作用包含两个 :

- 提供回滚(保证事务的原子性)
- MVCC(多版本并发控制)

undo log和redo log记录物理日志不一样，它是逻辑日志。可以认为当delete一条记录时，undo log中会记录一条对应的insert记录，反之亦然，当update一条记录时，它记录一条对应相反的 update记录。当执行rollback时，就可以从undo log中的逻辑记录读取到相应的内容并进行回滚。

Undo log销毁：undo log在事务执行时产生，事务提交时，并不会立即删除undo log，因为这些 日志可能还用于MVCC。

Undo log存储：undo log采用段的方式进行管理和记录

MVCC

基本概念

当前读：

读取的是记录的最新版本，读取时还要保证其他并发事务不能修改当前记录，会对读取的记录进行加锁。对于我们日常的操作，如：select ... lock in share mode(共享锁)，select ... for update、update、insert、delete(排他锁)都是一种当前读。

快照读：

简单的select（不加锁）就是快照读，快照读，读取的是记录数据的可见版本，有可能是历史数据， 不加锁，是非阻塞读。

- Read Committed：每次select，都生成一个快照读。
- Repeatable Read：开启事务后第一个select语句才是快照读的地方。
- Serializable：快照读会退化为当前读。

MVCC：

全称 Multi-Version Concurrency Control，多版本并发控制。指维护一个数据的多个版本， 使得读写操作没有冲突，快照读为MySQL实现MVCC提供了一个非阻塞读功能。MVCC的具体实现，还需 要依赖于数据库记录中的三个隐式字段、undo log日志、readView。

隐藏字段

隐藏字段	含义
DB_TRX_ID	最近修改事务ID，记录插入这条记录或最后一次修改该记录的事务ID。
DB_ROLL_PTR	回滚指针，指向这条记录的上一个版本，用于配合undo log，指向上一个版本。
DB_ROW_ID	隐藏主键，如果表结构没有指定主键，将会生成该隐藏字段。如果有主键，就不会生成此字段

undolog

回滚日志，在insert、update、delete的时候产生的便于数据回滚的日志。 当insert的时候，产生的undo log日志只在回滚时需要，在事务提交后，可被立即删除。 而update、delete的时候，产生的undo log日志不仅在回滚时需要，在快照读时也需要，不会立即 被删除。

版本链

不同事务或相同事务对同一条记录进行修改，会导致该记录的undolog生成一条 记录版本链表，链表的头部是最新的旧记录，链表尾部是最早的旧记录。

readview

ReadView（读视图）是 快照读 SQL执行时MVCC提取数据的依据，记录并维护系统当前活跃的事务（未提交的）id。

ReadView中包含四个核心字段：

字段	含义
m_ids	当前活跃的事务ID集合
min_trx_id	最小活跃事务ID
max_trx_id	预分配事务ID，当前最大事务ID+1（因为事务ID是自增的）
creator_trx_id	ReadView创建者的事务ID

了版本链数据的访问规则：

条件	是否可以访问	说明
trx_id == creator_trx_id	可以访问该版本	成立，说明数据是当前这个事 务更改的。
trx_id < min_trx_id	可以访问该版本	成立，说明数据已经提交了。
trx_id > max_trx_id	不可以访问该版本	成立，说明该事务是在ReadView生成后才开启。
min_trx_id <= trx_id <= max_trx_id	如果trx_id不在m_ids中， 可以访问该版本，否则，不可以访问该版本	成立，说明数据已经提交。

不同的隔离级别，生成ReadView的时机不同：

- READ COMMITTED ： 在事务中每一次执行快照读时生成ReadView。
- REPEATABLE READ：仅在事务中第一次执行快照读时生成ReadView，后续复用该ReadView。

MySQL管理

系统数据库

MySQL自带了一下四个数据库：

- mysql：存储MySQL服务器正常运行所需要的各种信息 （时区、主从、用 户、权限等）
- information_schema：提供了访问数据库元数据的各种表和视图，包含数据库、表、字段类 型及访问权限等
- performance_schema：为MySQL服务器运行时状态提供了一个底层监控功能，主要用于收集数据库服务器性能参数
- sys：包含了一系列方便 DBA 和开发人员利用 performance_schema 性能数据库进行性能调优和诊断的视图

常用工具

mysql

语法：

```
mysql [options] [database]
```

选项：

- -u, --user=name #指定用户名
- -p, --password[=name] #指定密码
- -h, --host=name #指定服务器IP或域名

- -P, --port=port #指定连接端口
- -e, --execute=name #执行SQL语句并退出

```
C:\Users\mao>mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.0.27 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
mysql -uroot -pxxx student1 -e"select * from student"
```

mysqladmin

mysqladmin 是一个执行管理操作的客户端程序。可以用它来检查服务器的配置和当前状态、创建并删除数据库等。

```
C:\Users\mao>mysqladmin --help
mysqladmin  Ver 8.0.27 for Win64 on x86_64 (MySQL Community Server - GPL)
Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Administration program for the mysqld daemon.
Usage: mysqladmin [OPTIONS] command command....
  --bind-address=name IP address to bind to.
  -c, --count=#       Number of iterations to make. This works with -i
                      (--sleep) only.
  -#, --debug[=#]     This is a non-debug version. Catch this and exit.
  --debug-check       This is a non-debug version. Catch this and exit.
  --debug-info        This is a non-debug version. Catch this and exit.
  -f, --force         Don't ask for confirmation on drop database; with
                      multiple commands, continue even if an error occurs.
  -C, --compress      Use compression in server/client protocol.
  --character-sets-dir=name
                      Directory for character set files.
  --default-character-set=name
```

```
                      Set the default character set.
-?, --help            Display this help and exit.
-h, --host=name       Connect to host.
-b, --no-beep         Turn off beep on error.
-p, --password[=name]
                      Password to use when connecting to server. If password is
                      not given it's asked from the tty.
-,, --password1[=name]
                      Password for first factor authentication plugin.
-,, --password2[=name]
                      Password for second factor authentication plugin.
-,, --password3[=name]
                      Password for third factor authentication plugin.
-W, --pipe            Use named pipes to connect to server.
-P, --port=#          Port number to use for connection or 0 for default to, in
                      order of preference, my.cnf, $MYSQL_TCP_PORT,
                      /etc/services, built-in default (3306).
--protocol=name       The protocol to use for connection (tcp, socket, pipe,
                      memory).
-r, --relative        Show difference between current and previous values when
                      used with -i. Currently only works with extended-status.
--shared-memory-base-name=name
                      Base name of shared memory.
-s, --silent          Silently exit if one can't connect to server.
-S, --socket=name     The socket file to use for connection.
-i, --sleep=#         Execute commands repeatedly with a sleep between.
--ssl-mode=name       SSL connection mode.
--ssl-ca=name         CA file in PEM format.
--ssl-capath=name     CA directory.
--ssl-cert=name       X509 cert in PEM format.
--ssl-cipher=name     SSL cipher to use.
--ssl-key=name        X509 key in PEM format.
--ssl-crl=name        Certificate revocation list.
--ssl-crlpath=name    Certificate revocation list path.
--tls-version=name    TLS version to use, permitted values are: TLSv1, TLSv1.1,
                      TLSv1.2, TLSv1.3
--ssl-fips-mode=name
                      SSL FIPS mode (applies only for OpenSSL); permitted
                      values are: OFF, ON, STRICT
--tls-ciphersuites=name
                      TLS v1.3 cipher to use.
--server-public-key-path=name
                      File path to the server public RSA key in PEM format.
--get-server-public-key
                      Get server public key
-u, --user=name       User for login if not current user.
-v, --verbose         Write more information.
-V, --version         Output version information and exit.
-E, --vertical        Print output vertically. Is similar to --relative, but
                      prints output vertically.
-w, --wait[=#]        Wait and retry if connection is down.
--connect-timeout=#
--shutdown-timeout=#
--plugin-dir=name     Directory for client-side plugins.
--default-auth=name   Default authentication client-side plugin to use.
--enable-cleartext-plugin
                      Enable/disable the clear text authentication plugin.
--show-warnings       Show warnings after execution
```

```
  --compression-algorithms=name
                      Use compression algorithm in server/client protocol.
                      Valid values are any combination of
                      'zstd','zlib','uncompressed'.
  --zstd-compression-level=#
                      Use this compression level in the client/server protocol,
                      in case --compression-algorithms=zstd. Valid range is
                      between 1 and 22, inclusive. Default is 3.

Variables (--variable-name=value)
and boolean options {FALSE|TRUE}  Value (after reading options)
--------------------------------- ----------------------------------------
bind-address                      (No default value)
count                             0
force                             FALSE
compress                          FALSE
character-sets-dir                (No default value)
default-character-set             auto
host                              (No default value)
no-beep                           FALSE
port                              0
relative                          FALSE
shared-memory-base-name           (No default value)
socket                            (No default value)
sleep                             0
ssl-ca                            (No default value)
ssl-capath                        (No default value)
ssl-cert                          (No default value)
ssl-cipher                        (No default value)
ssl-key                           (No default value)
ssl-crl                           (No default value)
ssl-crlpath                       (No default value)
tls-version                       (No default value)
tls-ciphersuites                  (No default value)
server-public-key-path            (No default value)
get-server-public-key             FALSE
user                              (No default value)
verbose                           FALSE
vertical                          FALSE
connect-timeout                   43200
shutdown-timeout                  3600
plugin-dir                        (No default value)
default-auth                      (No default value)
enable-cleartext-plugin           FALSE
show-warnings                     FALSE
compression-algorithms            (No default value)
zstd-compression-level            3

Default options are read from the following files in the given order:
C:\WINDOWS\my.ini C:\WINDOWS\my.cnf C:\my.ini C:\my.cnf C:\Program
Files\MySQL\MySQL Server 8.0\my.ini C:\Program Files\MySQL\MySQL Server
8.0\my.cnf
The following groups are read: mysqladmin client
The following options may be given as the first argument:
--print-defaults        Print the program argument list and exit.
--no-defaults           Don't read default options from any option file,
                        except for login file.
--defaults-file=#       Only read default options from the given file #.
```

```
--defaults-extra-file=# Read this file after the global files are read.
--defaults-group-suffix=#
                        Also read groups with concat(group, suffix)
--login-path=#          Read this path from the login file.

Where command is a one or more of: (Commands may be shortened)
  create databasename   Create a new database
  debug                 Instruct server to write debug information to log
  drop databasename     Delete a database and all its tables
  extended-status       Gives an extended status message from the server
  flush-hosts           Flush all cached hosts
  flush-logs            Flush all logs
  flush-status          Clear status variables
  flush-tables          Flush all tables
  flush-threads         Flush the thread cache
  flush-privileges      Reload grant tables (same as reload)
  kill id,id,...        Kill mysql threads
  password [new-password] Change old password to new-password in current format
  ping                  Check if mysqld is alive
  processlist           Show list of active threads in server
  reload                Reload grant tables
  refresh               Flush all tables and close and open logfiles
  shutdown              Take server down
  status                Gives a short status message from the server
  start-replica         Start replication
  start-slave           Deprecated: use start-replica instead
  stop-replica          Stop replication
  stop-slave            Deprecated: use stop-replica instead
  variables             Prints variables available
  version               Get version info from server

C:\Users\mao>
```

语法：

```
mysqladmin [options] command ...
```

选项：

- -u, --user=name #指定用户名
- -p, --password[=name] #指定密码
- -h, --host=name #指定服务器IP或域名
- -P, --port=port #指定连接端口

```
C:\Users\mao>mysqladmin -uroot -pxxx version
mysqladmin: [Warning] Using a password on the command line interface can be
insecure.
mysqladmin  Ver 8.0.27 for Win64 on x86_64 (MySQL Community Server - GPL)
Copyright (c) 2000, 2021, Oracle and/or its affiliates.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Server version          8.0.27
Protocol version        10
Connection              localhost via TCP/IP
TCP port                3306
Uptime:                 2 days 16 sec

Threads: 2  Questions: 252  Slow queries: 0  Opens: 202  Flush tables: 4  Open
tables: 64  Queries per second avg: 0.001

C:\Users\mao>
```

mysqlbinlog

mysqlbinlog： 日志管理工具

语法：

```
mysqlbinlog [options] log-files1 log-files2 ...
```

选项：

- -d, --database=name 指定数据库名称，只列出指定的数据库相关操作。
- -o, --offset=# 忽略掉日志中的前n行命令。
- -r,--result-file=name 将输出的文本格式日志输出到指定文件。
- -s, --short-form 显示简单格式， 省略掉一些信息。
- --start-datatime=date1 --stop-datetime=date2 指定日期间隔内的所有日志。
- --start-position=pos1 --stop-position=pos2 指定位置间隔内的所有日志。

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> ls


    目录: C:\ProgramData\MySQL\MySQL Server 8.0\Data


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/11     12:18                #innodb_temp
d-----          2022/6/2     15:22                hotel
d-----        2021/11/24     16:56                mysql
d-----        2021/11/24     16:56                performance_schema
d-----        2021/11/24     16:58                sakila
d-----         2022/1/27     21:16                student
d-----         2022/2/25     21:15                student1
d-----         2022/1/17     23:11                student_test
d-----        2021/11/24     16:56                sys
```

