# 用户操作

## #添加新用户

CREATE USER 'shenxuefa'@'localhost' IDENTIFIED BY 'shenxuefa';

## #修改用户名

RENAME USER 'shenxuefa'@'localhost' TO 'yabushan'@'localhost';

## #修改密码

SET PASSWORD FOR 'yabushan'@'localhost'=PASSWORD('password');

## #删除用户

DROP USER 'yabushan'@'localhost' ;

## #授权用户权限

USE test ;#数据库名

GRANT SELECT ON ACTION TO 'yabushan'@'localhost';

## #新建用户并授权

USE test;

GRANT SELECT ,UPDATE ON t\_user TO 'yabushan'@'localhost' IDENTIFIED BY 'yabushan';

#授予yabushan用户在test数据库中的所有表的select权限

GRANT SELECT ON test.\* TO 'yabushan'@'localhost';

#授予yabushan在test数据库的所有操作权限

USE test;

GRANT ALL ON \* TO 'yabushan'@'localhost';

#权限的转移和限制

#grant 语句的最后可以使用with子句，如果指定为：with grant option ，则表示该用户可以授权于其他用户相同的权利

GRANT SELECT ON test.`action` TO 'shuaige'@'localhost' IDENTIFIED BY '123' WITH GRANT OPTION;

## #回收权限

REVOKE SELECT ON test.`action` FROM 'shuaige'@'localhost';

# 数据库表维护

## #表的维护语句

#更新表action 的索引的可压缩性，并显示

ANALYZE TABLE test;

#MySQL 的Optimizer（优化元件）在优化SQL语句时，首先需要收集一些相关信息，

#其中就包括表的cardinality（可以翻译为“散列程度”），

#它表示某个索引对应的列包含多少个不同的值——如果cardinality大大少于数据的实际散列程度，那么索引就基本失效了。

#我们可以使用SHOW INDEX语句来查看索引的散列程度：

SHOW INDEX FROM test.`t\_humor`;

#检查表action 是否正确

CHECK TABLE t\_humor;

#数据在传输时，可能会发生变化，也有可能因为其它原因损坏，为了保证数据的一致，我们可以计算checksum（校验值）

#使用MyISAM引擎的表会把checksum存储起来，称为live checksum，当数据发生变化时，checksum会相应变化。

#在执行Checksum Table时，可以在最后指定选项qiuck或是extended；quick表示返回存储的checksum值，而extended会重新计算checksum，如果没有指定选项，则默认使用extended。

CHECKSUM TABLE ACTION ;

## #优化表

#经常更新数据的磁盘需要整理碎片，数据库也是这样，Optimize Table语句对MyISAM和InnoDB类型的表都有效。

#如果表经常更新，就应当定期运行Optimize Table语句，保证效率。

#与Analyze Table一样，Optimize Table也可以使用local来取消写入binlog。

OPTIMIZE TABLE ACTION;

# #sql基本语法

## #创建数据库

CREATE DATABASE IF NOT EXISTS sxf;

## #删除数据库

DROP DATABASE IF EXISTS sxf;

## #创建表

CREATE TABLE IF NOT EXISTS test\_table(id INT NOT NULL);

## #删除表

DROP TABLE IF EXISTS test\_table;

## #修改表名

ALTER TABLE ACTION RENAME TO actions;

## #修改表

#增加一列

ALTER TABLE ACTION ADD COLUMN a TINYINT NULL;

#增加一列 first | after col\_name :表示在某列的前或者后添加，不指定则添加到最后

ALTER TABLE ACTION ADD COLUMN b TINYINT NULL AFTER img;

#把一个integer 列的名称从a变更到bb

ALTER TABLE ACTION CHANGE a bb TINYINT;

#改变列的类型：把bb列的类型改为bigint

ALTER TABLE ACTION MODIFY bb BIGINT NOT NULL;

#从表中删除某列

ALTER TABLE ACTION DROP COLUMN bb ;

## #复制表

创建一个和actions表结构一样的表

CREATE TABLE IF NOT EXISTS ACTION LIKE actions ;

#复制表和数据

CREATE TABLE SDF SELECT \* FROM ACTIONS ;

