8.MySQL SQL常用语句(DDL online-DDL DML DCL)√

8.1 DDL(Data Definition Language）数据定义语言

Mysql中存储数据简单可以分为表数据，除了表数据之外的都可以统称为元数据（针对库，表的属性）

8.1.1 DDL针对库的定义

对库的增加，修改，删除

1.建库语法

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
```

花括号是必须书写，中括号是可选

[IF NOT EXISTS] 可重复性运行操作，如果有就不执行操作，如果没有则执行操作

1.建库规范例子

创建数据库时通常要制定字符集

mysql> create database [if not exists] oldboy charset utf8mb4;

1.建库规范

1.显示的设置字符集

2. 库的名称：不要大写字母，不要数字开头，不要超过18字符，不要用内置字符串，要和业务有关。

2.查看数据库（不属于DDL语句，属于DML语句中的元数据获取）

mysql> show databases; 查看所有的数据库

mysql> show create database oldguo; 查询建库的具体情况

3.修改库语法

ALTER {DATABASE | SCHEMA} [db_name] +修改内容

3.修改库例子

修改数据库中的字符集

mysql> alter database oldguo charset utf8mb4;

4.删除库语法

DROP {DATABASE | SCHEMA} [IF EXISTS] db_name

4.删除库列子

mysql> drop database oldguo;

8.1.2 DDL针对表的定义

1.建表的语法

mysql> create table 表名称（列信息）ENGINE=INNODB CHARSET=utf8mb4;
复制表结构
mysql> create table t2 like t1;

1.建表规范例子

Bash | Copy

```
CREATE TABLE stu(
id INT NOT NULL PRIMARY KEY AUTO_INCREMENT COMMENT '学号，主键',
sname VARCHAR(32) NOT NULL COMMENT '姓名',
sage TINYINT UNSIGNED NOT NULL DEFAULT 0 COMMENT '年龄',
telnum BIGINT NOT NULL UNIQUE COMMENT '手机号',
district ENUM('bj','sh','sz','gd') NOT NULL DEFAULT 'bj' COMMENT '省份',
intime DATETIME NOT NULL DEFAULT NOW() COMMENT '入学时间'                    now()内置函数
)ENGINE=INNODB CHARSET=utf8mb4 COMMENT '学生表';
```

1.建表规范

1.表的名称不要大写字母，不要数字开头，不要超过18字符，不要用内置字符串，要和业务有关。
2.表名称设置前缀为库名索引 test_user
3.列的名称不要内置字符，要和业务有关。
4.列的个数50个以内，长度不要超过18字符
5.列的数据类型：合适的，精简的，完整的。
6..列中设置有且只用一个主键，最好是数据自增列auto_increment
7.每个列设置not null（unique唯一值不能与not null 一起使用）和注释commit，尽量不要设置外键。
8.显示的设置存储引擎engine=innodb和字符集charset=utf8mb4
9.修改表结构(Online-DDL),要在业务不繁忙去做,8.0建议使用pt-osc gh-ost.

2.查询表

1.进入数据库查看所有的表
mysql> use 库名称
mysql> show tables;
2.查看库中的表
mysql> show tables from 库名
3.查看建表语句
mysql> show create table 表名;

2.查看表中列的情况

```
mysql> desc 表名;
```

3.修改表（修改表中的列）

3.1 加列(8.0版本之后可在线修改，称为onlin-DDL)

```
mysql> alter table student add stel char(11) not null unique comment '手机号';
```

在指定列的前面或者后面加列

```
在name列的前面加上stel列
mysql> alter table student add stel char(11) not null unique comment '手机号' first;
在name列的后面加上stel列
mysql> alter table student add stel char(11) not null unique comment '手机号' after name;
```

3.2 删除列（删除列，列中的数据也会被删除）

```
mysql> alter table 表名 drop 列名
```

3.3 修改列

```
修改列中的数据类型
mysql> alter table 表名 modify + 修改列之前的属性信息 （如果不加上会被覆盖式的修改）
 alter table t1 modify snum int(11) not null;
修改列中的列名称
mysql> alter table 表名 change 原列名 新列名 + 修改列之前的属性信息 （如果不加上会被覆盖式的修改）
```

8.2 Online DDL

业务在线期间使用DDL操作，alter语句居多

- 增列，删列
- 索引的增删改查
- 数据类型，属性的更改

8.2.1 Algorithm三种算法分类（5.6版本之后自动进行选择算法）

copy算法：

copy算法介绍