```
d-----         2022/5/15     21:23                test
d-----         2022/2/26     11:32                tx
d-----        2021/11/24     16:58                world
-a----         2022/6/10     22:22         196608 #ib_16384_0.dblwr
-a----         2022/6/10     21:27        8585216 #ib_16384_1.dblwr
-a----        2021/11/24     16:56             56 auto.cnf
-a----        2021/11/24     16:56           1680 ca-key.pem
-a----        2021/11/24     16:56           1112 ca.pem
-a----        2021/11/24     16:56           1112 client-cert.pem
-a----        2021/11/24     16:56           1680 client-key.pem
-a----         2022/6/11     13:10       12582912 ibdata1
-a----         2022/6/11     12:18       12582912 ibtmp1
-a----         2022/6/11     12:17           1786 ib_buffer_pool
-a----         2022/6/11     12:20       50331648 ib_logfile0
-a----         2022/6/11     13:10       50331648 ib_logfile1
-a----         2022/5/12     17:26            156 MAO-bin.000175
-a----         2022/5/13     23:12         188269 MAO-bin.000176
-a----         2022/5/14     19:45            156 MAO-bin.000177
-a----         2022/5/21     12:56         925670 MAO-bin.000178
-a----         2022/5/22     20:07         189888 MAO-bin.000179
-a----         2022/5/22     23:33            156 MAO-bin.000180
-a----         2022/5/23     12:14            156 MAO-bin.000181
-a----         2022/5/23     22:24            156 MAO-bin.000182
-a----         2022/5/24      9:55            156 MAO-bin.000183
-a----         2022/5/25     19:22            156 MAO-bin.000184
-a----         2022/5/26     16:29            156 MAO-bin.000185
-a----         2022/5/27      9:46            156 MAO-bin.000186
-a----         2022/5/28     13:00            156 MAO-bin.000187
-a----         2022/5/29     19:44            156 MAO-bin.000188
-a----         2022/5/30     21:47            156 MAO-bin.000189
-a----         2022/5/31     19:14            156 MAO-bin.000190
-a----          2022/6/1     12:55            156 MAO-bin.000191
-a----          2022/6/3     19:24          77277 MAO-bin.000192
-a----          2022/6/4     12:22            156 MAO-bin.000193
-a----          2022/6/8     12:24           1296 MAO-bin.000194
-a----          2022/6/8     19:23            372 MAO-bin.000195
-a----          2022/6/8     20:49            179 MAO-bin.000196
-a----          2022/6/9     17:36            179 MAO-bin.000197
-a----         2022/6/10     15:39            179 MAO-bin.000198
-a----         2022/6/11     12:17           6124 MAO-bin.000199
-a----         2022/6/11     12:18            156 MAO-bin.000200
-a----         2022/6/11     12:18            442 MAO-bin.index
-a----         2022/6/11     12:18          39001 MAO-slow.log
-a----         2022/6/12     19:37         395253 MAO.err
-a----         2022/6/11     12:18              5 mao.pid
-a----         2022/6/11     12:18       26214400 mysql.ibd
-a----        2021/11/24     16:56           1676 private_key.pem
-a----        2021/11/24     16:56            452 public_key.pem
-a----        2021/11/24     16:56           1112 server-cert.pem
-a----        2021/11/24     16:56           1676 server-key.pem
-a----         2022/6/11     12:20       16777216 undo_001
-a----         2022/6/11     12:20       16777216 undo_002


PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> mysqlbinlog -s MAO-bin.000196
MAO-bin.000192
WARNING: --short-form is deprecated and will be removed in a future version
```

```
# The proper term is pseudo_replica_mode, but we use this compatibility alias
# to make the statement usable on server versions 8.0.24 and older.
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
ROLLBACK/*!*/;
# [empty]
SET @@SESSION.GTID_NEXT= 'AUTOMATIC' /* added by mysqlbinlog */ /*!*/;
ROLLBACK/*!*/;
# [empty]
# original_commit_timestamp=1654154478861196 (2022-06-02 15:21:18.861196 中国标准
时间)
# immediate_commit_timestamp=1654154478861196 (2022-06-02 15:21:18.861196 中国标准
时间)
/*!80001 SET @@session.original_commit_timestamp=1654154478861196*//*!*/;
/*!80014 SET @@session.original_server_version=80027*//*!*/;
/*!80014 SET @@session.immediate_server_version=80027*//*!*/;
SET @@SESSION.GTID_NEXT= 'ANONYMOUS'/*!*/;
SET TIMESTAMP=1654154478/*!*/;
SET @@session.pseudo_thread_id=999999999/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0,
@@session.unique_checks=1, @@session.autocommit=1/*!*/;
SET @@session.sql_mode=1075838976/*!*/;
SET @@session.auto_increment_increment=1,
@@session.auto_increment_offset=1/*!*/;
/*!\C utf8mb4 *//*!*/;
SET
@@session.character_set_client=255,@@session.collation_connection=255,@@session.
collation_server=255/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
/*!80011 SET @@session.default_collation_for_utf8mb4=255*//*!*/;
/*!80016 SET @@session.default_table_encryption=0*//*!*/;
CREATE DATABASE `hotel` CHARACTER SET 'utf8'
/*!*/;
# original_commit_timestamp=1654154528619831 (2022-06-02 15:22:08.619831 中国标准
时间)
# immediate_commit_timestamp=1654154528619831 (2022-06-02 15:22:08.619831 中国标准
时间)
/*!80001 SET @@session.original_commit_timestamp=1654154528619831*//*!*/;
/*!80014 SET @@session.original_server_version=80027*//*!*/;
/*!80014 SET @@session.immediate_server_version=80027*//*!*/;
SET @@SESSION.GTID_NEXT= 'ANONYMOUS'/*!*/;
use `hotel`/*!*/;
SET TIMESTAMP=1654154528/*!*/;
SET @@session.foreign_key_checks=0/*!*/;
DROP TABLE IF EXISTS `tb_hotel` /* generated by server */
/*!*/;
# original_commit_timestamp=1654154528652913 (2022-06-02 15:22:08.652913 中国标准
时间)
# immediate_commit_timestamp=1654154528652913 (2022-06-02 15:22:08.652913 中国标准
时间)
/*!80001 SET @@session.original_commit_timestamp=1654154528652913*//*!*/;
/*!80014 SET @@session.original_server_version=80027*//*!*/;
/*!80014 SET @@session.immediate_server_version=80027*//*!*/;
SET @@SESSION.GTID_NEXT= 'ANONYMOUS'/*!*/;
SET TIMESTAMP=1654154528/*!*/;
/*!80013 SET @@session.sql_require_primary_key=0*//*!*/;
```

```
CREATE TABLE `tb_hotel`  (
  `id` bigint(20) NOT NULL COMMENT '閰掑簵id',
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL COMMENT '閰掑簵鍚嶇О',
  `address` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL COMMENT '閰掑簵鍦板潃',
  `price` int(10) NOT NULL COMMENT '閰掑簵浠锋牸',
  `score` int(2) NOT NULL COMMENT '閰掑簵璇勫垎',
  `brand` varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL COMMENT '閰掑簵鍝佺墝',
  `city` varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL COMMENT '鎵€鍦ㄥ煄甯?,
  `star_name` varchar(16) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT NULL COMMENT '閰掑簵鏄熺骇锛?鏄熺骇鍒?鏄熺紝1閽诲埌5閽?,
  `business` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT NULL COMMENT '鍟嗗湀',
  `latitude` varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL COMMENT '绾害',
  `longitude` varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL COMMENT '缁忓害',
  `pic` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT NULL COMMENT '閰掑簵鍥剧墖',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT = Compact
/*!*/;
/*!50718 SET TRANSACTION ISOLATION LEVEL READ COMMITTED*//*!*/;
# original_commit_timestamp=1654154528681977 (2022-06-02 15:22:08.681977 中国标准时间)
# immediate_commit_timestamp=1654154528681977 (2022-06-02 15:22:08.681977 中国标准时间)
/*!80001 SET @@session.original_commit_timestamp=1654154528681977*//*!*/;
/*!80014 SET @@session.original_server_version=80027*//*!*/;
/*!80014 SET @@session.immediate_server_version=80027*//*!*/;
SET @@SESSION.GTID_NEXT= 'ANONYMOUS'/*!*/;
SET TIMESTAMP=1654154528/*!*/;
BEGIN
/*!*/;
COMMIT/*!*/;
SET @@SESSION.GTID_NEXT= 'AUTOMATIC' /* added by mysqlbinlog */ /*!*/;
DELIMITER ;
# End of log file
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data>
```

mysqlshow

mysqlshow： 客户端对象查找工具，用来很快地查找存在哪些数据库、数据库中的表、表中的列或者索引。

语法：

```
mysqlshow [options] [db_name [table_name [col_name]]]
```

选项：

- --count 显示数据库及表的统计信息（数据库，表 均可以不指定）
- -i 显示指定数据库或者指定表的状态信息

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> mysqlshow -uroot -p --count
Enter password: ********
+--------------------+--------+--------------+
|     Databases      | Tables |  Total Rows  |
+--------------------+--------+--------------+
| hotel              |      1 |          201 |
| information_schema |     79 |        20402 |
| mysql              |     37 |         4250 |
| performance_schema |    110 |       241985 |
| sakila             |     23 |        50086 |
| student            |      4 |           36 |
| student1           |     13 |         9130 |
| student_test       |      1 |          800 |
| sys                |    101 |         6092 |
| test               |     15 |         1668 |
| tx                 |      3 |            3 |
| world              |      3 |         5302 |
+--------------------+--------+--------------+
12 rows in set.
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data>
```

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> mysqlshow -uroot -p student1 --
count
Enter password: ********
Database: student1
+-------------------------+----------+------------+
|         Tables          | Columns  | Total Rows |
+-------------------------+----------+------------+
| administrators          |        6 |         30 |
| administrators_password |        2 |         30 |
| class                   |        5 |         10 |
| course                  |        5 |         12 |
| forum                   |        6 |         70 |
| login_log               |        4 |         70 |
| news                    |        6 |          9 |
| score                   |        7 |       7199 |
| student                 |       13 |        600 |
| student_password        |        2 |        600 |
| teach                   |        5 |        300 |
| teacher                 |        7 |        100 |
| teacher_password        |        2 |        100 |
+-------------------------+----------+------------+
13 rows in set.
```

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data>
```

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> mysqlshow -uroot -p student1
score --count
Enter password: ********
Database: student1  Table: score  Rows: 7199
+-------------+-------------+-----------------+------+-----+---------+-------+--
-------------------------------+----------+
| Field       | Type        | Collation       | Null | Key | Default | Extra |
Privileges                      | Comment  |
+-------------+-------------+-----------------+------+-----+---------+-------+--
-------------------------------+----------+
| no          | bigint      |                 | NO   | MUL |         |       |
select,insert,update,references | 学生学号 |
| course_no   | bigint      |                 | NO   | MUL |         |       |
select,insert,update,references | 课程编号 |
| usual_score | float       |                 | YES  |     |         |       |
select,insert,update,references | 平时成绩 |
| end_score   | float       |                 | YES  |     |         |       |
select,insert,update,references | 期末成绩 |
| final_score | float       |                 | YES  |     |         |       |
select,insert,update,references | 最终分数 |
| grade_point | float       |                 | YES  |     |         |       |
select,insert,update,references | 绩点     |
| semester    | varchar(15) | utf8_general_ci | YES  |     |         |       |
select,insert,update,references | 学期     |
+-------------+-------------+-----------------+------+-----+---------+-------+--
-------------------------------+----------+
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data>
```

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> mysqlshow -uroot -p student1
score no --count
Enter password: ********
Database: student1  Table: score  Rows: 7199  Wildcard: no
+-------+--------+-----------+------+-----+---------+-------+------------------
--------------+----------+
| Field | Type   | Collation | Null | Key | Default | Extra | Privileges
             | Comment  |
+-------+--------+-----------+------+-----+---------+-------+------------------
--------------+----------+
| no    | bigint |           | NO   | MUL |         |       |
select,insert,update,references | 学生学号 |
+-------+--------+-----------+------+-----+---------+-------+------------------
--------------+----------+
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data>
```

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> mysqlshow -uroot -p student1
score -i
Enter password: ********
Database: student1  Wildcard: score
+-------+--------+---------+------------+------+----------------+-------------+-
---------------+--------------+-----------+----------------+-------------------
--+-------------+------------+-----------------+----------+--------------------
+---------+
| Name  | Engine | Version | Row_format | Rows | Avg_row_length | Data_length |
Max_data_length | Index_length | Data_free | Auto_increment | Create_time
| Update_time | Check_time | Collation       | Checksum | Create_options     |
Comment |
+-------+--------+---------+------------+------+----------------+-------------+-
---------------+--------------+-----------+----------------+-------------------
--+-------------+------------+-----------------+----------+--------------------
+---------+
| score | InnoDB | 10      | Dynamic    | 7199 | 220            | 1589248     |
0               | 393216       | 4194304   |                | 2022-02-25
21:15:18 |             |            | utf8_general_ci |          |
row_format=DYNAMIC |         |
+-------+--------+---------+------------+------+----------------+-------------+-
---------------+--------------+-----------+----------------+-------------------
--+-------------+------------+-----------------+----------+--------------------
+---------+
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data>
```

mysqldump

mysqldump： 客户端工具用来备份数据库或在不同数据库之间进行数据迁移。备份内容包含创建表，及插入表的SQL语句。

语法：

```
mysqldump [options] db_name [tables]
mysqldump [options] --database/-B db1 [db2 db3...]
mysqldump [options] --all-databases/-A
```

连接选项：

- -u, --user=name 指定用户名
- -p, --password[=name] 指定密码
- -h, --host=name 指定服务器ip或域名
- -P, --port=# 指定连接端口

输出选项：

- --add-drop-database 在每个数据库创建语句前加上 drop database 语句

- --add-drop-table 在每个表创建语句前加上 drop table 语句，默认开启；不 开启 (--skip-add-drop-table)
- -n, --no-create-db 不包含数据库的创建语句
- -t, --no-create-info 不包含数据表的创建语句
- -d --no-data 不包含数据
- -T, --tab=name 自动生成两个文件：一个.sql文件，创建表结构的语句；一 个.txt文件，数据文件

```
PS C:\Users\mao\Desktop\test> ls
PS C:\Users\mao\Desktop\test> mysqldump -uroot -p student1 > student1.sql
Enter password: ********
PS C:\Users\mao\Desktop\test> ls


    目录: C:\Users\mao\Desktop\test


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2022/6/13     12:51        1154034 student1.sql


PS C:\Users\mao\Desktop\test>
```

```
PS C:\Users\mao\Desktop\test> mysqldump -uroot -p -d student1 >
student1_tables.sql
Enter password: ********
PS C:\Users\mao\Desktop\test> ls


    目录: C:\Users\mao\Desktop\test


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2022/6/13     12:51        1154034 student1.sql
-a----         2022/6/13     12:53          28722 student1_tables.sql


PS C:\Users\mao\Desktop\test> mysqldump -uroot -p -t student1 >
student1_data.sql
Enter password: ********
PS C:\Users\mao\Desktop\test> ls


    目录: C:\Users\mao\Desktop\test


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2022/6/13     12:51        1154034 student1.sql
-a----         2022/6/13     12:54        1127872 student1_data.sql
-a----         2022/6/13     12:53          28722 student1_tables.sql
```

```
PS C:\Users\mao\Desktop\test>
```

mysqlimport

mysqlimport 是客户端数据导入工具，用来导入mysqldump 加 -T 参数后导出的文本文件。

语法：

```
mysqlimport [options] db_name textfile1 [textfile2...]
```

source

如果需要导入sql文件，可以使用mysql中的source 指令

语法：