#插入表数据

#mysql数据库的日期与oracle的当前时间获取格式不同，获取当前时间的 使用mysql自带的函数 curdate()

INSERT INTO sdf (id,title,img,b,describle,starttime,endtime) VALUES('2','2','2','2','2', CURDATE(), CURDATE());

#若原行中存在primary key 或unique key，而插入的数据行中含有与原有行中primary key 、unique key相同的列值，则insert 语句无法插入此行，可用replace 语句，在插入之前删除原有行

REPLACE INTO sdf (id,title,img,b,describle,starttime,endtime) VALUES('2','3','2','2','2', CURDATE(), CURDATE());

#插入图片:存储图片路径

INSERT INTO img VALUES('1','d:\image\1.jpg');

#插入图片：存储图片本身

INSERT INTO img VALUES('1',LOAD\_FILE('d:\image\1.jpg'));

## #删除表数据

DELETE FROM SDF;

#清除表数据:此语句将清除表格的所有数据，且无法恢复

TRUNCATE TABLE SDF;

#修改表数据:单表修改

UPDATE sdf s SET s.`b`='ss' WHERE s.`ID`=1;

#修改表数据：多表修改

UPDATE sdf s, actions a SET s.`b`='18',a.`DESCRIBLE`='18' WHERE s.`ID`=s.`ID` AND s.`ID`='1';

#显示一些常用的信息

DESC sdf ;

## #数据库的查询

#distinct 去掉重复行

SELECT DISTINCT s.`ID` FROM sdf s ;

#case 用法

SELECT DECODE(s.`ID`,1,'你好','不好') AS d FROM sdf s;

#mysql中没有decode这个函数，但可以用case来代替

SELECT

CASE WHEN s.id=1 THEN 'sdf'

WHEN s.id=2 THEN '你好'

ELSE 'sdfeee'

END AS '登记'

FROM sdf s;

## #聚合函数

用于对一组值进行计算，然后返回单个值，常与group by一起使用，除count() 函数外，聚合函数会忽略空值

### #count()

用于统计组中满足条件的行数或总行数，返回select语句检索到的行中非null值的数目

SELECT COUNT(TITLE) FROM SDF;

### #MAX、MIN、

求表达式中所有值的最大值与最小值，可以是数字，字符和时间日期

### #SUM、AVG函数

分别用于求表达式中所有值项的总和与平均值

### #variance 和stddev函数：

分别用于计算特定的表达式中的所有值的方差和标准差

### #GROUP\_CONCAT函数：

该函数用来返回来自一个组指定列的所有非空值，这些值一个接一个放置，中间用逗号隔开，并表示为一个长长的字符串，长度标准值为1024

SELECT TITLE,SUM(ID) FROM SDF GROUP BY TITLE;

SELECT VARIANCE(ID) FROM SDF GROUP BY TITLE;

SELECT GROUP\_CONCAT(B) FROM SDF;

SELECT \* FROM SDF;

### #JOIN 、left join

（以左边的表为基表查询数据） 、right join（以右边的表为基表查询数据）

SELECT \* FROM SDF S , ACTIONS A WHERE S.`ID`=A.ID;

SELECT \* FROM SDF;

SELECT \* FROM ACTIONS;

### #exists 、not exists :

这两个函数的返回值为true、或者false 当为true时，则相当于where 1=1 为false 相当于where 1=0

SELECT \* FROM SDF S WHERE NOT EXISTS(SELECT A.`ID` FROM ACTIONS A);

### #group by子句 ：

根据字段对行进行分组 asc(升序) 、 desc(降序)

#使用rollup 操作符的group by子句，指定在结果集内不仅包含有group by提供的正常行，还包含汇总行

SELECT COUNT(1) ,TITLE FROM SDF S GROUP BY S.`ID` WITH ROLLUP;

### #having 子句

：与 where 类似。不同：where 子句用来在from 之后，而having用在group by之后

SELECT COUNT(1) ,TITLE FROM SDF S GROUP BY S.`ID` HAVING s.`TITLE`>1;

### #order by 子句 ：

关键字：asc，desc 系统默认asc

#order by子句中还可以包含子查询 ,当对空值排序时，order by 子句将空值作为最小值对待

SELECT \* FROM SDF S WHERE S.`ID` >0 ORDER BY (SELECT AVG(ID) FROM SDF);