```
在执行DDL操作时，会先生成一个新的临时表，再将原数据（原封不动）逐行拷贝到新表中，拷贝过程会阻塞DML操作，会加上MDL元数据锁。
如果执行DML语句，DDL语句会执行失败，且MDL锁会花费大量时间等待
使用copy算法要额外准备与修改表大小的存储空间，否则也会失败。
```

copy算法的执行过程

```
1、锁表，期间DML不可并行执行
2、生成临时表以及临时表文件（.frm .ibd）
```

3、拷贝原表数据到临时表
4、重命令临时表及文件
5、删除原表及文件
6、提交事务，释放锁

inplace算法：

inplace算法介绍

无需拷贝全表数据到新表，但可能还是需要IN-PLACE方式（原地，无需生成新的临时表）重建整表。这种情况下，在DDL的初始准备和最后结束两个阶段时通常需要加排他MDL锁（metadata lock，元数据锁），除此外，DDL期间不会阻塞DML

inplace算法具体操作

inplace主要分为三个阶段，准备阶段，执行阶段，提交阶段

准备阶段:（此时DML语句不能并行）

1.对表加元数据共享升级锁，并升级为排他锁。**（此时DML语句不能并行）**
2.在原表所在的路径下创建.frm和.ibd作为临时中转文件。这个阶段是共享锁的。**（此时DML可以并行）**
特殊情况：no-rebuild（除了创建二级索引外，只创建.frm文件。）
其中创建二级索引操作最特殊，该操作属于no-rebuild，不会生成.ibd。但实际上对.ibd文件做了修改，操作会在参数tmpdir指定路径下生成临时文件，用于存储索引排序结果。然后再合并到.ibd文件中。
3.申请row log空间（存放DDL执行阶段产生的DML操作）**（no-rebuild不需要）**

执行阶段：（此时DML可以并行）

1.释放排他锁，保留元数据共享升级锁。**（此时DML可以并行）**
2.扫描原表主键以及二级索引的所有数据页，生成 B+ 树，存储到临时文件中
3.将所有对原表的DML操作记录在日志文件row log中

提交阶段：（此时DML语句不能并行）

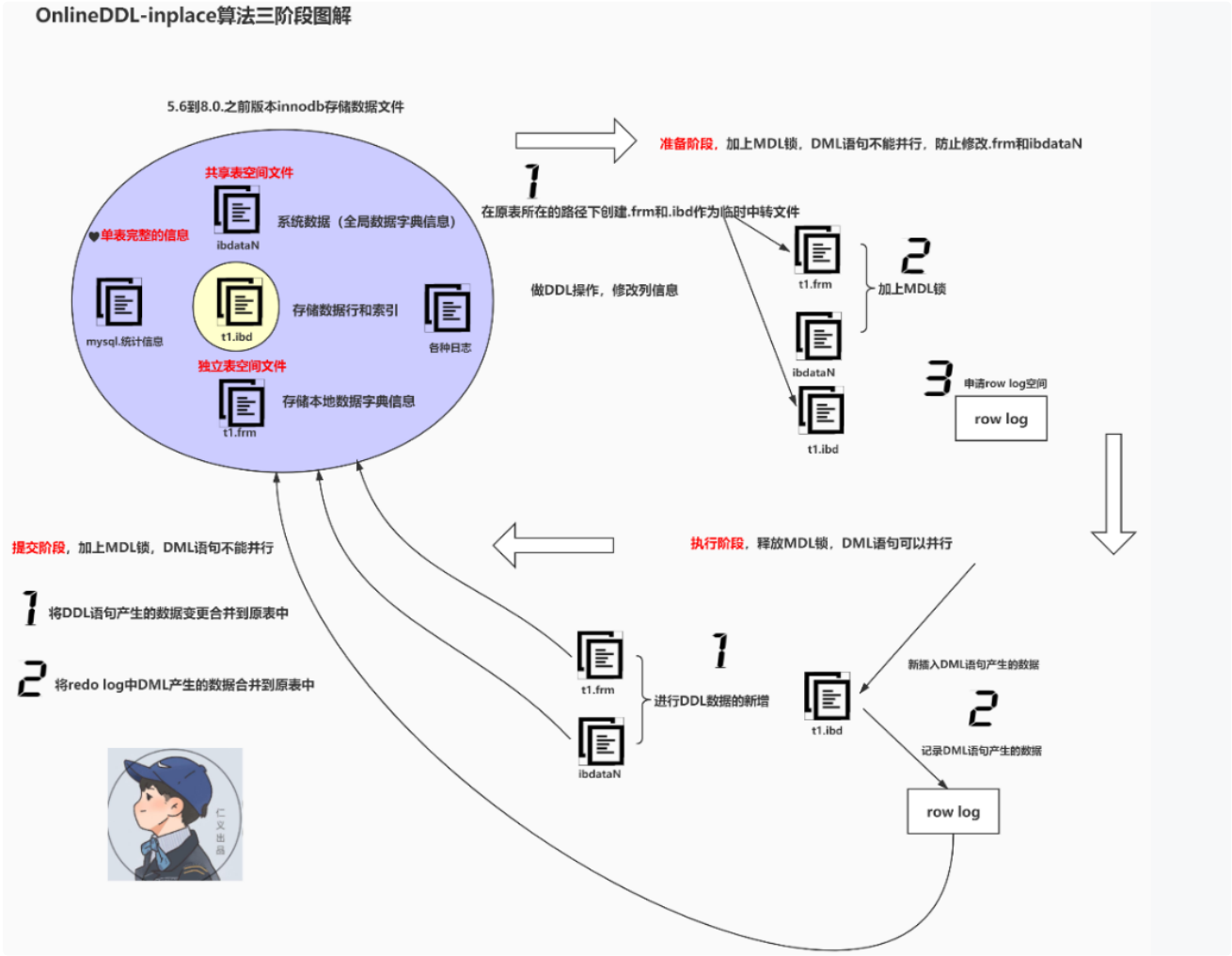
1.升级元数据共享升级锁，产生排他锁锁表。**（此时DML语句不能并行）**
2.重做row log中的内容；**（no-rebuild不需要）**
3.重命名原表文件，将临时文件改名为原表文件名，删除原表文件
4.提交事务，变更完成

inplace算法原理总结

在DDL期间产生的数据，会按照正常操作一样，写入原表，记redolog、undolog、binlog，并同步到从库去执行，只是额外会记录在row log中， 并且写入row log的操作本身也会记录redolog，而在提交阶段才进行row log重做，此阶段会锁表，此时主库（新表空间+row log）和从库（表空 间）数据是一致的，在主库DDL操作执行完成并提交，这个DDL才会写入binlog传到从库执行，在从库执行该DDL时，这个DDL对于从库本地来讲仍然是online的，也就是在从库本地直接写入数据是不会阻塞的，也会像主库一样产生row log。但是对于主库同步过来DML，此时会被阻塞，是 offline的，DDL是排他锁的在复制线程中也是一样，所以不只会阻塞该表， 而是后续所有从主库同步过来的操作（主要是在复制线程并行时会排他，同一时间只有他自己在执行）。所以大表的DDL操作，会造成同步延迟。

为什么inplace在准备阶段和提交阶段要加上MDL锁?