```
source xxxxx.sql
```

日志

错误日志

错误日志是 MySQL 中最重要的日志之一，它记录了当 mysqld 启动和停止时，以及服务器在运行过 程中发生任何严重错误时的相关信息。当数据库出现任何故障导致无法正常使用时，建议首先查看此日志。

查看日志位置：

```
show variables like '%log_error%';
```

```
mysql> show variables like '%log_error%';
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| binlog_error_action        | ABORT_SERVER                         |
| log_error                  | .\MAO.err                            |
| log_error_services         | log_filter_internal; log_sink_internal |
| log_error_suppression_list |                                      |
| log_error_verbosity        | 2                                    |
+----------------------------+--------------------------------------+
5 rows in set, 1 warning (0.00 sec)

mysql>
```

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> ls


    目录: C:\ProgramData\MySQL\MySQL Server 8.0\Data


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/11     12:18                #innodb_temp
d-----          2022/6/2     15:22                hotel
d-----        2021/11/24     16:56                mysql
d-----        2021/11/24     16:56                performance_schema
d-----        2021/11/24     16:58                sakila
d-----         2022/1/27     21:16                student
d-----         2022/2/25     21:15                student1
d-----         2022/1/17     23:11                student_test
d-----        2021/11/24     16:56                sys
d-----         2022/5/15     21:23                test
d-----         2022/2/26     11:32                tx
d-----        2021/11/24     16:58                world
-a----         2022/6/13     12:47         196608 #ib_16384_0.dblwr
-a----         2022/6/10     21:27        8585216 #ib_16384_1.dblwr
-a----        2021/11/24     16:56             56 auto.cnf
-a----        2021/11/24     16:56           1680 ca-key.pem
-a----        2021/11/24     16:56           1112 ca.pem
-a----        2021/11/24     16:56           1112 client-cert.pem
-a----        2021/11/24     16:56           1680 client-key.pem
-a----         2022/6/13     12:47       12582912 ibdata1
-a----         2022/6/11     12:20       12582912 ibtmp1
-a----         2022/6/11     12:17           1786 ib_buffer_pool
-a----         2022/6/13     12:47       50331648 ib_logfile0
-a----         2022/6/13     12:47       50331648 ib_logfile1
-a----         2022/5/12     17:26            156 MAO-bin.000175
-a----         2022/5/13     23:12         188269 MAO-bin.000176
-a----         2022/5/14     19:45            156 MAO-bin.000177
-a----         2022/5/21     12:56         925670 MAO-bin.000178
-a----         2022/5/22     20:07         189888 MAO-bin.000179
-a----         2022/5/22     23:33            156 MAO-bin.000180
-a----         2022/5/23     12:14            156 MAO-bin.000181
-a----         2022/5/23     22:24            156 MAO-bin.000182
```

```
-a----          2022/5/24      9:55            156 MAO-bin.000183
-a----          2022/5/25     19:22            156 MAO-bin.000184
-a----          2022/5/26     16:29            156 MAO-bin.000185
-a----          2022/5/27      9:46            156 MAO-bin.000186
-a----          2022/5/28     13:00            156 MAO-bin.000187
-a----          2022/5/29     19:44            156 MAO-bin.000188
-a----          2022/5/30     21:47            156 MAO-bin.000189
-a----          2022/5/31     19:14            156 MAO-bin.000190
-a----           2022/6/1     12:55            156 MAO-bin.000191
-a----           2022/6/3     19:24          77277 MAO-bin.000192
-a----           2022/6/4     12:22            156 MAO-bin.000193
-a----           2022/6/8     12:24           1296 MAO-bin.000194
-a----           2022/6/8     19:23            372 MAO-bin.000195
-a----           2022/6/8     20:49            179 MAO-bin.000196
-a----           2022/6/9     17:36            179 MAO-bin.000197
-a----          2022/6/10     15:39            179 MAO-bin.000198
-a----          2022/6/11     12:17           6124 MAO-bin.000199
-a----          2022/6/11     12:18            156 MAO-bin.000200
-a----          2022/6/11     12:18            442 MAO-bin.index
-a----          2022/6/11     12:18          39001 MAO-slow.log
-a----          2022/6/13     12:40         395416 MAO.err
-a----          2022/6/11     12:18              5 mao.pid
-a----          2022/6/13     12:47       26214400 mysql.ibd
-a----         2021/11/24     16:56           1676 private_key.pem
-a----         2021/11/24     16:56            452 public_key.pem
-a----         2021/11/24     16:56           1112 server-cert.pem
-a----         2021/11/24     16:56           1676 server-key.pem
-a----          2022/6/13     12:47       16777216 undo_001
-a----          2022/6/13     12:47       16777216 undo_002


PS C:\ProgramData\MySQL\MySQL Server 8.0\Data>
```

二进制日志

二进制日志（BINLOG）记录了所有的 DDL（数据定义语言）语句和 DML（数据操纵语言）语句，但 不包括数据查询（SELECT、SHOW）语句。

作用：

- 灾难时的数据恢复
- MySQL的主从复制

查看：

```
show variables like '%log_bin%';
```

- log_bin_basename：当前数据库服务器的binlog日志的基础名称(前缀)，具体的binlog文 件名需要再该basename的基础上加上编号(编号从000001开始)。
- log_bin_index：binlog的索引文件，里面记录了当前服务器关联的binlog文件有哪些。

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 40
Server version: 8.0.27 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables like '%log_bin%';
+---------------------------------+-----------------------------------------
-------------+
| Variable_name                   | Value
          |
+---------------------------------+-----------------------------------------
-------------+
| log_bin                         | ON
           |
| log_bin_basename                | C:\ProgramData\MySQL\MySQL Server
8.0\Data\MAO-bin       |
| log_bin_index                   | C:\ProgramData\MySQL\MySQL Server
8.0\Data\MAO-bin.index |
| log_bin_trust_function_creators | OFF
          |
| log_bin_use_v1_row_events       | OFF
           |
| sql_log_bin                     | ON
           |
+---------------------------------+-----------------------------------------
-------------+
6 rows in set, 1 warning (0.00 sec)

mysql>
```

```
PS C:\ProgramData\MySQL\MySQL Server 8.0\Data> ls


    目录: C:\ProgramData\MySQL\MySQL Server 8.0\Data


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/11     12:18                #innodb_temp
d-----          2022/6/2     15:22                hotel
```

```
d-----        2021/11/24     16:56              mysql
d-----        2021/11/24     16:56              performance_schema
d-----        2021/11/24     16:58              sakila
d-----         2022/1/27     21:16              student
d-----         2022/2/25     21:15              student1
d-----         2022/1/17     23:11              student_test
d-----        2021/11/24     16:56              sys
d-----         2022/5/15     21:23              test
d-----         2022/2/26     11:32              tx
d-----        2021/11/24     16:58              world
-a----         2022/6/13     12:47         196608 #ib_16384_0.dblwr
-a----         2022/6/10     21:27        8585216 #ib_16384_1.dblwr
-a----        2021/11/24     16:56             56 auto.cnf
-a----        2021/11/24     16:56           1680 ca-key.pem
-a----        2021/11/24     16:56           1112 ca.pem
-a----        2021/11/24     16:56           1112 client-cert.pem
-a----        2021/11/24     16:56           1680 client-key.pem
-a----         2022/6/13     12:47       12582912 ibdata1
-a----         2022/6/11     12:20       12582912 ibtmp1
-a----         2022/6/11     12:17           1786 ib_buffer_pool
-a----         2022/6/13     12:47       50331648 ib_logfile0
-a----         2022/6/13     12:47       50331648 ib_logfile1
-a----         2022/5/12     17:26            156 MAO-bin.000175
-a----         2022/5/13     23:12         188269 MAO-bin.000176
-a----         2022/5/14     19:45            156 MAO-bin.000177
-a----         2022/5/21     12:56         925670 MAO-bin.000178
-a----         2022/5/22     20:07         189888 MAO-bin.000179
-a----         2022/5/22     23:33            156 MAO-bin.000180
-a----         2022/5/23     12:14            156 MAO-bin.000181
-a----         2022/5/23     22:24            156 MAO-bin.000182
-a----         2022/5/24      9:55            156 MAO-bin.000183
-a----         2022/5/25     19:22            156 MAO-bin.000184
-a----         2022/5/26     16:29            156 MAO-bin.000185
-a----         2022/5/27      9:46            156 MAO-bin.000186
-a----         2022/5/28     13:00            156 MAO-bin.000187
-a----         2022/5/29     19:44            156 MAO-bin.000188
-a----         2022/5/30     21:47            156 MAO-bin.000189
-a----         2022/5/31     19:14            156 MAO-bin.000190
-a----          2022/6/1     12:55            156 MAO-bin.000191
-a----          2022/6/3     19:24          77277 MAO-bin.000192
-a----          2022/6/4     12:22            156 MAO-bin.000193
-a----          2022/6/8     12:24           1296 MAO-bin.000194
-a----          2022/6/8     19:23            372 MAO-bin.000195
-a----          2022/6/8     20:49            179 MAO-bin.000196
-a----          2022/6/9     17:36            179 MAO-bin.000197
-a----         2022/6/10     15:39            179 MAO-bin.000198
-a----         2022/6/11     12:17           6124 MAO-bin.000199
-a----         2022/6/11     12:18            156 MAO-bin.000200
-a----         2022/6/11     12:18            442 MAO-bin.index
-a----         2022/6/11     12:18          39001 MAO-slow.log
-a----         2022/6/13     12:40         395416 MAO.err
-a----         2022/6/11     12:18              5 mao.pid
-a----         2022/6/13     12:47       26214400 mysql.ibd
-a----        2021/11/24     16:56           1676 private_key.pem
-a----        2021/11/24     16:56            452 public_key.pem
-a----        2021/11/24     16:56           1112 server-cert.pem
-a----        2021/11/24     16:56           1676 server-key.pem
-a----         2022/6/13     12:47       16777216 undo_001
```

```
-a----         2022/6/13     12:47       16777216 undo_002


PS C:\ProgramData\MySQL\MySQL Server 8.0\Data>
```

格式

日志格式	含义
STATEMENT	基于SQL语句的日志记录，记录的是SQL语句，对数据进行修改的SQL都会记录在日志文件中。
ROW	基于行的日志记录，记录的是每一行的数据变更。（默认）
MIXED	混合了STATEMENT和ROW两种格式，默认采用STATEMENT，在某些特殊情况下会 自动切换为ROW进行记录。

查看：

```
show variables like '%binlog_format%'
```

```
mysql> show variables like '%binlog_format%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW   |
+---------------+-------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

删除

对于比较繁忙的业务系统，每天生成的binlog数据巨大，如果长时间不清除，将会占用大量磁盘空 间。可以通过以下几种方式清理日志：

- reset master：删除全部 binlog 日志，删除之后，日志编号，将 从 binlog.000001重新开始
- purge master logs to 'binlog.*' ： 删除 * 编号之前的所有日志
- purge master logs before 'yyyy-mm-dd hh24:mi:ss' ： 删除日志为 "yyyy-mm-dd hh24:mi:ss" 之前 产生的所有日志

也可以在mysql的配置文件中配置二进制日志的过期时间，设置了之后，二进制日志过期会自动删除。

```
show variables like '%binlog_expire_logs_seconds%';
```

```
mysql> show variables like '%binlog_expire_logs_seconds%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| binlog_expire_logs_seconds | 2592000 |
+------------------------------+---------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

查询日志

查询日志中记录了客户端的所有操作语句，而二进制日志不包含查询数据的SQL语句。默认情况下，查询日志是未开启的

开启查询日志：

更改my.conf文件

```
#该选项用来开启查询日志 ， 可选值 ： 0 或者 1 ； 0 代表关闭， 1 代表开启
general_log=1
#设置日志的文件名 ， 如果没有指定， 默认的文件名为 host_name.log
general_log_file=mysql_query.log
```

慢查询日志

慢查询日志记录了所有执行时间超过参数 long_query_time 设置值并且扫描记录数不小于 min_examined_row_limit 的所有的SQL语句的日志，默认未开启。long_query_time 默认为10 秒，最小为 0， 精度可以到微秒。

开启慢查询日志：

更改my.conf文件

```
#慢查询日志
slow_query_log=1
#执行时间参数
long_query_time=3
```

默认情况下，不会记录管理语句，也不会记录不使用索引进行查找的查询。可以使用 log_slow_admin_statements和 更改此行为 log_queries_not_using_indexes

```
#记录执行较慢的管理语句
log_slow_admin_statements =1
#记录执行较慢的未使用索引的语句
log_queries_not_using_indexes = 1
```

主从复制

主从复制是指将主数据库的 DDL 和 DML 操作通过二进制日志传到从库服务器中，然后在从库上对这 些日志重新执行（也叫重做），从而使得从库和主库的数据保持同步。

MySQL支持一台主库同时向多台从库进行复制， 从库同时也可以作为其他从服务器的主库，实现链状复制。

主从复制的优点：

- 主库出现问题，可以快速切换到从库提供服务。
- 实现读写分离，降低主库的访问压力。
- 可以在从库中执行备份，以避免备份期间影响主库服务。

原理

- Master 主库在事务提交时，会把数据变更记录在二进制日志文件 Binlog 中。
- slave节点的 IO线程从库读取主库的二进制日志文件 Binlog ，写入到从库的中继日志 Relay Log
- slave节点的SQL线程重做中继日志中的事件，将改变反映它自己的数据。

搭建

使用docker容器，容器名mysql2为master节点，端口号为3308，容器名mysql3为slave节点，端口号为3309

1. 搜索和下载

命令：docker pull mysql

2. 查看是否拉取成功

docker images

```
PS C:\Users\mao\Desktop> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
tomcat       latest    c795915cb678   2 weeks ago   680MB
redis        latest    53aa81e8adfa   2 weeks ago   117MB
mysql        latest    65b636d5542b   2 weeks ago   524MB
ubuntu       latest    d2e4e1f51132   6 weeks ago   77.8MB
PS C:\Users\mao\Desktop>
```

3. 运行

创建mysql网络

```
C:\Users\mao>docker network create mysql
a713e60a41df792986ce14cc86888691dbdbc6850a009dd5625437cee3da64cd

C:\Users\mao>
```

master节点:

```
docker run -p 3308:3306 --privileged=true --name mysql2 --network mysql -v
H:/Docker/mysql2/log/:/var/log/mysql/ -v H:/Docker/mysql2/data/:/var/lib/mysql/
-v H:/Docker/mysql2/conf/:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456 -d
mysql
```

slave节点:

```
docker run -d -p 3309:3306 --privileged=true --network mysql -v
H:/Docker/mysql3/log/:/var/log/mysql -v H:/Docker/mysql3/data:/var/lib/mysql -v
H:/Docker/mysql3/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456  --name
mysql3 mysql
```

使用了容器数据卷，映射到了主机的H:/Docker/mysql2和H:/Docker/mysql3目录下，密码为123456

```
PS C:\Users\mao\Desktop> docker run -p 3308:3306 --privileged=true --name mysql2
--network mysql -v H:/Docker/mysql2/log/:/var/log/mysql/ -v
H:/Docker/mysql2/data/:/var/lib/mysql/ -v
H:/Docker/mysql2/conf/:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456 -d mysql
acc4ae47d7fe4c70a9ce0968495fde6ba1562b8d5c348d44caec902f1b8fe8f0
PS C:\Users\mao\Desktop> docker run -d -p 3309:3306 --privileged=true --network
mysql -v H:/Docker/mysql3/log/:/var/log/mysql -v
H:/Docker/mysql3/data:/var/lib/mysql -v H:/Docker/mysql3/conf:/etc/mysql/conf.d
-e MYSQL_ROOT_PASSWORD=123456  --name mysql3 mysql
bc3a894f3f5ad0d1015a6e1a5fe155de88c89077b15ba8b68d0e47b437608efe
PS C:\Users\mao\Desktop>
```

4. 验证是否启动成功

docker ps -a

```
CONTAINER ID   IMAGE                COMMAND                  CREATED
STATUS          PORTS                                          NAMES
39eda98e3f14   mysql                "docker-entrypoint.s…"   35 seconds ago   Up
34 seconds   33060/tcp, 0.0.0.0:3309->3306/tcp                 mysql3
27c63b3f1ae1   mysql                "docker-entrypoint.s…"   45 seconds ago   Up
44 seconds   33060/tcp, 0.0.0.0:3308->3306/tcp                 mysql2
1219851e3bc5   grafana/grafana      "/run.sh"                4 days ago       Up
4 minutes    0.0.0.0:3000->3000/tcp                           desktop_grafana_1
059cf60a61b1   google/cadvisor      "/usr/bin/cadvisor -…"   4 days ago       Up
4 minutes    0.0.0.0:8080->8080/tcp                           desktop_cadvisor_1
71da6b2b40a2   tutum/influxdb:0.9   "/run.sh"                4 days ago       Up
4 minutes    0.0.0.0:8083->8083/tcp, 0.0.0.0:8086->8086/tcp   desktop_influxdb_1
PS C:\Users\mao\Desktop>
```

5. 验证容器数据卷是否创建成功

```
PS H:\Docker> ls


    目录: H:\Docker


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/13     22:40                mysql
d-----         2022/6/14     13:24                mysql2
d-----         2022/6/14     13:24                mysql3
d-----         2022/6/13     21:35                redis
d-----         2022/6/13     23:05                tomcat


PS H:\Docker> cd mysql2
PS H:\Docker\mysql2> ls


    目录: H:\Docker\mysql2


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/14     13:24                conf
da----         2022/6/14     13:26                data
d-----         2022/6/14     13:24                log
```

```
PS H:\Docker\mysql2> cd data
PS H:\Docker\mysql2\data> ls


    目录: H:\Docker\mysql2\data


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/14     13:26                #innodb_temp
d-----         2022/6/14     13:24                mysql
d-----         2022/6/14     13:24                performance_schema
d-----         2022/6/14     13:24                sys
-a----         2022/6/14     13:26         196608 #ib_16384_0.dblwr
-a----         2022/6/14     13:24        8585216 #ib_16384_1.dblwr
-a----         2022/6/14     13:24             56 auto.cnf
-a----         2022/6/14     13:26        3116922 binlog.000001
-a----         2022/6/14     13:26            157 binlog.000002
-a----         2022/6/14     13:26             32 binlog.index
-a----         2022/6/14     13:24           1676 ca-key.pem
-a----         2022/6/14     13:24           1112 ca.pem
-a----         2022/6/14     13:24           1112 client-cert.pem
-a----         2022/6/14     13:24           1676 client-key.pem
-a----         2022/6/14     13:26       12582912 ibdata1
-a----         2022/6/14     13:26       12582912 ibtmp1
-a----         2022/6/14     13:26           5730 ib_buffer_pool
-a----         2022/6/14     13:26       50331648 ib_logfile0
-a----         2022/6/14     13:24       50331648 ib_logfile1
-a----         2022/6/14     13:26       31457280 mysql.ibd
-a----         2022/6/14     13:24           1676 private_key.pem
-a----         2022/6/14     13:24            452 public_key.pem
-a----         2022/6/14     13:24           1112 server-cert.pem
-a----         2022/6/14     13:24           1676 server-key.pem
-a----         2022/6/14     13:26       16777216 undo_001
-a----         2022/6/14     13:26       16777216 undo_002
```

```
PS H:\Docker> cd mysql3
PS H:\Docker\mysql3> ls


    目录: H:\Docker\mysql3


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/14     13:24                conf
da----         2022/6/14     13:27                data
d-----         2022/6/14     13:24                log


PS H:\Docker\mysql3> cd data
PS H:\Docker\mysql3\data> ls
```

```
    目录: H:\Docker\mysql3\data


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2022/6/14     13:27                #innodb_temp
d-----         2022/6/14     13:25                mysql
d-----         2022/6/14     13:25                performance_schema
d-----         2022/6/14     13:26                sys
-a----         2022/6/14     13:29         196608 #ib_16384_0.dblwr
-a----         2022/6/14     13:24        8585216 #ib_16384_1.dblwr
-a----         2022/6/14     13:24             56 auto.cnf
-a----         2022/6/14     13:27        3116922 binlog.000001
-a----         2022/6/14     13:27            157 binlog.000002
-a----         2022/6/14     13:27             32 binlog.index
-a----         2022/6/14     13:25           1676 ca-key.pem
-a----         2022/6/14     13:25           1112 ca.pem
-a----         2022/6/14     13:25           1112 client-cert.pem
-a----         2022/6/14     13:25           1680 client-key.pem
-a----         2022/6/14     13:27       12582912 ibdata1
-a----         2022/6/14     13:27       12582912 ibtmp1
-a----         2022/6/14     13:27           5730 ib_buffer_pool
-a----         2022/6/14     13:29       50331648 ib_logfile0
-a----         2022/6/14     13:24       50331648 ib_logfile1
-a----         2022/6/14     13:27       31457280 mysql.ibd
-a----         2022/6/14     13:25           1680 private_key.pem
-a----         2022/6/14     13:25            452 public_key.pem
-a----         2022/6/14     13:25           1112 server-cert.pem
-a----         2022/6/14     13:25           1676 server-key.pem
-a----         2022/6/14     13:29       16777216 undo_001
-a----         2022/6/14     13:29       16777216 undo_002


PS H:\Docker\mysql3\data>
```

6. 创建配置文件

在H:\Docker\mysql2\conf下创建my.conf

在H:\Docker\mysql3\conf下创建my.conf

输入以下内容:

H:\Docker\mysql2\conf:

```
[mysqld]
## 设置server_id，同一局域网中需要唯一
server_id=101
## 指定不需要同步的数据库名称
binlog-ignore-db=mysql
## 开启二进制日志功能
log-bin=mall-mysql-bin
## 设置二进制日志使用内存大小（事务
```

```
binlog_cache_size=1M
## 设置使用的二进制日志格式（mixed,statement,row）
binlog_format=mixed
## 二进制日志过期清理时间。默认值为0，表示不自动清理。
expire_logs_days=7
## 跳过主从复制中遇到的所有错误或指定类型的错误，避免slave端复制中断。
## 如：1062错误是指一些主键重复，1032错误是因为主从数据库数据不一致
#是否只读,1 代表只读, 0 代表读写
read-only=0
slave_skip_errors=1062
default_character_set=utf8
collation_server = utf8_general_ci
character_set_server = utf8
```

H:\Docker\mysql3\conf:

```
[mysqld]
## 设置server_id，同一局域网中需要唯一
server_id=102
## 指定不需要同步的数据库名称
binlog-ignore-db=mysql
## 开启二进制日志功能，以备Slave作为其它数据库实例的Master时使用
log-bin=mall-mysql-slave1-bin
## 设置二进制日志使用内存大小（事务）
binlog_cache_size=1M
## 设置使用的二进制日志格式（mixed,statement,row）
binlog_format=mixed
## 二进制日志过期清理时间。默认值为0，表示不自动清理。
expire_logs_days=7
## 跳过主从复制中遇到的所有错误或指定类型的错误，避免slave端复制中断。
## 如：1062错误是指一些主键重复，1032错误是因为主从数据库数据不一致
slave_skip_errors=1062
## relay_log配置中继日志
relay_log=mall-mysql-relay-bin
## log_slave_updates表示slave将复制事件写进自己的二进制日志
log_slave_updates=1
## slave设置为只读（具有super权限的用户除外）
read_only=1
default_character_set=utf8
collation_server = utf8_general_ci
character_set_server = utf8
```

7. 重启服务器

docker restart mysql2

docker restart mysql3

```
PS C:\Users\mao\Desktop> docker restart mysql2
mysql2
PS C:\Users\mao\Desktop> docker restart mysql3
mysql3
PS C:\Users\mao\Desktop> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
             PORTS                               NAMES
81902b3cfdc4   mysql     "docker-entrypoint.s…"   2 minutes ago   Up 10 seconds
             33060/tcp, 0.0.0.0:3309->3306/tcp   mysql3
24ee3e986397   mysql     "docker-entrypoint.s…"   2 minutes ago   Up 23 seconds
             33060/tcp, 0.0.0.0:3308->3306/tcp   mysql2
2d379d342bb6   mysql     "docker-entrypoint.s…"   25 hours ago    Exited (0) 55
minutes ago                                      mysql1
3ca156e4541d   tomcat    "catalina.sh run"        26 hours ago    Exited (143)
25 hours ago                                      tomcat1
3948921a6099   redis     "docker-entrypoint.s…"   26 hours ago    Exited (0) 25
hours ago                                        redis1
PS C:\Users\mao\Desktop>
```

8. 登录master节点

mysql -P 3308 -u root -p

密码为123456

```
C:\Users\mao>mysql -P 3308 -u root -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.29 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

或者

```
docker exec -it mysql2 /bin/bash
```

```
mysql -u root -p
```

9. 查看二进制日志坐标

show master status;

```
mysql> show master status;
+---------------+----------+--------------+------------------+------------------
-+
| File          | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set
|
+---------------+----------+--------------+------------------+------------------
-+
| binlog.000001 |      157 |              |                  |
|
+---------------+----------+--------------+------------------+------------------
-+
1 row in set (0.00 sec)

mysql>
```

说明:

- file : 从哪个日志文件开始推送日志文件
- position： 从哪个位置开始推送日志
- binlog_ignore_db : 指定不需要同步的数据库

10.创建slave账户

```
CREATE USER 'slave'@'%' IDENTIFIED BY '123456';
```

授予权限

```
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'%';
```

```
mysql> CREATE USER 'slave'@'%' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.05 sec)

mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'%';
Query OK, 0 rows affected (0.06 sec)

mysql>
```

11 .登录slave节点

mysql -P 3309 -u root -p

或者进入容器：docker exec -it mysql3 /bin/bash

再使用mysql -u root -p

密码为123456

```
PS C:\Users\mao\Desktop> docker exec -it mysql3 /bin/bash
root@81902b3cfdc4:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.29 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

12. 登录slave节点设置主从配置

```
CHANGE REPLICATION SOURCE TO SOURCE_HOST='mysql2',
SOURCE_USER='slave',SOURCE_PORT=3306,
SOURCE_PASSWORD='', SOURCE_LOG_FILE='binlog.000001',
SOURCE_LOG_POS=157;
```

8.0.23之前：

```
CHANGE MASTER TO MASTER_HOST='mysql2', MASTER_USER='slave',MASTER_PORT=3306,
MASTER_PASSWORD='123456', MASTER_LOG_FILE='binlog.000001',
MASTER_LOG_POS=157;
```

参数名	含义	8.0.23之前
SOURCE_HOST	主库IP地址	MASTER_HOST
SOURCE_USER	连接主库的用户名	MASTER_USER
SOURCE_PASSWORD	连接主库的密码	MASTER_PASSWORD
SOURCE_LOG_FILE	binlog日志文件名	MASTER_LOG_FILE
SOURCE_LOG_POS	binlog日志文件位置	MASTER_LOG_POS
SOURCE_PORT	端口号	MASTER_PORT

注意：binlog日志文件名和binlog日志文件位置要和master节点查出来的数据要一致，命令是show master status

```
mysql> CHANGE REPLICATION SOURCE TO SOURCE_HOST='mysql2',
SOURCE_USER='slave',SOURCE_PORT=3306,
    -> SOURCE_PASSWORD='123456', SOURCE_LOG_FILE='binlog.000003',
    -> SOURCE_LOG_POS=157;
Query OK, 0 rows affected, 1 warning (0.63 sec)

mysql>
```

13. 开启同步操作

在slave节点中

输入以下命令：

```
start replica;
```

8.0.23之前：

```
start slave;
```

```
mysql> start slave;
Query OK, 0 rows affected, 1 warning (0.17 sec)

mysql>
```

14. 查看主从同步状态

在slave节点中

输入以下命令：

```
show replica status;
```

8.0.23之前：

```
show slave status;
```

或者

```
show replica status \G;
```

```
show slave status \G;
```

```

```

如果Slave_IO_Running和Slave_SQL_Running都为yes，则为成功

15. 测试

在master节点，创建数据库，创建表，插入数据

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql> create database db1;
Query OK, 1 row affected (0.09 sec)

mysql> use db1;
Database changed
mysql> create table test(id int , name char(5));
Query OK, 0 rows affected (0.37 sec)
```

```
mysql> insert into test values(1,"abc");
Query OK, 1 row affected (0.11 sec)

mysql> insert into test values(1,"张三");
Query OK, 1 row affected (0.07 sec)

mysql> select * from test;
+------+------+
| id   | name |
+------+------+
|    1 | abc  |
|    1 | 张三 |
+------+------+
2 rows in set (0.00 sec)

mysql>
```

