## 一 终端操作

cmd中输入mysql –uroot –p，然后输入密码

退出：quit或者exit

一个语句结束，需要用分号‘;’隔开

远程连接mysql

mysql –h ip地址 –uroot –p

## 二 常用类型

### 2.1整数类型：

|  |  |
| --- | --- |
| Mysql整数类型 | 含义 |
| Tinyint | 1个字节，+-2^7 ，一百多 |
| Smallint | 2个字节，+-2^15 ，三万多 |
| Mediumint | 3个字节，+-2^23 ，八百万多 |
| Int | 4个字节，+-2^31 ， 二十一亿万多 |
| Bigint | 8个字节，+-2^63 ，…. |

### 2.2浮点数

|  |  |
| --- | --- |
| Mysql浮点类型 | 含义 |
| Float(m,d) | 单精度浮点数，8位精度，4个字节 |
| Double (m,d) | 双精度浮点数，16位精度，8个字节 |

总位数： m<= 255

小数位数： d<=30

|  |
| --- |
| create table test\_t(  f float(255,30),  d double(255,30)  )  Insert into test\_t (f,d) value(123456.7891012345678901234567890,  123456.7891012345678901234567890) |

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
|  | Float(255,30) | Double(255,30) |
| 插入的值 | 123456.7891012345678901234567890 | 123456.7891012345678901234567890 |
| Mysql中的值 | 123456.789062500000000000000000000000 | 123456.789101234560000000000000000000 |

|  |
| --- |
| 结论：  float用四舍五入的方法会保留的8位数  double四舍五入保留16位数  未四舍五入保留的部分就不一定是准确的了 |

### 2.3定点数

decimal(m,d)

总位数：m<=65

小数位数：d<=30

占用m+2个字节

|  |
| --- |
| Create table test\_t2(  d decimal(65,30)  )  Insert into test\_t2 value (1234567890.123456789012345678901234567890) |

|  |
| --- |
|  |

|  |
| --- |
| 结论  定点数不会损失精度，存什么数就是什么数 |

### 2.4字符串

|  |  |  |  |
| --- | --- | --- | --- |
| 命令 | 最大数量 | 占用空间 | 能否设置默认值 |
| Char(n) | 固定长度，最多(2^8)-1个字符 | n个字符长度的字节 | 能 |
| Varchar(n) | 可变长度 | L+1个字节 | 能 |
| Tinytext | 可变长度，最多(2^8)-1个字符 | L+1个字节 | 不能 |
| Text | 可变长度，最多(2^16)-1个字符 | L+2个字节 | 不能 |
| Mediumtext | 可变长度，最多(2^24)-1个字符 | L+3个字节 | 不能 |
| Longtext | 可变长度，最多(2^32)-1个字符 | L+4个字节 | 不能 |

注：L=储存字符占用的字节数（utf-8中，数字和英文占用一个字节，汉字占用3-6个字节）

char(n) 插入的字符数小于n，会用空格填充剩余的部分

所有的字符串类型，插入的字符串最大长度溢出存储长度，不会报错，而是优先保留前面的字符串

例如：

|  |
| --- |
| Create table test\_t3(  V varchar(5)  )  Insert into test\_t3 (v) value (‘123456’) |

|  |
| --- |
|  |

|  |
| --- |
| 说明： |
| 1. 网上好多资料说char(n)占用n个字节，是不对的，n指定的是字符的最大长度，例如：   一个字段指定char(5)  插入’一二三四五’ ：占用的是这5个汉字的字节数，而不是5个字节  插入’12345’ ：占用5个字节   1. 很多资料说varchar(n)中n的最大取值是65535，但是测试发现，可以插入比这个更大的数 |