为了保证共享表空间ibdata1和frm两个存储系统数据字典的信息一致，数据字典信息就是我们DDL操作时改变的元数据信息，列信息等。



instant算法：

instant算法介绍

> 只需修改数据字典中的元数据，无需拷贝数据也无需重建整表，同样，也无需加排他MDL锁，原表数据也不受影响。整个DDL过程几乎是瞬间完成的，也不会阻塞DML。这个新特性是8.0.12引入的，再次感谢腾讯互娱DBA团队的贡献

8.2.2 Online DDL 常见操作模式

Operation	In-Place?	Copies Table?	Allows Concurrent DML?	Allows Concurrent Query?	Notes
添加索引	Yes*	No*	Yes	Yes	对全文索引的一些限制
删除索引	Yes	No	Yes	Yes	仅修改表的元数据
OPTIMIZE TABLE	Yes	Yes	Yes	Yes	从 5.6.17开始使用 ALGORITHM=INPLACE，当然如果指定了 `old_alter_table=1`或 mysqld启动带`--skip-new` 则将还是COPY模式。如果表上有全文索引只支持 COPY
对一列设置默认值	Yes	No	Yes	Yes	仅修改表的元数据
对一列修改auto-increment 的值	Yes	No	Yes	Yes	仅修改表的元数据
添加 foreign key constraint	Yes*	No*	Yes	Yes	为了避免拷贝表，在约束创建时会禁用 foreign_key_checks
删除 foreign key constraint	Yes	No	Yes	Yes	foreign_key_checks 不影响
改变列名	Yes*	No*	Yes*	Yes	为了允许DML并发, 如果保持相同数据类型，仅改变列名
添加列	Yes*	Yes*	Yes*	Yes	尽管允许 ALGORITHM=INPLACE ，但数据大幅重组，所以它仍然是一项昂贵的操作

Operation	In-Place?	Copies Table?	Allows Concurrent DML?	Allows Concurrent Query?	Notes
更改列顺序	Yes	Yes	Yes	Yes	尽管允许ALGORITHM=INPLACE，但数据大幅重组，所以它仍然是一项昂贵的操作
修改ROW_FORMAT和KEY_BLOCK_SIZE	Yes	Yes	Yes	Yes	尽管允许ALGORITHM=INPLACE，但数据大幅重组，所以它仍然是一项昂贵的操作
设置列属性NULL或NOT NULL	Yes	Yes	Yes	Yes	尽管允许ALGORITHM=INPLACE，但数据大幅重组，所以它仍然是一项昂贵的操作
添加主键	Yes*	Yes	Yes	Yes	尽管允许ALGORITHM=INPLACE，但数据大幅重组，所以它仍然是一项昂贵的操作。如果列定义必须转化NOT NULL，则不允许INPLACE
删除并添加主键	Yes	Yes	Yes	Yes	在同一个 ALTER TABLE 语句删除就主键、添加新主键时，才允许inplace；数据大幅重组,所以它仍然是一项昂贵的操作。
删除主键	No	Yes	No	Yes	不允许并发DML，要拷贝表，而且如果没有在同一ATLER TABLE 语句里同时添加主键则会收到限制

操作	Instant	In Place	重建表	可并行DML	只修改元数据
新增辅助索引	否	是	否	是	否
删除辅助索引	否	是	否	是	是
修改索引名	否	是	否	是	是
新增主键	否	是	是	是	否
删除主键	否	否	是	否	否
删除并同时新增主键	否	是	是	是	否