分库分表

拆分策略

垂直拆分和水平拆分

垂直拆分

垂直分库

以表为依据，根据业务将不同表拆分到不同库中。

- 每个库的表结构都不一样。
- 每个库的数据也不一样。
- 所有库的并集是全量数据。

垂直分表

以字段为依据，根据字段属性将不同字段拆分到不同表中。

- 每个表的结构都不一样。
- 每个表的数据也不一样，一般通过一列（主键/外键）关联。
- 所有表的并集是全量数据。

水平拆分

水平分库

以字段为依据，按照一定策略，将一个库的数据拆分到多个库中。

- 每个库的表结构都一样。
- 每个库的数据都不一样。
- 所有库的并集是全量数据。

水平分表

以字段为依据，按照一定策略，将一个表的数据拆分到多个表中。

- 每个表的表结构都一样。
- 每个表的数据都不一样。
- 所有表的并集是全量数据。

实现技术

- shardingJDBC：基于AOP原理，在应用程序中对本地执行的SQL进行拦截，解析、改写、路由处理。需要自行编码配置实现，只支持java语言，性能较高。
- MyCat：数据库分库分表中间件，不用调整代码即可实现分库分表，支持多种语言，性能不及前者

MyCat

下载

下载地址：http://dl.mycat.org.cn/

目录结构

- bin : 存放可执行文件，用于启动停止
- mycat conf：存放mycat的配置文件
- lib：存放mycat的项目依赖包（jar）
- logs：存放mycat的日志文件

启动

启动

```
bin/mycat start
```

停止

```
bin/mycat stop
```

端口号 8066

MyCat配置

schema.xml

schema.xml 作为MyCat中最重要的配置文件之一，涵盖了MyCat的逻辑库 、 逻辑表 、 分片规 则、分片节点及数据源的配置。

标签：

- schema标签
- datanode标签
- datahost标签

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">

    <schema name="TESTDB" checkSQLschema="true" sqlMaxLimit="100"
randomDataNode="dn1">
        <!-- auto sharding by id (long) -->
        <!--splitTableNames 启用<table name 属性使用逗号分割配置多个表,即多个表使用这个配
置-->
<!--fetchStoreNodeByJdbc 启用ER表使用JDBC方式获取DataNode-->
        <table name="customer" primaryKey="id" dataNode="dn1,dn2"
rule="sharding-by-intfile" autoIncrement="true" fetchStoreNodeByJdbc="true">
            <childTable name="customer_addr" primaryKey="id"
joinKey="customer_id" parentKey="id"> </childTable>
        </table>
        <!-- <table name="oc_call" primaryKey="ID" dataNode="dn1$0-743"
rule="latest-month-calldate"
            /> -->
    </schema>



    <!-- <dataNode name="dn1$0-743" dataHost="localhost1" database="db$0-743"
        /> -->
    <dataNode name="dn1" dataHost="localhost1" database="db1" />
    <dataNode name="dn2" dataHost="localhost1" database="db2" />
    <dataNode name="dn3" dataHost="localhost1" database="db3" />
    <!--<dataNode name="dn4" dataHost="sequoiadb1" database="SAMPLE" />
     <dataNode name="jdbc_dn1" dataHost="jdbchost" database="db1" />
    <dataNode   name="jdbc_dn2" dataHost="jdbchost" database="db2" />
    <dataNode name="jdbc_dn3"   dataHost="jdbchost" database="db3" /> -->



    <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"
              writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1"
 slaveThreshold="100">
        <heartbeat>select user()</heartbeat>
        <!-- can have multi write hosts -->
```

```
        <writeHost host="hostM1" url="jdbc:mysql://localhost:3306" user="root"
                   password="root">
        </writeHost>
        <!-- <writeHost host="hostM2" url="localhost:3316" user="root"
password="123456"/> -->
    </dataHost>
    <!--
        <dataHost name="sequoiadb1" maxCon="1000" minCon="1" balance="0"
dbType="sequoiadb" dbDriver="jdbc">
        <heartbeat>         </heartbeat>
         <writeHost host="hostM1"
url="sequoiadb://1426587161.dbaas.sequoialab.net:11920/SAMPLE" user="jifeng"
password="jifeng"></writeHost>
         </dataHost>

      <dataHost name="oracle1" maxCon="1000" minCon="1" balance="0"
writeType="0"   dbType="oracle" dbDriver="jdbc"> <heartbeat>select 1 from
dual</heartbeat>
        <connectionInitSql>alter session set nls_date_format='yyyy-mm-dd
hh24:mi:ss'</connectionInitSql>
        <writeHost host="hostM1" url="jdbc:oracle:thin:@127.0.0.1:1521:nange"
user="base"   password="123456" > </writeHost> </dataHost>

        <dataHost name="jdbchost" maxCon="1000"     minCon="1" balance="0"
writeType="0" dbType="mongodb" dbDriver="jdbc">
        <heartbeat>select   user()</heartbeat>
        <writeHost host="hostM" url="mongodb://192.168.0.99/test" user="admin"
password="123456" ></writeHost> </dataHost>

        <dataHost name="sparksql" maxCon="1000" minCon="1" balance="0"
dbType="spark" dbDriver="jdbc">
        <heartbeat> </heartbeat>
         <writeHost host="hostM1" url="jdbc:hive2://feng01:10000" user="jifeng"
password="jifeng"></writeHost> </dataHost> -->

    <!-- <dataHost name="jdbchost" maxCon="1000" minCon="10" balance="0"
dbType="mysql"
        dbDriver="jdbc"> <heartbeat>select user()</heartbeat> <writeHost
host="hostM1"
        url="jdbc:mysql://localhost:3306" user="root" password="123456">
</writeHost>
        </dataHost> -->
</mycat:schema>
```

schema标签

```
<schema name="TESTDB" checkSQLschema="true" sqlMaxLimit="100"
randomDataNode="dn1">
        <!-- auto sharding by id (long) -->
        <!--splitTableNames 启用<table name 属性使用逗号分割配置多个表,即多个表使用这个配
置-->
<!--fetchStoreNodeByJdbc 启用ER表使用JDBC方式获取DataNode-->
        <table name="customer" primaryKey="id" dataNode="dn1,dn2"
rule="sharding-by-intfile" autoIncrement="true" fetchStoreNodeByJdbc="true">
            <childTable name="customer_addr" primaryKey="id"
joinKey="customer_id" parentKey="id"> </childTable>
        </table>
        <!-- <table name="oc_call" primaryKey="ID" dataNode="dn1$0-743"
rule="latest-month-calldate"
            /> -->
    </schema>
```

schema 标签用于定义 MyCat实例中的逻辑库 , 一个MyCat实例中, 可以有多个逻辑库 , 可以通 过 schema 标签来划分不同的逻辑库。MyCat中的逻辑库的概念，等同于MySQL中的database概念 , 需要操作某个逻辑库下的表时, 也需要切换逻辑库(use xxx)。

属性：

- name：指定自定义的逻辑库库名
- checkSQLschema：在SQL语句操作时指定了数据库名称，执行时是否自动去除；true：自动去除，false：不自动去除
- sqlMaxLimit：如果未指定limit进行查询，列表查询模式查询多少条记录

table 标签定义了MyCat中逻辑库schema下的逻辑表 , 所有需要拆分的表都需要在table标签中定义

属性：

- name：定义逻辑表表名，在该逻辑库下唯一
- dataNode：定义逻辑表所属的dataNode，该属性需要与dataNode标签中name对应；多个dataNode逗号分隔
- rule：分片规则的名字，分片规则名字是在rule.xml中定义的
- primaryKey：逻辑表对应真实表的主键
- type：逻辑表的类型，目前逻辑表只有全局表和普通表，如果未配置，就是普通表；全局表，配置为 global
- fetchStoreNodeByJdbc：启用ER表使用JDBC方式获取DataNode

datanode标签

```
<!-- <dataNode name="dn1$0-743" dataHost="localhost1" database="db$0-743"
        /> -->
    <dataNode name="dn1" dataHost="localhost1" database="db1" />
    <dataNode name="dn2" dataHost="localhost1" database="db2" />
    <dataNode name="dn3" dataHost="localhost1" database="db3" />
    <!--<dataNode name="dn4" dataHost="sequoiadb1" database="SAMPLE" />
     <dataNode name="jdbc_dn1" dataHost="jdbchost" database="db1" />
    <dataNode   name="jdbc_dn2" dataHost="jdbchost" database="db2" />
    <dataNode name="jdbc_dn3"   dataHost="jdbchost" database="db3" /> -->
```

属性：

- name：定义数据节点名称
- dataHost：数据库实例主机名称，引用自 dataHost 标签中name属性
- database：定义分片所属数据库

datahost标签

```
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"
              writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1"
 slaveThreshold="100">
        <heartbeat>select user()</heartbeat>
        <!-- can have multi write hosts -->
        <writeHost host="hostM1" url="jdbc:mysql://localhost:3306" user="root"
                   password="root">
        </writeHost>
        <!-- <writeHost host="hostM2" url="localhost:3316" user="root"
password="123456"/> -->
    </dataHost>
```

该标签在MyCat逻辑库中作为底层标签存在, 直接定义了具体的数据库实例、读写分离、心跳语句。

属性：

- name：唯一标识，供上层标签使用
- maxCon/minCon：最大连接数/最小连接数
- balance：负载均衡策略，取值 0,1,2,3
- writeType：写操作分发方式（0：写操作转发到第一个writeHost，第一个挂了，切换到第二 个；1：写操作随机分发到配置的writeHost）
- dbDriver：数据库驱动，支持 native、jdbc，mysql8.0以后的版本建议使用jdbc

rule.xml

rule.xml中定义所有拆分表的规则, 在使用过程中可以灵活的使用分片算法, 或者对同一个分片算法 使用不同的参数, 它让分片过程可配置化。主要包含两类标签：tableRule、Function。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mycat:rule SYSTEM "rule.dtd">
<mycat:rule xmlns:mycat="http://io.mycat/">
    <tableRule name="rule1">
        <rule>
            <columns>id</columns>
            <algorithm>func1</algorithm>
        </rule>
    </tableRule>

    <tableRule name="sharding-by-date">
        <rule>
            <columns>createTime</columns>
            <algorithm>partbyday</algorithm>
        </rule>
    </tableRule>

    <tableRule name="rule2">
        <rule>
            <columns>user_id</columns>
            <algorithm>func1</algorithm>
        </rule>
    </tableRule>

    <tableRule name="sharding-by-intfile">
        <rule>
            <columns>sharding_id</columns>
            <algorithm>hash-int</algorithm>
        </rule>
    </tableRule>
    <tableRule name="auto-sharding-long">
        <rule>
            <columns>id</columns>
            <algorithm>rang-long</algorithm>
        </rule>
    </tableRule>
    <tableRule name="mod-long">
        <rule>
            <columns>id</columns>
            <algorithm>mod-long</algorithm>
        </rule>
    </tableRule>
    <tableRule name="sharding-by-murmur">
        <rule>
            <columns>id</columns>
            <algorithm>murmur</algorithm>
        </rule>
    </tableRule>
    <tableRule name="crc32slot">
        <rule>
```

```
            <columns>id</columns>
            <algorithm>crc32slot</algorithm>
        </rule>
    </tableRule>
    <tableRule name="sharding-by-month">
        <rule>
            <columns>create_time</columns>
            <algorithm>partbymonth</algorithm>
        </rule>
    </tableRule>
    <tableRule name="latest-month-calldate">
        <rule>
            <columns>calldate</columns>
            <algorithm>latestMonth</algorithm>
        </rule>
    </tableRule>

    <tableRule name="auto-sharding-rang-mod">
        <rule>
            <columns>id</columns>
            <algorithm>rang-mod</algorithm>
        </rule>
    </tableRule>

    <tableRule name="jch">
        <rule>
            <columns>id</columns>
            <algorithm>jump-consistent-hash</algorithm>
        </rule>
    </tableRule>




    <function name="murmur"
              class="io.mycat.route.function.PartitionByMurmurHash">
        <property name="seed">0</property><!-- 默认是0 -->
        <property name="count">2</property><!-- 要分片的数据库节点数量，必须指定，否则
没法分片 -->
        <property name="virtualBucketTimes">160</property><!-- 一个实际的数据库节点
被映射为这么多虚拟节点，默认是160倍，也就是虚拟节点数是物理节点数的160倍 -->
        <!-- <property name="weightMapFile">weightMapFile</property> 节点的权重，没
有指定权重的节点默认是1。以properties文件的格式填写，以从0开始到count-1的整数值也就是节点索引
为key，以节点权重值为值。所有权重值必须是正整数，否则以1代替 -->
        <!-- <property name="bucketMapPath">/etc/mycat/bucketMapPath</property>
            用于测试时观察各物理节点与虚拟节点的分布情况，如果指定了这个属性，会把虚拟节点的
murmur hash值与物理节点的映射按行输出到这个文件，没有默认值，如果不指定，就不会输出任何东西 --
>
    </function>

    <function name="crc32slot"
              class="io.mycat.route.function.PartitionByCRC32PreSlot">
        <property name="count">2</property><!-- 要分片的数据库节点数量，必须指定，否则
没法分片 -->
    </function>
    <function name="hash-int"
              class="io.mycat.route.function.PartitionByFileMap">
        <property name="mapFile">partition-hash-int.txt</property>
```

```
    </function>
    <function name="rang-long"
              class="io.mycat.route.function.AutoPartitionByLong">
        <property name="mapFile">autopartition-long.txt</property>
    </function>
    <function name="mod-long" class="io.mycat.route.function.PartitionByMod">
        <!-- how many data nodes -->
        <property name="count">3</property>
    </function>

    <function name="func1" class="io.mycat.route.function.PartitionByLong">
        <property name="partitionCount">8</property>
        <property name="partitionLength">128</property>
    </function>
    <function name="latestMonth"
              class="io.mycat.route.function.LatestMonthPartion">
        <property name="splitOneDay">24</property>
    </function>
    <function name="partbymonth"
              class="io.mycat.route.function.PartitionByMonth">
        <property name="dateFormat">yyyy-MM-dd</property>
        <property name="sBeginDate">2015-01-01</property>
    </function>


    <function name="partbyday"
              class="io.mycat.route.function.PartitionByDate">
        <property name="dateFormat">yyyy-MM-dd</property>
        <property name="sNaturalDay">0</property>
        <property name="sBeginDate">2014-01-01</property>
        <property name="sEndDate">2014-01-31</property>
        <property name="sPartionDay">10</property>
    </function>

    <function name="rang-mod"
class="io.mycat.route.function.PartitionByRangeMod">
        <property name="mapFile">partition-range-mod.txt</property>
    </function>

    <function name="jump-consistent-hash"
class="io.mycat.route.function.PartitionByJumpConsistentHash">
        <property name="totalBuckets">3</property>
    </function>
</mycat:rule>
```

server.xml

server.xml配置文件包含了MyCat的系统配置信息，主要有两个重要的标签：system、user。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE mycat:server SYSTEM "server.dtd">
<mycat:server xmlns:mycat="http://io.mycat/">