### #limit 子句 ：

select 语句的最后面的一个子句，主要用于限制被select 语句返回的行数

SELECT \* FROM SDF S LIMIT 2;

### #union 、union all

把来自多个select语句的结果组合到一个结果集中，字段个数、名字、类型要一致。union会去掉重复的行、union all 则不会去掉重复

### #handle语句：

能够一行一行地浏览表中的数据

#使用handle语句时，要先使用handler open语句打开一个表

#再使用handler read 语句浏览打开表的行。

#浏览完后必须使用handle close 语句关闭已经打开的表

HANDLER sdf OPEN;

HANDLER sdf READ FIRST

WHERE id>1;

HANDLER sdf READ NEXT;

HANDLER sdf CLOSE;

# 视图

## #创建视图

#使用视图时，如果视图关联的基本表中添加了新字段，则该视图将不包含新字段；如果表被删除，则该视图不能使用

CREATE OR REPLACE VIEW sdf\_view

AS

SELECT s.`ID`,s.`STARTTIME` FROM sdf s WHERE s.b>5

WITH CHECK OPTION; #创建视图时，如果带着这个选项，则在插入数据是，必须满足select语句后的where的条件

SELECT \* FROM sdf\_view;

## #修改视图

ALTER VIEW sdf\_view

AS

SELECT s.`ID`,s.`STARTTIME` FROM sdf s WHERE s.b>6

WITH CHECK OPTION;

## #删除视图:

一次可删除多个视图，用逗号分隔开

DROP VIEW sdf\_view;

# #索引和数据完整性约束

## #索引的作用：

#快速读取数据；

#保证数据记录的唯一性；

#实现表与表之间的参照完整性

#在使用group by ，order by子句进行数据检索时，利用索引可以减少排序和分组的时间

#当设置一个表的主键时，会在该主键上设置一个索引

## #创建索引:

可以在一个索引

CREATE INDEX xy\_index ON sdf(id);

## #删除索引

DROP INDEX xy\_index;

## #primary 和unique的区别：

#一个表只有一个主键，但可以有多个unique 键；

#主键字段的值不允许null，而unique字段的值可以去null，但是必须使用null或者not null声明

#都会自动产生索引

# #参照完整性约束

#一种特殊的完整性约束，实现为：外键

#引用的外键必须是唯一的

#on delete | on update 为每一个外键定义参照动作

#指定采取的动作 restrict、cascade 、set null

#1. CASCADE: 从父表中删除或更新对应的行，同时自动的删除或更新自表中匹配的行。ON DELETE CANSCADE和ON UPDATE CANSCADE都被InnoDB所支持。

#2. SET NULL: 从父表中删除或更新对应的行，同时将子表中的外键列设为空。注意，这些在外键列没有被设为NOT NULL时才有效。ON DELETE SET NULL和ON UPDATE SET SET NULL都被InnoDB所支持。

#3. NO ACTION: InnoDB拒绝删除或者更新父表。

#4. RESTRICT: 拒绝删除或者更新父表。指定RESTRICT（或者NO ACTION）和忽略ON DELETE或者ON UPDATE选项的效果是一样的。

#5. SET DEFAULT: InnoDB目前不支持。

# 外键约束使用最多的两种情况无外乎：

# 1）父表更新时子表也更新，父表删除时如果子表有匹配的项，删除失败；

#2）父表更新时子表也更新，父表删除时子表匹配的项也删除。

#前一种情况，在外键定义中，我们使用ON UPDATE CASCADE ON DELETE RESTRICT；后一种情况，可以使用ON UPDATE CASCADE ON DELETE CASCADE。

#默认为restrict

ALTER TABLE tbl\_name

ADD [CONSTRAINT [symbol]] FOREIGN KEY

[index\_name] (index\_col\_name, ...)

REFERENCES tbl\_name (index\_col\_name,...)