新增字段	是（追加式）	是	否	是	否
删除字段	否	是	是	是	否
修改字段数据类型	否	否	是	否	否
扩展VARCHAR列长度	否	是	否	是	是
新增STORED虚拟列	否	否	是	否	否
新增VIRTUAL虚拟列	是	是	否	是	是

转换表字符集	否	否	是	否	否
opitmize table	否	是	是	是	否
修改表名	是	是	否	是	是

新
。如

如果DDL操作涉及二级索引的创建，会在MySQL临时目录产生临时的排序文件，将中间的排序结果写入文件，最终将内容合并到最终表或索引中，然后自动删除临时排序文件。这个路径默认为mysql全局参数tmpdir指定（默认值为/tmp，如果手动指定了innodb_tmpdir参数的路径，则tmpdir会被覆盖），且不会在原始表的目录中创建临时排序文件。tmpdir需要保证能够容纳要创建的二级索引，临时排序文件最大可能需要的空间等于表中的数据量加上索引，否则执行将报错。

中间表空间：

如果DDL操作涉及rebuild表，则会在原表所在目录创建临时表空间文件（以#sql开头），临时表空间大小需要等于原表大小，重建完成后会自动重命名临时表空间，删除原表空间。所以执行rebuild操作时需要保证原表所在路径下有足够空间

8.2.4 Online DDL生成使用建议

1.一般DDL操作最好都采用pt-osc或gh-ost这样的工具来实施，并且实施之前务必要先检查当前目标表上是否有事务或大查询未结束，避免严重的MDL锁等待
2.除了8.0以上版本，除了追加式新增列、表改名、新增虚拟列这三种支持INSTANT的操作可以直接跑DDL，其余的都统统采用pt-osc/gh-osc工具.
3.执行DDL操作时，ALGORITHM选项不要手动指定。mysql会自动按照INSTANT、INPLACE、COPY的顺序自动选择合适的模式。

8.3 DCL（Data Control Language）数据控制语言

8.3.1 DCL基础语句

DCL: 用来设置或者更改数据库用户角色权限等的语句。

create user(role) 创建用户（角色）
alter user 修改用户
drop user(role) 删除用户（角色）
grant user 用户的授权
revoke user 用户权限的回收

8.3.2 8.0新特性资源组（Resource Group）

1.资源组介绍

MySQL是单进程多线程的程序，MySQL线程包括后台线程（Master Thread、IO Thread、Purge Thread等），以及用户线程。在8.0之前，
所有线程的优先级都是一样的，并且所有的线程的资源都是共享的。但是在 MySQL8.0之后，由于Resource Group特性的引入，我们可以来通过资源组的方式修改线程的优先级以及所能使用的资源，可以指定不同的线程使用特定的资源。

> 在目前版本中DBA只能操控CPU资源，并且控制的最小力度为vCPU，即操作系统逻辑CPU核数（可以通过lscpu命令查看可控制CPU总数）。
> DBA经常会遇到需要执行跑批任务的需求，这种跑批的SQL一般都是很复杂、
> 运行时间长、消耗资源多的SQL。所以很多跑批任务都是在业务低峰期的时候执行，并且在从库上执行，尽可能降低对业务产生影响。但是对于一些数据一 致性比较高的跑批任务，需要在主库上执行，在跑批任务运行的过程中很容易影响到其他线程的运行。那么现在Resource Group就是DBA的福音了，我们可以对跑批任务指定运行的资源组，限制任务使用的资源，减少对其他线程的影响。