    <system>
    <property name="nonePasswordLogin">0</property> <!-- 0为需要密码登陆、1为不需要密码登陆 ,默认为0，设置为1则需要指定默认账户-->
    <property name="ignoreUnknownCommand">0</property><!-- 0遇上没有实现的报文(Unknown command:),就会报错、1为忽略该报文，返回ok报文。
    在某些mysql客户端存在客户端已经登录的时候还会继续发送登录报文,mycat会报错,该设置可以绕过这个错误-->
    <property name="useHandshakeV10">1</property>
    <property name="removeGraveAccent">1</property>
    <property name="useSqlStat">0</property>  <!-- 1为开启实时统计、0为关闭 -->
    <property name="useGlobleTableCheck">0</property>  <!-- 1为开启全加班一致性检测、0为关闭 -->
    <property name="sqlExecuteTimeout">300</property>  <!-- SQL 执行超时 单位:秒-->
        <property name="sequenceHandlerType">1</property>
    <!--<property name="sequnceHandlerPattern">(?:(\s*next\s+value\s+for\s*MYCATSEQ_(\w+))(,|\)|\s)*)+</property>
    INSERT INTO `travelrecord` (`id`,user_id) VALUES ('next value for MYCATSEQ_GLOBAL',"xxx");
    -->
    <!--必须带有MYCATSEQ_或者 mycatseq_进入序列匹配流程 注意MYCATSEQ_有空格的情况-->
    <property name="sequnceHandlerPattern">(?:(\s*next\s+value\s+for\s*MYCATSEQ_(\w+))(,|\)|\s)*)+</property>
    <property name="subqueryRelationshipCheck">false</property> <!-- 子查询中存在关联查询的情况下,检查关联字段中是否有分片字段 .默认 false -->
    <property name="sequenceHanlderClass">io.mycat.route.sequence.handler.HttpIncrSequenceHandler</property>
      <!--  <property name="useCompression">1</property>--> <!--1为开启mysql压缩协议-->
        <!--  <property name="fakeMySQLVersion">5.6.20</property>--> <!--设置模拟的MySQL版本号-->
    <!-- <property name="processorBufferChunk">40960</property> -->
    <!--
    <property name="processors">1</property>
    <property name="processorExecutor">32</property>
     -->
        <!--默认为type 0: DirectByteBufferPool | type 1 ByteBufferArena | type 2 NettyBufferPool -->
        <property name="processorBufferPoolType">0</property>
        <!--默认是65535 64K 用于sql解析时最大文本长度 -->
        <!--<property name="maxStringLiteralLength">65535</property>-->
        <!--<property name="sequenceHandlerType">0</property>-->
        <!--<property name="backSocketNoDelay">1</property>-->
        <!--<property name="frontSocketNoDelay">1</property>-->
        <!--<property name="processorExecutor">16</property>-->
        <!--
            <property name="serverPort">8066</property> <property name="managerPort">9066</property>
            <property name="idleTimeout">300000</property> <property name="bindIp">0.0.0.0</property>
            <property name="dataNodeIdleCheckPeriod">300000</property> 5 * 60 * 1000L; //连接空闲检查
            <property name="frontWriteQueueSize">4096</property> <property name="processors">32</property> -->
```

```
        <!--分布式事务开关，0为不过滤分布式事务，1为过滤分布式事务（如果分布式事务内只涉及全局表，则不过滤），2为不过滤分布式事务,但是记录分布式事务日志-->
        <property name="handleDistributedTransactions">0</property>

            <!--
            off heap for merge/order/group/limit      1开启   0关闭
        -->
        <property name="useOffHeapForMerge">0</property>

        <!--
            单位为m
        -->
        <property name="memoryPageSize">64k</property>

        <!--
            单位为k
        -->
        <property name="spillsFileBufferSize">1k</property>

        <property name="useStreamOutput">0</property>

        <!--
            单位为m
        -->
        <property name="systemReserveMemorySize">384m</property>


        <!--是否采用zookeeper协调切换  -->
        <property name="useZKSwitch">false</property>

        <!-- XA Recovery Log日志路径 -->
        <!--<property name="XARecoveryLogBaseDir">./</property>-->

        <!-- XA Recovery Log日志名称 -->
        <!--<property name="XARecoveryLogBaseName">tmlog</property>-->
        <!--如果为 true的话 严格遵守隔离级别,不会在仅仅只有select语句的时候在事务中切换连接-->
        <property name="strictTxIsolation">false</property>
        <!--如果为0的话,涉及多个DataNode的catlet任务不会跨线程执行-->
        <property name="parallExecute">0</property>
    </system>



    <!-- 全局SQL防火墙设置 -->
    <!--白名单可以使用通配符%或着*-->
    <!--例如<host host="127.0.0.*" user="root"/>-->
    <!--例如<host host="127.0.*" user="root"/>-->
    <!--例如<host host="127.*" user="root"/>-->
    <!--例如<host host="1*7.*" user="root"/>-->
    <!--这些配置情况下对于127.0.0.1都能以root账户登录-->
    <!--
    <firewall>
       <whitehost>
          <host host="1*7.0.0.*" user="root"/>
       </whitehost>
       <blacklist check="false">
       </blacklist>
```

```
    </firewall>
    -->

    <user name="root" defaultAccount="true">
        <property name="password">123456</property>
        <property name="schemas">TESTDB</property>
        <property name="defaultSchema">TESTDB</property>
        <!--No MyCAT Database selected 错误前会尝试使用该schema作为schema，不设置则为
null,报错 -->

        <!-- 表级 DML 权限设置 -->
        <!--
        <privileges check="false">
            <schema name="TESTDB" dml="0110" >
                <table name="tb01" dml="0000"></table>
                <table name="tb02" dml="1111"></table>
            </schema>
        </privileges>
         -->
    </user>

    <user name="user">
        <property name="password">user</property>
        <property name="schemas">TESTDB</property>
        <property name="readOnly">true</property>
        <property name="defaultSchema">TESTDB</property>
    </user>

</mycat:server>
```

system标签

属性	取值	含义
charset	utf8	设置Mycat的字符集, 字符集需要与MySQL的 字符集保持一致
nonePasswordLogin	0,1	0为需要密码登陆、1为不需要密码登陆 ,默认 为0，设置为1则需要指定默认账户
ignoreUnknownCommand	0,1	0遇上没有实现的报文(Unknown command:),就会报错、1为忽略该报文，返回ok报文。 在某些mysql客户端存在客户端已经登录的时候还会继续发送登录报文,mycat会报错,该设置可以绕过这个错误
useSqlStat	0,1	为开启实时统计、0为关闭。开启之后, MyCat会自动统计SQL语句的执行 情况 ; mysql -h 127.0.0.1 -P 9066 -u root -p 查看MyCat执行的SQL, 执行 效率比较低的SQL , SQL的整体执行情况、读 写比例等
useGlobleTableCheck	0,1	1为开启全加班一致性检测、0为关闭
sqlExecuteTimeout	1000	SQL语句执行的超时时间 , 单位为 s ;
sequenceHandlerType	0,1,2	用来指定Mycat全局序列类型，0 为本地文件，1 为数据库方式，2 为时间戳列方式，默认使用本地文件方式，文件方式主要用于测试
sequnceHandlerPattern	正则表达式	必须带有MYCATSEQ或者 mycatseq进入序列 匹配流程 注意MYCATSEQ_有空格的情况
subqueryRelationshipCheck	true,false	子查询中存在关联查询的情况下,检查关联字 段中是否有分片字段 默认 false
useCompression	0,1	开启mysql压缩协议 , 0 : 关闭, 1 : 开 启
fakeMySQLVersion	5.5,5.6	设置模拟的MySQL版本号
defaultSqlParser		由于MyCat的最初版本使用了FoundationDB 的SQL解析器, 在MyCat1.3后增加了Druid 解析器, 所以要设置defaultSqlParser属 性来指定默认的解析器; 解析器有两个 : druidparser 和 fdbparser, 在 MyCat1.4之后,默认是druidparser, fdbparser已经废除了
processors	1,2....	指定系统可用的线程数量, 默认值为CPU核心 x 每个核心运行线程数量; processors 会 影响 processorBufferPool, processorBufferLocalPercent, processorExecutor属性, 所有, 在性能 调优时, 可以适当地修改processors值
processorBufferChunk		指定每次分配Socket Direct Buffer默认 值为4096字节, 也会影响BufferPool长度, 如果一次性获取字节过多而导致buffer不够 用, 则会出现警告, 可以调大该值

属性	取值	含义
processorExecutor	32	指定NIOProcessor上共享 businessExecutor固定线程池的大小; MyCat把异步任务交给businessExecutor 线程池中, 在新版本的MyCat中这个连接池使 用频次不高, 可以适当地把该值调小
packetHeaderSize		指定MySQL协议中的报文头长度, 默认4个字 节
maxPacketSize		指定MySQL协议可以携带的数据最大大小, 默 认值为16M
idleTimeout	30	指定连接的空闲时间的超时长度;如果超时,将 关闭资源并回收, 默认30分钟
txIsolation	1,2,3,4	初始化前端连接的事务隔离级别,默认为REPEATED_READ , 对应数字为3 READ_UNCOMMITED=1; READ_COMMITTED=2; REPEATED_READ=3; SERIALIZABLE=4;
sqlExecuteTimeout	300	执行SQL的超时时间, 如果SQL语句执行超时, 将关闭连接; 默认300秒;
serverPor	8066	定义MyCat的使用端口, 默认8066
managerPort	9066	定义MyCat的管理端口, 默认9066
useZKSwitch	true,false	是否采用zookeeper协调切换
XARecoveryLogBaseDir	./	XA Recovery Log日志路径
XARecoveryLogBaseName	tmlog	XA Recovery Log日志名称
strictTxIsolation	true,false	如果为 true的话 严格遵守隔离级别,不会在仅仅只有select语句的时候在事务中切换连接
parallExecute	0	如果为0的话,涉及多个DataNode的catlet任务不会跨线程执行

firewall标签

全局SQL防火墙设置

```
    <firewall>
       <whitehost>
          <host host="1*7.0.0.*" user="root"/>
       </whitehost>
       <blacklist check="false">
       </blacklist>
    </firewall>
```

whitehost：白名单，

user标签

配置MyCat中的用户、访问密码，以及用户针对于逻辑库、逻辑表的权限信息

```
   <user name="root" defaultAccount="true">
        <property name="password">123456</property>
        <property name="schemas">TESTDB</property>
        <property name="defaultSchema">TESTDB</property>
        <!--No MyCAT Database selected 错误前会尝试使用该schema作为schema，不设置则为
null,报错 -->

        <!-- 表级 DML 权限设置 -->
        <!--
        <privileges check="false">
            <schema name="TESTDB" dml="0110" >
                <table name="tb01" dml="0000"></table>
                <table name="tb02" dml="1111"></table>
            </schema>
        </privileges>
         -->
    </user>

    <user name="user">
        <property name="password">user</property>
        <property name="schemas">TESTDB</property>
        <property name="readOnly">true</property>
        <property name="defaultSchema">TESTDB</property>
    </user>
```

- name：用户名
- password：密码
- schemas：用户可以访问的逻辑库，多个库可以用逗号分隔
- privileges check：是否开启表级权限检查
- dml：配置表的权限，四位0或者1，分别代表增、改、查、删
- readOnly：是否只读

MyCat2配置

server.json

服务相关的配置

```
{
  "loadBalance":{
    "defaultLoadBalance":"BalanceRandom",
```

```
    "loadBalances":[]
  },
  "mode":"local",
  "properties":{},
  "server":{
    "bufferPool":{

    },
    "idleTimer":{
      "initialDelay":3,
      "period":60000,
      "timeUnit":"SECONDS"
    },
    "ip":"0.0.0.0",
    "mycatId":1,
    "port":8066,
    "reactorNumber":8,
    "tempDirectory":null,
    "timeWorkerPool":{
      "corePoolSize":0,
      "keepAliveTime":1,
      "maxPendingLimit":65535,
      "maxPoolSize":2,
      "taskTimeout":5,
      "timeUnit":"MINUTES"
    },
    "workerPool":{
      "corePoolSize":1,
      "keepAliveTime":1,
      "maxPendingLimit":65535,
      "maxPoolSize":1024,
      "taskTimeout":5,
      "timeUnit":"MINUTES"
    }
  }
}
```


{用户名}.user.json

所在目录：mycat/conf/users


```
{
    "dialect":"mysql",
    "ip":null,
    "password":"123456",
    "transactionType":"proxy",
    "username":"root"
}
```

- ip：客户端访问ip，建议为空,填写后会对客户端的ip进行限制
- username：用户名
- password：密码
- isolation：设置初始化的事务隔离级别
  - READ_UNCOMMITTED:1
  - READ_COMMITTED:2
  - REPEATED_READ:3,默认
  - SERIALIZABLE:4
- transactionType：事务类型
  - proxy 本地事务,在涉及大于 1 个数据库的事务,commit 阶段失败会导致不一致,但是兼容性最好
  - xa 事务,需要确认存储节点集群类型是否支持 XA

数据源

所在目录：mycat/conf/datasources

命名：{数据源名字}.datasource.json

prototypeDs.datasource.json：

```
{
    "dbType":"mysql",
    "idleTimeout":60000,
    "initSqls":[],
    "initSqlsGetConnection":true,
    "instanceType":"READ_WRITE",
    "maxCon":1000,
    "maxConnectTimeout":3000,
    "maxRetryCount":5,
    "minCon":1,
    "name":"prototypeDs",
    "password":"123456",
    "type":"JDBC",
    "url":"jdbc:mysql://localhost:3306/mysql?
useUnicode=true&serverTimezone=Asia/Shanghai&characterEncoding=UTF-8",
    "user":"root",
    "weight":0
}
```

- dbType：数据库类型，mysql
- name：用户名
- password：密码
- type：数据源类型，默认 JDBC

- url：访问数据库地址
- idleTimeout：空闲连接超时时间
- initSqls：初始化sql
- initSqlsGetConnection：对于 jdbc 每次获取连接是否都执行 initSql
- instanceType：配置实例只读还是读写
  - 可选值：READ_WRITE,READ,WRITE
- weight ： 负载均衡权重

集群

所在目录：mycat/conf/clusters

命名：{集群名字}.cluster.json

prototype.cluster.json：