[ON DELETE reference\_option]

[ON UPDATE reference\_option]

ALTER TABLE actions ADD CONSTRAINT FK\_ID FOREIGN KEY(ID) REFERENCES SDF(id)

ON DELETE RESTRICT

ON UPDATE RESTRICT;

#ON DELETE CASCADE

#ON UPDATE CASCADE;

## #删除外键约束

ALTER TABLE `actions` DROP FOREIGN KEY `FK\_ID`

## #check完整性约束(会被分析，但不会器作用)

#作用：指定某个字段的取值范围

ALTER TABLE ACTIONS ADD CONSTRAINT ss CHECK(TITLE IN('1','2'));

## #删除约束

ALTER TABLE actions DROP PRIMARY KEY ;

# #mysql内置函数

## #greatest() 和 least() 函数

：获取一组数中的最大值和最小值

SELECT GREATEST(10,2,3,4,5,6);

SELECT LEAST(10,2,3,4,5,6);

## #floor()和ceiling()函数:

获得小于一个数的最大整数值和大于一个数的最小整数值

SELECT FLOOR(-12.4) ,CEILING(-12.3);

## #round()

获得一个数的四舍五入的整数值

SELECT ROUND(4.5);

## #truncate():

把一个数字截取为一个指定小数个数的数字，逗号后面的数字表示指定小数的个数

SELECT TRUNCATE(1.2345,3);

## #abs()函数：

获得一个数的绝对值

SELECT ABS(-2);

## #sign()

返回的结果是正数（1），负数（-1），或者0

SELECT SIGN(1-40);

## #sqrt()

返回一个数的平方根

SELECT SQRT(3);

## #pow()

一个数作为另外一个数的指数

SELECT POW(3,3);

## #SIN()、cos()、tan()

//正弦，余弦，正切

## #bin()\otc()\hex()

SELECT BIN(5);

# #聚合函数

## #ASCII() 函数

SELECT ASCII('a');

## #char()函数

SELECT CHAR(67,68,69);

SELECT CHAR('a');

## #left(str,x) 和right函数

：返回从字符串左右边开始指定x个字符

SELECT LEFT('asdcsedfdsfsd',4);

SELECT RIGHT('asdcsedfdsfsd',4);

## #trim(),ltrim,rtrim 函数：

分别删除字符串中、前、后所有空格

SELECT TRIM(' sdfdsf dsfdf dsf ');

SELECT LTRIM(' sdfdsf dsfdf dsf ');

SELECT RTRIM(' sdfdsf dsfdf dsf ');

## #rpad() 和 lpad 函数：rpad | lpad(str,n,pad):

用字符串pad，对字符串str的左右边进行填补，直到str中字符数目达到n个，最后返回。

SELECT RPAD('asd',9,'f');

SELECT LPAD('asd',9,'f');

## #replace(str1,str2,str3)

:用字符串3替换str1中所有出现的字符串str2，最后返回替换后的字符串

SELECT REPLACE('aaabbcc','b','f');

## #concat 函数:

连接字符串

SELECT CONCAT('abc','dfs','gcx');

## #substring(exp,start,end) 函数：

返回expression中指定的部分数据，

SELECT SUBSTRING('abcdefghijklmn',3);

## #strcmp(s1,s2):

比较两个字符串，相等返回0，s1>s2返回1，s1<s2返回-1

SELECT STRCMP('ABC','DEF');

## #日期和时间函数

### #now():

已yyyy-mm-dd HH:MM:SS格式返回

SELECT NOW();

### #curtime() curdate():

分别返回当前时间和日期

SELECT CURTIME();

SELECT CURDATE();

### #year

SELECT YEAR(CURDATE());

### #month(), monthname():

以数值、字符串的形式返回

SELECT MONTH(NOW());

SELECT MONTHNAME(NOW());

### #day of year(),day of month() ,day of week()

SELECT DAYOFYEAR(NOW());

SELECT DAYOFWEEK(NOW());

SELECT DAYOFMONTH(NOW());

SELECT DAYNAME(NOW());

#WEEK()返回指定日期是一年的第几个星期

SELECT WEEK('2016-06-19');

#yearweek()：返回年+第几个星期

SELECT YEARWEEK('2016-06-19');

### #hour(),minute(),second():

返回日期的时分秒

SELECT HOUR(NOW());

SELECT MINUTE(NOW());

SELECT SECOND(NOW());

### #date\_add(),date\_sub(date,interval int keyword):

表示时间间隔，

SELECT DATE\_ADD(NOW(),INTERVAL 10 YEAR);