INFORMATION_SCHEMA库下的RESOURCE_GROUPS表中记录了所有定义的资源组的情况

> mysql> select * from information_schema.resource_groups;
> MySQL8.0默认会创建两个资源组，一个是USR_default另一个是SYS_default。

PERFORMANCE_SCHEMA库下的THREADS表中，可以查看当前线程使用资源组的情况：

> mysql > select * from performance_schema.threads limit 5;

2.资源组创建

> CREATE RESOURCE GROUP oldguo
> TYPE = USER
> VCPU = 0
> THREAD_PRIORITY = 10;
> 说明：
> oldguo 为资源组名字
> type=user来源是用户端的慢SQL
> vcpu=0 给它分配到哪个CPU核上（你可以用cat /proc/cpuinfo | grep processor查看CPU有多少核），或者使用top查看哪个核心较为空闲.
> thread_priority为优先级别，范围是0到19，19是最低优先级，0是最高优先级

3.资源组应用

将创建的oldguo资源组绑定到执行的线程上，有两种方式：

方式一：

从PERFORMANCE_SCHEMA.THREADS表中查找需要绑定执行的线程ID

> mysql> select * from performance_schema.threads where
> TYPE='FOREGROUND';
> SET RESOURCE GROUP oldguo FOR 65;

方式二：

采用Optimizer Hints的方式指定SQL使用的资源组：

> mysql> SELECT /*+ RESOURCE_GROUP(oldguo) */ * FROM t2 ;

4.资源组修改及删除

> 修改资源组配置
> 可能跑批任务使用CPU资源不够，那就需要修改资源组的配置。
> ALTER RESOURCE GROUP oldguo VCPU = 10-20;
> 修改资源组优先级：

> ALTER RESOURCE GROUP oldguo THREAD_PRIORITY = 5;
> 禁止使用资源组：
> ALTER RESOURCE GROUP oldguo DISABLE FORCE;
> 删除资源组
> 对于不用的资源组可以删除
> DROP RESOURCE GROUP oldguo;

5.资源组使用限制

> Linux 平台上需要开启 CAP_SYS_NICE 特性才能使用RESOURCE GROUP
> 检查mysqld进程是否开启CAP_SYS_NICE特性
> getcap /usr/local/mysql8/bin/mysqld
> 给mysqld进程开启CAP_SYS_NICE特性
> setcap cap_sys_nice+ep /usr/local/mysql8/bin/mysqld
> 或者
> systemctl edit mysqld
> [Service]
> AmbientCapabilities=CAP_SYS_NICE
> 另外：
> mysql 线程池开启后RG失效。
> freebsd,solaris 平台thread_priority 失效。
> 目前只能绑定CPU，不能绑定其他资源。

8.4 DML(Data Manipulation Language)数据操作语言

查看表结构 desc

Bash | Copy