```
{
    "clusterType":"MASTER_SLAVE",
    "heartbeat":{
        "heartbeatTimeout":1000,
        "maxRetry":3,
        "minSwitchTimeInterval":300,
        "slaveThreshold":0
    },
    "masters":[
        "prototypeDs"
    ],
    "maxCon":200,
    "name":"prototype",
    "readBalanceType":"BALANCE_ALL",
    "switchType":"SWITCH"
}
```

- clusterType：集群类型
  - SINGLE_NODE:单一节点
  - MASTER_SLAVE:普通主从
  - GARELA_CLUSTER:garela cluster/PXC 集群
  - MHA：MHA 集群
  - MGR：MGR 集群
- readBalanceType：查询负载均衡策略
  - BALANCE_ALL(默认值)： 获取集群中所有数据源
  - BALANCE_ALL_READ： 获取集群中允许读的数据源
  - BALANCE_READ_WRITE ： 获取集群中允许读写的数据源,但允许读的数据源优先
  - BALANCE_NONE： 获取集群中允许写数据源,即主节点中选择
- switchType：切换类型
  - NOT_SWITCH：不进行主从切换

    - SWITCH：进行主从切换
- masters：配置多个主节点,在主挂的时候会选一个检测存活的数据源作为主节点

逻辑库表

所在目录：mycat/conf/schemas

命名：{库名}.schema.json

mysql.schema.json：

```
{
    "customTables":{},
    "globalTables":{},
    "normalTables":{
        "spm_baseline":{
            "createTableSQL":"CREATE TABLE mysql.spm_baseline (\n\t`id`
bigint(22) NOT NULL AUTO_INCREMENT,\n\t`sql` longtext,\n\t`fix_plan_id`
bigint(22) DEFAULT NULL,\n\tKEY `id` (`id`)\n) ENGINE = InnoDB CHARSET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci",
            "locality":{
                "schemaName":"mysql",
                "tableName":"spm_baseline",
                "targetName":"prototype"
            }
        },
        "role_edges":{
            "createTableSQL":"CREATE TABLE mysql.role_edges (\n\t`FROM_HOST`
char(255) CHARACTER SET ascii COLLATE ascii_general_ci NOT NULL DEFAULT
'',\n\t`FROM_USER` char(32) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT
'',\n\t`TO_HOST` char(255) CHARACTER SET ascii COLLATE ascii_general_ci NOT NULL
DEFAULT '',\n\t`TO_USER` char(32) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL
DEFAULT '',\n\t`WITH_ADMIN_OPTION` enum('N', 'Y') CHARACTER SET utf8 COLLATE
utf8_general_ci NOT NULL DEFAULT 'N',\n\tPRIMARY KEY (`FROM_HOST`, `FROM_USER`,
`TO_HOST`, `TO_USER`)\n) ENGINE = InnoDB CHARSET = utf8 COLLATE = utf8_bin
STATS_PERSISTENT = 0 ROW_FORMAT = DYNAMIC COMMENT 'Role hierarchy and role
grants'",
            "locality":{
                "schemaName":"mysql",
                "tableName":"role_edges",
                "targetName":"prototype"
            }
        },
        "mycat_sequence":{
            "createTableSQL":"CREATE TABLE mysql.mycat_sequence (\n\t`NAME`
varchar(64) NOT NULL,\n\t`current_value` bigint(20) NOT NULL,\n\t`increment`
int(11) NOT NULL DEFAULT '1',\n\tPRIMARY KEY (`NAME`)\n) ENGINE = InnoDB CHARSET
= utf8mb4 COLLATE = utf8mb4_0900_ai_ci",
            "locality":{
                "schemaName":"mysql",
                "tableName":"mycat_sequence",
                "targetName":"prototype"
            }
```

```
        },
        "proc":{
            "createTableSQL":"CREATE TABLE `mysql`.`proc` (\n  `db` varchar(64)
DEFAULT NULL,\n  `name` varchar(64) DEFAULT NULL,\n  `type`
enum('FUNCTION','PROCEDURE','PACKAGE', 'PACKAGE BODY'),\n  `specific_name`
varchar(64) DEFAULT NULL,\n  `language` enum('SQL'),\n  `sql_data_access`
enum('CONTAINS_SQL', 'NO_SQL', 'READS_SQL_DATA', 'MODIFIES_SQL_DATA'),\n
`is_deterministic` enum('YES','NO'),\n  `security_type`
enum('INVOKER','DEFINER'),\n  `param_list` blob,\n  `returns` longblob,\n  `body`
longblob,\n  `definer` varchar(141),\n  `created` timestamp,\n  `modified`
timestamp,\n  `sql_mode` \tset('REAL_AS_FLOAT', 'PIPES_AS_CONCAT', 'ANSI_QUOTES',
'IGNORE_SPACE', 'IGNORE_BAD_TABLE_OPTIONS', 'ONLY_FULL_GROUP_BY',
'NO_UNSIGNED_SUBTRACTION', 'NO_DIR_IN_CREATE', 'POSTGRESQL', 'ORACLE', 'MSSQL',
'DB2', 'MAXDB', 'NO_KEY_OPTIONS', 'NO_TABLE_OPTIONS', 'NO_FIELD_OPTIONS',
'MYSQL323', 'MYSQL40', 'ANSI', 'NO_AUTO_VALUE_ON_ZERO', 'NO_BACKSLASH_ESCAPES',
'STRICT_TRANS_TABLES', 'STRICT_ALL_TABLES', 'NO_ZERO_IN_DATE', 'NO_ZERO_DATE',
'INVALID_DATES', 'ERROR_FOR_DIVISION_BY_ZERO', 'TRADITIONAL',
'NO_AUTO_CREATE_USER', 'HIGH_NOT_PRECEDENCE', 'NO_ENGINE_SUBSTITUTION',
'PAD_CHAR_TO_FULL_LENGTH', 'EMPTY_STRING_IS_NULL', 'SIMULTANEOUS_ASSIGNMENT'),\n
`comment` text,\n  `character_set_client` char(32),\n  `collation_connection`
\tchar(32),\n  `db_collation` \tchar(32),\n  `body_utf8` \tlongblob,\n
`aggregate` \tenum('NONE', 'GROUP')\n) ",
            "locality":{
                "schemaName":"mysql",
                "tableName":"proc",
                "targetName":"prototype"
            }
        },
        "spm_plan":{
            "createTableSQL":"CREATE TABLE mysql.spm_plan (\n\t`id` bigint(22)
NOT NULL AUTO_INCREMENT,\n\t`sql` longtext,\n\t`rel` longtext,\n\t`code`
longtext,\n\t`baseline_id` bigint(22) DEFAULT NULL,\n\t`accept` tinyint(1)
DEFAULT NULL,\n\tKEY `id` (`id`)\n) ENGINE = InnoDB CHARSET = utf8mb4 COLLATE =
utf8mb4_0900_ai_ci",
            "locality":{
                "schemaName":"mysql",
                "tableName":"spm_plan",
                "targetName":"prototype"
            }
        },
        "innodb_index_stats":{
            "createTableSQL":"CREATE TABLE mysql.innodb_index_stats
(\n\t`database_name` varchar(64) COLLATE utf8_bin NOT NULL,\n\t`table_name`
varchar(199) COLLATE utf8_bin NOT NULL,\n\t`index_name` varchar(64) COLLATE
utf8_bin NOT NULL,\n\t`last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP,\n\t`stat_name` varchar(64) COLLATE utf8_bin NOT
NULL,\n\t`stat_value` bigint(20) UNSIGNED NOT NULL,\n\t`sample_size` bigint(20)
UNSIGNED DEFAULT NULL,\n\t`stat_description` varchar(1024) COLLATE utf8_bin NOT
NULL,\n\tPRIMARY KEY (`database_name`, `table_name`, `index_name`,
`stat_name`)\n) ENGINE = InnoDB CHARSET = utf8 COLLATE = utf8_bin
STATS_PERSISTENT = 0 ROW_FORMAT = DYNAMIC",
            "locality":{
                "schemaName":"mysql",
                "tableName":"innodb_index_stats",
                "targetName":"prototype"
            }
        },
        "innodb_table_stats":{
```

```
            "createTableSQL":"CREATE TABLE mysql.innodb_table_stats
(\n\t`database_name` varchar(64) COLLATE utf8_bin NOT NULL,\n\t`table_name`
varchar(199) COLLATE utf8_bin NOT NULL,\n\t`last_update` timestamp NOT NULL
DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,\n\t`n_rows` bigint(20)
UNSIGNED NOT NULL,\n\t`clustered_index_size` bigint(20) UNSIGNED NOT
NULL,\n\t`sum_of_other_index_sizes` bigint(20) UNSIGNED NOT NULL,\n\tPRIMARY KEY
(`database_name`, `table_name`)\n) ENGINE = InnoDB CHARSET = utf8 COLLATE =
utf8_bin STATS_PERSISTENT = 0 ROW_FORMAT = DYNAMIC",
            "locality":{
                "schemaName":"mysql",
                "tableName":"innodb_table_stats",
                "targetName":"prototype"
            }
        }
    },
    "schemaName":"mysql",
    "shardingTables":{},
    "targetName":"prototype"
}
```

- schemaName：逻辑库名
- targetName：目的数据源或集群。targetName自动从prototype目标加载test库下的物理表或者视图作为单表,prototype 必须是mysql服务器

单表配置：

```
#单表配置
{
    "schemaName": "mysql-test",
    "normalTables": {
        "role_edges": {
        "createTableSQL":null,//可选
            "locality": {
                "schemaName": "mysql",//物理库,可选
                "tableName": "role_edges",//物理表,可选
                "targetName": "prototype"//指向集群,或者数据源
        }
    }
......
```

MyCat分片

垂直拆分

mycat配置示例：

schema.xml：

```
<schema name="SHOPPING" checkSQLschema="true" sqlMaxLimit="100">
<table name="tb_goods_base" dataNode="dn1" primaryKey="id" />
<table name="tb_goods_brand" dataNode="dn1" primaryKey="id" />
<table name="tb_goods_cat" dataNode="dn1" primaryKey="id" />
<table name="tb_goods_desc" dataNode="dn1" primaryKey="goods_id" />
<table name="tb_goods_item" dataNode="dn1" primaryKey="id" />
<table name="tb_order_item" dataNode="dn2" primaryKey="id" />
<table name="tb_order_master" dataNode="dn2" primaryKey="order_id" />
<table name="tb_order_pay_log" dataNode="dn2" primaryKey="out_trade_no" />
<table name="tb_user" dataNode="dn3" primaryKey="id" />
<table name="tb_user_address" dataNode="dn3" primaryKey="id" />
<table name="tb_areas_provinces" dataNode="dn3" primaryKey="id"/>
<table name="tb_areas_city" dataNode="dn3" primaryKey="id"/>
<table name="tb_areas_region" dataNode="dn3" primaryKey="id"/>
</schema>
<dataNode name="dn1" dataHost="dhost1" database="shopping" />
<dataNode name="dn2" dataHost="dhost2" database="shopping" />
<dataNode name="dn3" dataHost="dhost3" database="shopping" />
<dataHost name="dhost1" maxCon="1000" minCon="10" balance="0"
writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1"
slaveThreshold="100">
<heartbeat>select user()</heartbeat>
<writeHost host="master" url="jdbc:mysql://192.168.200.210:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="1234" />
</dataHost>
<dataHost name="dhost2" maxCon="1000" minCon="10" balance="0"
writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1"
slaveThreshold="100">
<heartbeat>select user()</heartbeat>
<writeHost host="master" url="jdbc:mysql://192.168.200.213:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="1234" />
</dataHost>
<dataHost name="dhost3" maxCon="1000" minCon="10" balance="0"
writeType="0" dbType="mysql" dbDriver="jdbc" switchType="1"
slaveThreshold="100">
<heartbeat>select user()</heartbeat>
<writeHost host="master" url="jdbc:mysql://192.168.200.214:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="1234" />
</dataHost>
```

server.xml：

```
<user name="root" defaultAccount="true">
<property name="password">123456</property>
<property name="schemas">SHOPPING</property>
<!-- 表级 DML 权限设置 -->
```

```
<!--
<privileges check="true">
<schema name="DB01" dml="0110" >
<table name="TB_ORDER" dml="1110"></table>
</schema>
</privileges>
-->
</user>
<user name="user">
<property name="password">123456</property>
<property name="schemas">SHOPPING</property>
<property name="readOnly">true</property>
</user>
```

全局表

```
<table name="tb_areas_provinces" dataNode="dn1,dn2,dn3" primaryKey="id"
type="global"/>
<table name="tb_areas_city" dataNode="dn1,dn2,dn3" primaryKey="id"
type="global"/>
<table name="tb_areas_region" dataNode="dn1,dn2,dn3" primaryKey="id"
type="global"/>
```

水平拆分

mycat配置示例：

schema.xml：

```
<schema name="ITCAST" checkSQLschema="true" sqlMaxLimit="100">
<table name="tb_log" dataNode="dn4,dn5,dn6" primaryKey="id" rule="mod-long" />
</schema>
<dataNode name="dn4" dataHost="dhost1" database="itcast" />
<dataNode name="dn5" dataHost="dhost2" database="itcast" />
<dataNode name="dn6" dataHost="dhost3" database="itcast" />
```

server.xml：

```
<user name="root" defaultAccount="true">
<property name="password">123456</property>
<property name="schemas">SHOPPING,ITCAST</property>
<!-- 表级 DML 权限设置 -->
<!--
<privileges check="true">
<schema name="DB01" dml="0110" >
<table name="TB_ORDER" dml="1110"></table>
</schema>
</privileges>
-->
</user>
```

分片规则

范围分片

根据指定的字段及其配置的范围与数据节点的对应情况， 来决定该数据属于哪一个分片。

规则名称：auto-sharding-long

```
<tableRule name="auto-sharding-long">
        <rule>
            <columns>id</columns>
            <algorithm>rang-long</algorithm>
        </rule>
    </tableRule>
```

```
<function name="rang-long"
              class="io.mycat.route.function.AutoPartitionByLong">
        <property name="mapFile">autopartition-long.txt</property>
    </function>
```

属性	描述
columns	标识将要分片的表字段
algorithm	指定分片函数与function的对应关系
class	指定该分片算法对应的类
mapFile	对应的外部配置文件 type 默认值为0 ; 0 表示Integer , 1 表示String
defaultNode	默认节点 默认节点的所用:枚举分片时,如果碰到不识别的枚举值, 就让它路 由到默认节点 ; 如果没有默认值,碰到不识别的则报错 。

autopartition-long.txt：