SELECT DATE\_ADD(NOW(),INTERVAL 10 MONTH);

SELECT DATE\_ADD(NOW(),INTERVAL 10 DAY);

SELECT DATE\_ADD(NOW(),INTERVAL 10 MINUTE);

SELECT DATE\_ADD(NOW(),INTERVAL 10 HOUR);

SELECT DATE\_ADD(NOW(),INTERVAL 10 SECOND);

SELECT DATE\_SUB(NOW(),INTERVAL 10 YEAR);

SELECT DATE\_SUB(NOW(),INTERVAL 10 MONTH);

SELECT DATE\_SUB(NOW(),INTERVAL 10 DAY);

SELECT DATE\_SUB(NOW(),INTERVAL 10 MINUTE);

SELECT DATE\_SUB(NOW(),INTERVAL 10 HOUR);

SELECT DATE\_SUB(NOW(),INTERVAL 10 SECOND);

## #加密函数

### #AES\_ENCRYPT 和 AES\_DECRYPT

SELECT AES\_ENCRYPT('abc',5);

SELECT AES\_DECRYPT(AES\_ENCRYPT('abc',5),5);

### #ENCODE 和 DECODE

SELECT ENCODE('abc',5);

SELECT DECODE(ENCODE('abc',5),5);

### #ENCRYPT

SELECT ENCRYPT('w3resource');

### #PASSWORD

SELECT PASSWORD('sdfd');

## #控制流函数

### #IFNULL(exp1,exp2)

判断exp1 是否为空，当exp1 为空，返回exp2

SELECT IFNULL('sd','sdfd');

SELECT IFNULL(NULL,'sdfd');

### # NULLIF(exp1,exp2) 函数:

判断exp1 和exp2 是否相等。相等则返回null ，不相等则返回exp1

SELECT NULLIF('s','s');

SELECT NULLIF('sa','s');

### #if(EXP1,EXP2,EXP3):

判断EXP1是否为真，是则返回EXP2,否则返回EXP3 0:不为真，非0 为真

SELECT IF(-2,'A','B');

SELECT IF(0,'A','B');

### #格式化函数FORMAT(X,Y)

,X为要格式化的数，Y为格式化保留的小数,碰到非数字，则为0

SELECT FORMAT('4.34dsf342323',3);

### #cast(exp,as type)类型转换

#支持转换的类型：binary、char、date、time、datetime、signed、unsigned

SELECT CAST('77' AS UNSIGNED);

SELECT CAST('77' AS SIGNED);

## #系统信息函数

SELECT DATABASE();

SELECT USER();

SELECT VERSION();

SELECT BENCHMARK();

#用户返回最后一个select语句返回的记录行的数目

SELECT FOUND\_ROWS();

# #过程式数据库对象

#创建存储过程:必须具备有create、ROUTINE权限

#使用存储过程的优点：

#存储过程在服务器端运行，执行速度快

#存储过程执行一次后，其执行规划就驻留在高速缓存

#保证数据库的安全性

#创建一个存储过程可能涉及到多条查询，这时可用delimiter $$ 将语句结束符设置为“$$” 而不是“;”

#declare 声明变量

#调用存储过程 call

CREATE PROCEDURE acount\_num()

SELECT COUNT(1) FROM sdf;

CALL acount\_num();

## #创建一个存储过程

根据传入的值删除一条数据

DELIMITER $$

CREATE PROCEDURE info\_delete(IN id INT,OUT str CHAR(10))

BEGIN

IF id=1 THEN

UPDATE sdf s SET s.title='sdfdf' WHERE s.id=id;

SET str='update';

END IF;

END $$

DELIMITER ;

## #删除存储过程

DROP PROCEDURE IF EXISTS info\_delete;

#修改存储过程 (有几个if 就有几个end if)

DELIMITER $$

CREATE PROCEDURE info\_delete(IN id INT,OUT str CHAR(10))

BEGIN

IF id=1 THEN

UPDATE sdf s SET s.title='sdfdf' WHERE s.id=id;

SET str='update';

ELSE IF id=2 THEN

SET str='undo';

END IF;

END IF;

END $$

DELIMITER ;

## #执行

CALL info\_delete(2,@str);