```
mysql> desc stu;
+---------+------------------+------+-----+---------+----------------+
| Field   | Type             | Null | Key | Default | Extra          |
+---------+------------------+------+-----+---------+----------------+
| sid     | int              | NO   | PRI | NULL    | auto_increment |
| sname   | varchar(40)      | NO   | MUL | NULL    |                |
| sage    | tinyint unsigned | NO   |     | 99      |                |
| sgender | char(1)          | NO   |     | M       |                |
| tel     | char(12)         | NO   | UNI | NULL    |                |
+---------+------------------+------+-----+---------+----------------+
```

insert 插入数据语句

1.不指定列名，默认为顺序插入，如果插入的有主键约束不能重复插入

```
mysql> insert into stu values(1,'oldboy',18,'M','111');
       insert into 表名 赋值 （列信息）
```

2.指定列名插入,(id是自增列，不插入数值可以自增)

```
mysql> insert into stu(sname,sage,sgender,tel)values('cry',18,'M','112');
       insert into 表名（列名）赋值（列信息）
```

3.指定列名，插入多行数据

```
mysql> insert into stu(sname,sage)values('oldxu',19),('oldli',20);
       insert into 表名（列名）赋值（列1信息）（列2信息）
```

4.特殊插入。存在优化问题（后面要加where语句优化）

```
mysql> create table t3 like t2;
mysql> insert into t3 select * from t2 先查t2表中所有信息，再插入时t3表时t2会面临全表锁的问题，所以后面要加where语句优化。
```

生产环境中录入数据，会整合数据后，批量插入

update 更改数据语句

update普通更改

```
mysql> update t2 set sname='olddog' where id=5; 将id=5的名字修改为olddog
       update 表名  set 列名='修改内容' where 索引列='修改的地方'
```

update精细化where修改

```
在where语句中使用and指定多个匹配条件
mysql> update stu set sage='21' where sid=3 and  sname='old';
       update 表名 set 列名='修改内容' where '修改的列既满足sid=3又满足sname=old的行进行修改'
```

update注意事项

1.update语句在使用时后面必须加上where语句，否则将会整表的数据进行修改存在危险性(逻辑损坏）。所以我们要设置一个安全参数 `sql_safe_updates`

2.update语句后面的where语句必须要指定索引列（主键索引，辅助索引等） `sql_safe_updates开启后也会限制`

update设置安全参数的方法

方法一：mysql中在线设置

> 查看sql_safe_updates参数信息，默认是关闭（0）
> mysql> select @@sql_safe_updates;
> 开启sql_safe_updates安全参数
> mysql> set global sql_safe_updates=1;

方法二：在配置文件里客户端标签下设置，重启生效

> vim /etc/my.cnf
> [mysql]
> safe_updates=1
>]# systemctl restart mysqld

delete 删除数据语句

delete语句在使用时和update语句一样需要加where,且where指定的是索引列。同样也收到安全参数的限制 `sql_safe_updates`

delete普通删除

Bash | Copy

```
delete from t2  where id=2                删除来自t2表中id值等于2的哪一行信息
delete from 表名 where 索引列='指定删除的位置点'
```

delete伪删除：

伪删除理念就是用update代替delete操作

> a. 添加状态列
> mysql> alter table stu add state tinyint not null default 1 comment '状态列,1存在,0不存在';
> b. 原删除语句替换
> 原来: delete from stu where sid=7;
> 替换为: update stu set state=0 where sid=7;
> c. 原业务语句查询替换
> 原来: select * from stu;
> 替换为: select * from stu where state=1;

select 查询数据语句

应用场景一：查询参数的值，设置用户参数变量及调用

1.查看数据库中所有的参数(589个)

> mysql> show variables ;
> 模糊匹配参数名称
> mysql> show variables like '%不完整参数名称%'

2.查看参数变量语法：`mysql> select @@参数名称;`

> 1.查看端口 mysql> `select @@port;`
> 2.查看数据目录 mysql> `select` `@@datadir` ;

3.设置用户参数变量及调用

@@ 代表系统变量 @代表用户变量

> 设置用户变量
> mysql> select @用户变量名称：=赋的值；或者mysql> set @用户变量名称：=赋的值;
> 调用用户变量
> mysql> select @设置的用户变量名称;

应用场景二：查询函数，函数运算

1.查看函数语法：`mysql> select 函数名称 () ;`

> 1.查看mysql版本 mysql> select version();
> 2.查看当前登陆用户 mysql> select user();
> 3.查看当前时间 mysql> select now();
> 4.拼接字符串 mysql> select concat();

2.拼接concat举例

Bash | Copy

```
mysql> select concat(user,'@',host) from mysql.user;
+----------------------------+
| concat(user,'@',host)      |
+----------------------------+
| oldguo@10.0.0.%            |
| mysql.infoschema@localhost |
| mysql.session@localhost    |
| mysql.sys@localhost        |
| root@localhost             |
+----------------------------+
```

3.函数运算（计算机功能）

可以加减乘除

Bash | Copy

```
加
mysql> select 1+1;
+-----+
| 1+1 |
+-----+
|   2 |
+-----+
减
mysql> select 2-1;
+-----+
| 2-1 |
+-----+
|   1 |
+-----+
乘
mysql> select 2*2;
+-----+
| 2*2 |
+-----+
|   4 |
+-----+
除
mysql> select 4/2;
+--------+
| 4/2    |
+--------+
| 2.0000 |
+--------+
```

应用场景三：标准使用

练习库文件

- t100w.sql（49.8 MB）
- school.sql（4 kB）
- world.sql（397 kB）

单表查询

1.多子句的执行顺序