```
0-500M=0
500M-1000M=1
1000M-1500M=2
```

含义：0-500万之间的值，存储在0号数据节点(数据节点的索引从0开始)； 500万-1000万之间的 数据存储在1号数据节点； 1000万-1500万的数据节点存储在2号节点

取模分片

根据指定的字段值与节点数量进行求模运算，根据运算结果， 来决定该数据属于哪一个分片

规则名称：mod-long

```
<tableRule name="mod-long">
        <rule>
            <columns>id</columns>
            <algorithm>mod-long</algorithm>
        </rule>
    </tableRule>
```

```
<function name="mod-long" class="io.mycat.route.function.PartitionByMod">
        <!-- how many data nodes -->
        <property name="count">3</property>
    </function>
```

- columns： 标识将要分片的表字段
- algorithm： 指定分片函数与function的对应关系

- class： 指定该分片算法对应的类
- count ： 数据节点的数量

一致性hash分片

一致性哈希，相同的哈希因子计算值总是被划分到相同的分区表中，不会因为分区节点的增加而改 变原来数据的分区位置，有效的解决了分布式数据的拓容问题

先算出字段的哈希值，再根据哈希值计算要到哪一个节点

规则名称：sharding-by-murmur

```
<tableRule name="sharding-by-murmur">
        <rule>
            <columns>id</columns>
            <algorithm>murmur</algorithm>
        </rule>
    </tableRule>
```

```
<function name="murmur"
              class="io.mycat.route.function.PartitionByMurmurHash">
        <property name="seed">0</property><!-- 默认是0 -->
        <property name="count">2</property><!-- 要分片的数据库节点数量，必须指定，否则没法分片 -->
        <property name="virtualBucketTimes">160</property><!-- 一个实际的数据库节点被映射为这么多虚拟节点，默认是160倍，也就是虚拟节点数是物理节点数的160倍 -->
        <!-- <property name="weightMapFile">weightMapFile</property> 节点的权重，没有指定权重的节点默认是1。以properties文件的格式填写，以从0开始到count-1的整数值也就是节点索引为key，以节点权重值为值。所有权重值必须是正整数，否则以1代替 -->
        <!-- <property name="bucketMapPath">/etc/mycat/bucketMapPath</property>
            用于测试时观察各物理节点与虚拟节点的分布情况，如果指定了这个属性，会把虚拟节点的murmur hash值与物理节点的映射按行输出到这个文件，没有默认值，如果不指定，就不会输出任何东西 -->
    </function>
```

- columns： 标识将要分片的表字段
- algorithm： 指定分片函数与function的对应关系
- class： 指定该分片算法对应的类
- seed ： 创建murmur_hash对象的种子，默认0
- count ： 要分片的数据库节点数量，必须指定，否则没法分片
- virtualBucketTimes： 一个实际的数据库节点被映射为这么多虚拟节点，默认是160倍，也 就是虚拟节点数是物理节点数的160 倍;virtualBucketTimes*count就是虚拟结点数量

- weightMapFile： 节点的权重，没有指定权重的节点默认是1。以properties文件的 格式填写，以从0开始到count-1的整数值也就是节点索引为key， 以节点权重值为值。所有权重值必须是正整数，否则以1代替
- bucketMapPath： 用于测试时观察各物理节点与虚拟节点的分布情况，如果指定了这个 属性，会把虚拟节点的murmur hash值与物理节点的映射按行输出 到这个文件，没有默认值，如果不指定，就不会输出任何东西

枚举分片

通过在配置文件中配置可能的枚举值, 指定数据分布到不同数据节点上

规则名称：sharding-by-intfile

```
<tableRule name="sharding-by-intfile">
        <rule>
            <columns>sharding_id</columns>
            <algorithm>hash-int</algorithm>
        </rule>
    </tableRule>
```

```
<function name="hash-int"
              class="io.mycat.route.function.PartitionByFileMap">
        <property name="mapFile">partition-hash-int.txt</property>
    </function>
```

partition-hash-int.txt：

```
1=0
2=1
3=2
```

- columns： 标识将要分片的表字段
- algorithm： 指定分片函数与function的对应关系
- class： 指定该分片算法对应的类
- mapFile： 对应的外部配置文件
- type： 默认值为0 ; 0 表示Integer , 1 表示String
- defaultNode： 默认节点 ; 小于0 标识不设置默认节点 , 大于等于0代表设置默认节点 ; 默认节点的所用:枚举分片时,如果碰到不识别的枚举值, 就让它路由到默认节 点 ; 如果没有默认值,碰到不识别的则报错 。

应用指定算法

运行阶段由应用自主决定路由到那个分片，直接根据字符子串（必须是数字）计算分片号。

比如根据字符串的前两位计算，前两位为00时到第一个节点，前两位为01时到第二个节点，前两位为02时到第三个节点。

规则名称：sharding-by-substring

```
<tableRule name="sharding-by-substring">
    <rule>
        <columns>id</columns>
        <algorithm>sharding-by-substring</algorithm>
    </rule>
</tableRule>
```

```
<function name="sharding-by-substring"
class="io.mycat.route.function.PartitionDirectBySubString">
    <property name="startIndex">0</property> <!-- zero-based -->
    <property name="size">2</property>
    <property name="partitionCount">3</property>
    <property name="defaultPartition">0</property>
</function>
```

- columns： 标识将要分片的表字段
- algorithm： 指定分片函数与function的对应关系
- class： 指定该分片算法对应的类
- startIndex ： 字符子串起始索引
- size ： 字符长度
- partitionCount ： 分区(分片)数量
- defaultPartition： 默认分片(在分片数量定义时, 字符标示的分片编号不在分片数量内时, 使用默认分片)

示例：id=05-100000002 , 在此配置中代表根据id中从 startIndex=0，开始，截取siz=2位数字即 05，05就是获取的分区，如果没找到对应的分片则默认分配到defaultPartition 。

固定分片hash算法

该算法类似于十进制的求模运算，但是为二进制的操作，例如，取 id 的二进制低 10 位 与 1111111111 进行位 & 运算，位与运算最小值为 0000000000，最大值为1111111111，转换为十 进制，也就是位于0-1023之间。

- 如果是求模，连续的值，分别分配到各个不同的分片；但是此算法会将连续的值可能分配到相同的分片，降低事务处理的难度。
- 可以均匀分配，也可以非均匀分配。
- 分片字段必须为数字类型。

规则名称：sharding-by-long-hash

```
<tableRule name="sharding-by-long-hash">
    <rule>
        <columns>id</columns>
        <algorithm>sharding-by-long-hash</algorithm>
    </rule>
</tableRule>
```

```
<!-- 分片总长度为1024，count与length数组长度必须一致； -->
<function name="sharding-by-long-hash"
class="io.mycat.route.function.PartitionByLong">
    <property name="partitionCount">2,1</property>
    <property name="partitionLength">256,512</property>
</function>
```

- columns： 标识将要分片的表字段名
- algorithm ： 指定分片函数与function的对应关系
- class ： 指定该分片算法对应的类
- partitionCount ： 分片个数列表
- partitionLength： 分片范围列表

字符串hash解析算法

截取字符串中的指定位置的子字符串, 进行hash算法， 算出分片。

规则名称：sharding-by-stringhash

```
<tableRule name="sharding-by-stringhash">
    <rule>
        <columns>name</columns>
        <algorithm>sharding-by-stringhash</algorithm>
    </rule>
</tableRule>
```

```
<function name="sharding-by-stringhash"
class="io.mycat.route.function.PartitionByString">
    <property name="partitionLength">512</property> <!-- zero-based -->
    <property name="partitionCount">2</property>
    <property name="hashSlice">0:2</property>
</function>
```

- columns： 标识将要分片的表字段
- algorithm ： 指定分片函数与function的对应关系
- class： 指定该分片算法对应的类
- partitionLength： hash求模基数 ; length*count=1024 (出于性能考虑)
- partitionCount ： 分区数
- hashSlice： hash运算位 , 根据子字符串的hash运算 ; 0 代表 str.length() , -1 代表 str.length()-1 , 大于0只代表数字自身 ; 可以理解 为substring（start，end） ， start为0则只表示0

按天分片算法

按照日期及对应的时间周期来分片。

规则名称：sharding-by-date

```
<tableRule name="sharding-by-date">
        <rule>
            <columns>createTime</columns>
            <algorithm>partbyday</algorithm>
        </rule>
    </tableRule>
```

```
<function name="partbyday"
              class="io.mycat.route.function.PartitionByDate">
        <property name="dateFormat">yyyy-MM-dd</property>
        <property name="sNaturalDay">0</property>
        <property name="sBeginDate">2014-01-01</property>
        <property name="sEndDate">2014-01-31</property>
        <property name="sPartionDay">10</property>
    </function>
<!--
从开始时间开始，每10天为一个分片，到达结束时间之后，会重复开始分片插入
配置表的 dataNode 的分片，必须和分片规则数量一致，例如 2022-01-01 到 2022-12-31 ，每
10天一个分片，一共需要37个分片。
-->
```

- columns 标识将要分片的表字段
- algorithm 指定分片函数与function的对应关系
- class 指定该分片算法对应的类
- dateFormat 日期格式
- sBeginDate 开始日期
- sEndDate 结束日期，如果配置了结束日期，则代码数据到达了这个日期的分片后，会重复从开始分片插入
- sPartionDay 分区天数，默认值 10 ，从开始日期算起，每个10天一个分区

自然月分片

使用场景为按照月份来分片, 每个自然月为一个分片

规则名称：sharding-by-month

```
<tableRule name="sharding-by-month">
        <rule>
            <columns>create_time</columns>
            <algorithm>partbymonth</algorithm>
        </rule>
    </tableRule>
```

```
<function name="partbymonth"
              class="io.mycat.route.function.PartitionByMonth">
        <property name="dateFormat">yyyy-MM-dd</property>
        <property name="sBeginDate">2015-01-01</property>
    </function>
<!--
从开始时间开始，一个月为一个分片，到达结束时间之后，会重复开始分片插入
配置表的 dataNode 的分片，必须和分片规则数量一致，例如 2022-01-01 到 2022-12-31 ，一
共需要12个分片。
-->
```

- columns 标识将要分片的表字段
- algorithm 指定分片函数与function的对应关系
- class 指定该分片算法对应的类
- dateFormat 日期格式
- sBeginDate 开始日期
- sEndDate 结束日期，如果配置了结束日期，则代码数据到达了这个日期的分片后，会重复从开始分片插入

MyCat管理及监控

MyCat管理

Mycat默认开通2个端口，可以在server.xml中进行修改。

- 8066 数据访问端口，即进行 DML 和 DDL 操作。
- 9066 数据库管理端口，即 mycat 服务管理控制功能，用于管理mycat的整个集群状态

连接：

```
mysql -p 9066 -u root -p
```

命令	含义
show @@help	查看Mycat管理工具帮助文档
show @@version	查看Mycat的版本
reload @@config	重新加载Mycat的配置文件
show @@datasource	查看Mycat的数据源信息
show @@datanode	查看MyCat现有的分片节点信息
show @@threadpool	查看Mycat的线程池信息
show @@sql	查看执行的SQL
show @@sql.sum	查看执行的SQL统计

MyCat-eye

Mycat-web(Mycat-eye)是对mycat-server提供监控服务，功能不局限于对mycat-server使 用。他通过JDBC连接对Mycat、Mysql监控，监控远程服务器(目前仅限于linux系统)的cpu、内 存、网络、磁盘。Mycat-eye运行过程中需要依赖zookeeper。

https://gitee.com/MycatOne/Mycat-Eye/

- etc：jetty配置文件
- lib ： 依赖jar包
- mycat-web：mycat-web项目
- readme.txt
- start.jar：启动jar
- start.sh ： linux启动脚本

启动：

```
sh start.sh
```

或者运行jar包

访问：http://192.168.200.210:8082/mycat

需要开启实时统计

```
<property name="useSqlStat">1</property> <!-- 1为开启实时统计、0为关闭 -->
```

读写分离

一主一从

schema.xml配置：

```
<dataHost name="dhost" maxCon="1000" minCon="10" balance="1" writeType="0"
dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <writeHost host="master1" url="jdbc:mysql://mysql2:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="123456" >
        <readHost host="slave1" url="jdbc:mysql://mysql3:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="123456" />
    </writeHost>
</dataHost>
```

writeHost代表的是写操作对应的数据库，readHost代表的是读操作对应的数据库。 所以我们要想 实现读写分离，就得配置writeHost关联的是主库，readHost关联的是从库。

而仅仅配置好了writeHost以及readHost还不能完成读写分离，还需要配置一个非常重要的负责均衡 的参数 balance，取值有4种，具体含义如下：

- 0： 不开启读写分离机制 , 所有读操作都发送到当前可用的writeHost上
- 1： 全部的readHost 与 备用的writeHost 都参与select 语句的负载均衡（主要针对 于双主双从模式）
- 2 ： 所有的读写操作都随机在writeHost , readHost上分发
- 3： 所有的读请求随机分发到writeHost对应的readHost上执行, writeHost不负担读压力

双主双从

一个主机 Master1 用于处理所有写请求，它的从机 Slave1 和另一台主机 Master2 还有它的从 机 Slave2 负责所有读请求。当 Master1 主机宕机后，Master2 主机负责写请求，Master1 、 Master2 互为备机。

schema.xml：

```
```

```
<schema name="ITCAST_RW2" checkSQLschema="true" sqlMaxLimit="100"
dataNode="dn7">
</schema>

<dataNode name="dn7" dataHost="dhost7" database="db01" />


<dataHost name="dhost7" maxCon="1000" minCon="10" balance="1" writeType="0"
dbType="mysql" dbDriver="jdbc" switchType="1" slaveThreshold="100">
<heartbeat>select user()</heartbeat>
<writeHost host="master1" url="jdbc:mysql://master1:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="123456" >
<readHost host="slave1" url="jdbc:mysql://slave1:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="123456" />
</writeHost>
<writeHost host="master2" url="jdbc:mysql://master2:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="123456" >
<readHost host="slave2" url="jdbc:mysql://slave2:3306?
useSSL=false&amp;serverTimezone=Asia/Shanghai&amp;characterEncoding=utf8"
user="root" password="123456" />
</writeHost>
</dataHost>
```

writeType：

- 0 : 写操作都转发到第1台writeHost, writeHost1无法使用, 会切换到writeHost2上
- 1 : 所有的写操作都随机地发送到配置的writeHost上

switchType：

- -1 : 不自动切换
- 1 : 自动切换

---

end

by mao

2022 06 23