SELECT @str;

## #创建存储过程，查询数据

DELIMITER$$

#DROP PROCEDURE IF EXISTS select\_info;

CREATE PROCEDURE select\_info(IN id INT)

BEGIN

SELECT \* FROM sdf s WHERE s.id=id;

END $$

DELIMITER;

CALL select\_info(6);

# #存储函数

（和存储过程相差不大）

#create function \show function status;

## #创建一个存储函数

返回title

DELIMITER //

CREATE FUNCTION NameByT(id INT)

RETURNS CHAR(50)

RETURN (SELECT title FROM sdf s WHERE s.id=id);

//

DELIMITER ;

## #查询存储函数

SELECT NameByT(1);

## #删除存储函数

DROP FUNCTION IF EXISTS NameByT;

# #触发器

## #创建一个触发器 create trigger \show trigger

#创建一个触发器，当删除sdf表中的一条记录时，同时将actions表的有关该条记录的数据全部删除

#对于insert语句：只有new.column是合法的

#对于delete语句：只有old.column是合法的

#对于update语句：可以与new或old同时使用

#当触发器设计对触发表自身的更新操作时，只能使用before，after触发器不被允许(update中)

DELIMITER $$

CREATE TRIGGER sdf\_delete\_actions AFTER DELETE

ON sdf FOR EACH ROW

BEGIN

DELETE FROM actions WHERE id=old.id;

END $$

DELIMITER ;

## #删除触发器

DROP TRIGGER sdf\_delete\_actions;

## #触发器中可以执行存储过程

DELIMITER $$

CREATE TRIGGER sdf\_delete\_ AFTER DELETE

ON sdf FOR EACH ROW

BEGIN

CALL select\_info(1);

END $$

DELIMITER ;

DROP TRIGGER sdf\_delete\_;

# #事件

#主要作用：

#关闭账户，打开或关闭数据库指示器，使数据库中的数据在某个间隔后刷新，执行对进入数据库的复杂的检查工作

## #开启：

SET GLOBAL event\_scheduler="ON"

## #关闭：

SET GLOBAL event\_scheduler="OFF"

## #创建事件：

#创建一个立即启动的事件

CREATE EVENT direct\_now ON SCHEDULE AT NOW()

DO INSERT INTO sdf(id,title) VALUE('10','sdfs');

#创建一个每隔10秒插入

CREATE EVENT IF NOT EXISTS t1 ON SCHEDULE EVERY 10 SECOND

ON COMPLETION PRESERVE

DO INSERT INTO sdf(id) VALUES('AAA');

#创建一个事件，每月启动一次，开始于下个月，结束于2017年12月

DELIMITER $$

CREATE EVENT start\_month ON SCHEDULE EVERY 1 MONTH

STARTS CURDATE()+INTERVAL 1 MONTH

ENDS '2017-12-31'

DO

BEGIN

IF YEAR(CURDATE())<2017 THEN

INSERT INTO sdf(id) VALUES(1);

END IF;

END $$

DELIMITER ;

#删除事件

DROP EVENT start\_month;

## #备份与恢复

#使用客户端和程序备份恢复数据

#使用mysqldump备份数据(可在mysql的bin下看到备份文件)

mysqldump -h localhost -u root -p 123456 数据库名 表名 > test.sql

#备份多个数据库

mysqldump -u root -p 123456 --database test ssh >d:/FILE/data.sql

#将test数据库中所有表的结构和数据都备份到D盘file文件夹下

mysqldump -u root -p 123456 --tab=D:/FILE/ test

#恢复数据库

#备份：

mysqldump -u root -p 123455 test >test.sql

#恢复

mysql -u root -p 123456 test<test.sql

#使用sql语句备份和恢复表数据（不包含表结构）

#导出

SELECT \* FROM sdf INTO OUTFILE 'E:/MYFILE.txt'

# #锁定与解锁

## #在sdf表上设置一个只读锁定

LOCK TABLES sdf READ;

## #在sdf表上设置一个写锁定

LOCK TABLES sdf WRITE;

## #解锁

UNLOCK TABLES;

# 参考文献

更多API讲解，请看：

<http://www.cnblogs.com/lyhabc/p/3842015.html>