```
select
    from 从哪来
    where 过滤条件
    group by 分组条件
            select_list 列条件(书写时跟在select 语句后)
  having 后过滤条件
    order by 排序条件
    limit 分页
```

2.from子句(必须项)

select + from表示指定查看的来源

> 1.查看表中所有信息 全表扫描不建议使用
> `mysql> select * from table(表名);`
> 2.查看表中指定列信息
> `mysql> select sid,sname from table;`
> 列名称 表名称

3. where子句（选择项）

select +from +where 使用

3.1 where + 比较判断符(= ,> ,< ,>= , <= , != （不等于）)

实列

```
等值查询
---查询中国所有的城市信息
SELECT * FROM city WHERE countrycode='CHN';
小于查询
---查询city表中人口数小于100的信息
mysql> select * from world.city where population <100;
```

3.2 where + 逻辑连接符(and , between, or ,in， not in)

实列

```
1.and 并且（必须两个同时满足）
---查询中国 河北省的城市信息
mysql> select * from city  where country='CHN' and district='hb';
2.between
与 and 在同一种条件时同时使用，表示范围（在...之间）
---查询中国人口数100000到200000的城市
mysql> select * from city where country='CHN' and population between 100000 and 200000
3.or 或者（两者满足一个即可）
--- 查询中国或者美国城市信息
mysql> select * from city where countrycode='CHN' or countrycode='USA';
4.in （与or类似）
--- 查询中国或者美国城市信息 （等价or）
mysql> select * from city where countrycode in ('CHN','USA');
5.not in
```

3.3 WHERE + LIKE 模糊查询

实列

注意：在实际生产环境中不要出现%%号，因为%%不走索引是全表扫描，%是走索引的

```bash
#查询城市名称是qing开头的
- '%'  '%%'
mysql> SELECT  * FROM city WHERE NAME LIKE 'qing%'
mysql> SELECT  * FROM city WHERE NAME LIKE '%qing%'
```

3.4 where +(union，union all) 合并查询结果集，查询的结果集结构相同。

实列

```bash
union all 合并查询结果
--- 查询中国或者美国城市信息
mysql> select * from city where countrycode='CHN'
union all
mysql> select * from city where countrycode='USA';
```

union，union all 两者的区别?

> union自动去除重复行，会有更多性能消耗（因为要先排序，再去重）
> union all 不排序去重，只是将结果集合并在一起

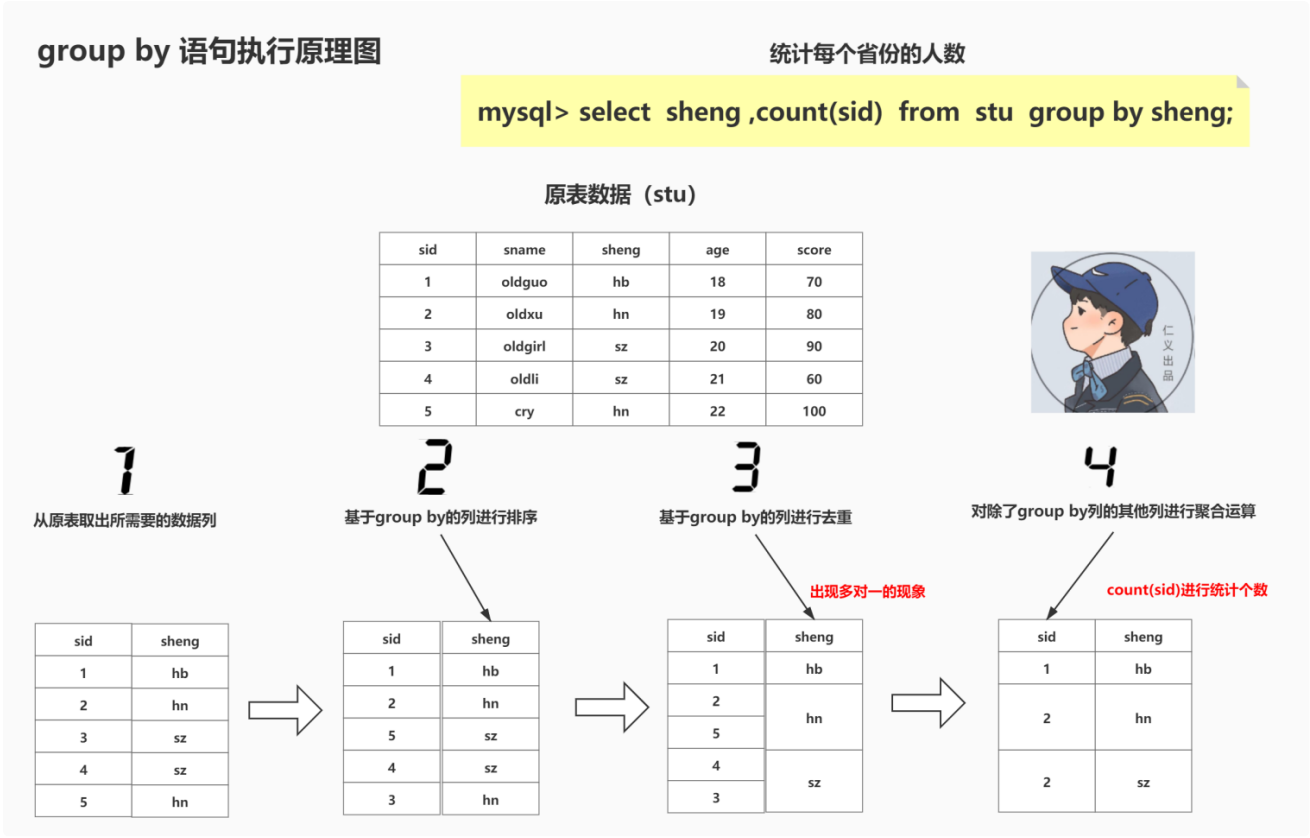
4.group by子句与聚合函数（选择项）

select +from +where + group by（分组）+ 聚合函数 使用

4.1聚合函数的种类(处理统计分析类的需求)

> 1.COUNT() ---- 统计个数
> 2.SUM() ---- 求和
> 3.AVG() ---- 求平均值
> 3.MAX() ----- 最高值
> 4.MIN() ---- 最低值
> 5.GROUP_CONCAT() ----group by列外 其他的列

4.2 group by 执行原理图



4.3 sql_mode=only_full_group_by的限制

> 1.如果select 后的查询列表,不是group by 的条件,又不在聚合函数中存在.就会和SQL_mode不兼容
> 2.如果group by后的列是主键或唯一键时,可以不使用group by，因为本来就没有重复
> 所以其他的列我们用到函数 group_concat()

5.HAVING子句 （选择项）

作用 ： 在group by 之后进行数据的过滤

Bash | Copy

```
#统计每个国家的总人口,过滤输出总人口超过5000w的信息
SELECT countrycode,SUM(population) FROM city GROUP BY countrycode HAVING SUM(population) > 50000000
```

6.order by子句 （选择项）

作用：对结果集进行排序

order by 默认是顺序 加上desc 逆序

Bash | Copy

```
#查询中国所有城市信息,并按人口数排序输出结果
select * from city
where countrycode='CHN'
order by population          #默认从小到大显示

select * from city
where countrycode='CHN'
order by population desc     #从大到小显示
```

7.limit子句（选择项）

作用：显示指定的行，默认1,0显示为 跳过1行，显示0行

```
#统计每个国家的总人口,过滤输出总人口超过5000w的信息,只显示前三名
SELECT countrycode,SUM(population)
FROM city
GROUP BY countrycode
HAVING SUM(population) >50000000
LIMIT 3
#以上条件显示4-6名
SELECT countrycode,SUM(population)
FROM city
GROUP BY countrycode
HAVING SUM(population)
LIMIT 3,3
#统计每个国家的总人口,过滤输出总人口超过5000w的信息,只显示2-7名
SELECT countrycode,SUM(population)
FROM city
GROUP BY countrycode
HAVING SUM(population) > 50000000
LIMIT 4 OFFSET 2          #跳过两行，显示4行
```

8. 别名的使用

```
列的别名：
SELECT countrycode AS guojia,SUM(population) AS renshu
FROM city
```

```
GROUP BY guojia
HAVING renshu > 50000000
order by renshu desc
LIMIT 4 OFFSET 2
表的别名：
SELECT a.countrycode,SUM(population)
FROM city AS a
GROUP BY a.countrycode
HAVING SUM(population) > 50000000
ORDER BY SUM(population)
LIMIT 4 OFFSET 2
```

DDL%20DML%20DCL%EF%BC%89%E2%88%9A%20%7C%208.1%20DDL(Data%20Definition%20Language%EF%BC%89%E6%95%B0%E6%8D%AE%E5%AE%9A%E4%B9%89%E8%AF%AD%E8%A8%80Mys