### 2.5日期

|  |  |  |
| --- | --- | --- |
| Mysql日期类型 | 日期格式 | 占用空间 |
| Datetime | YYYY-MM-DD HH:MM:SS | 8字节 |
| Timestamp | YYYY-MM-DD HH:MM:SS | 4字节 |
| Date | YYYY-MM-DD | 4字节 |
| Time | HH:MM:SS | 3字节 |
| Year | YYYY | 1字节 |

注：

timestamp是时间戳，如果表格中有timestamp字段，插入行的时候，它会自动记录当前操作的时间

### 2.6布尔

boolean类型在mysql中会被转成tinyint(1)，以1或0的方式存储，存错只需要一个字节

## 三 DDL（data definition language）数据库定义语言

### 3.1常见关键词

|  |  |
| --- | --- |
| 主要动词 | 解释 |
| Create | 创建 |
| Drop | 删除 |
| Alter | 修改 |
| Truncate | 截断 |
| Rename | 重命名 |

查看版本 select version

查看当前时间：select now

### 3.2常用的命令

|  |  |
| --- | --- |
| 操作 | 命令 |
| 创建数据库 | create database 数据库名 charset=’utf8’  注意这里utf8没有“-” |
| 使用数据库 | use 数据库名 |
| 查看当前使用的数据库 | select database() |
| 查看当前数据库编码 | show variables like ‘character\_set\_database’ |
| 查看所有数据库 | show databases; |
| 修改数据库的编码 | alter database 数据库名 charset=gbk |
| 删除数据库 | drop database 数据库名 |
| 查看表结构 | desc 表名 |
| 查看当前数据库的所有表 | show tables; |
| 查看所有字段 | show columns from 表名 |

## 四 字段约束与创建表结构

### 4.1常见的字段约束

|  |  |
| --- | --- |
| 约束类型 | 关键字 |
| 主键 | Primary key |
| 外键 | Foreign key references |
| 唯一 | Unique |
| 非空 | Not null |
| 自增 | Auto\_increment |
| 默认值 | Default |
| 注解 | Comment |

### 4.2创建表结构

|  |
| --- |
| create table 【表名】(  【字段名字】 【字段类型】 【约束】,  【字段名字】 【字段类型】 【约束】  ); |

例如：

|  |
| --- |
| create table t\_user (  id int primary key auto\_increment comment ‘主键id’,  username varchar(16) unique not null comment ‘用户名’ ,  password varchar(16) not null comment ‘密码’,  gender tinyint(1) default 1 comment ‘性别1男0女’,  datetime datetime  )  create table t\_user\_comment(  id int primary key auto\_increment,  uid int not null,  foreign key(uid) references t\_user(id)  ) |

## 五 DML（data manipulation language）数据的增删改

### 5.1增加insert

|  |
| --- |
| 插入一行、根据字段索引对应 |
| insert into 表名 value (【第一个值】,【第二个值】) |
| 插入多行、根据字段索引对应 |
| insert into 表名 values (【第一个值】,【第二个值】), (【第一个值】,【第二个值】) |
| 根据字段名称，插入数据 |
| insert into 表名 (字段名字1，字段名字2) value (……)  insert into 表名 (字段名字1，字段名字2) values (……),(……..) |

注：values后面也可以只接一行数据

|  |
| --- |
| 不指定要添加的字段，就按照索引一一对应  Insert into t\_user value(1,’name1’,’pass1’,1,’2018-11-10 11:15:00’)  指定字段  Insert into t\_user (username,password,gender,datetime)  value (‘name2’,’pass2’,1,’2018-11-10 11:16:10’)  增加多个value  Insert into t\_user(username,password,datetime)  Values(‘name3’,’pass3’,’2018-11-10 11:17:10’),  (‘name4’,’pass4’,’2018-11-10 11:17:10’) |

### 5.2删除delete、drop、turncate

|  |  |
| --- | --- |
| 关键字 | 解释 |
| Delete | 用于删除表内行 |
| Drop | 用于删除整个表，可以恢复 |
| Turncate | 用于删除整个表的内容，速度快，不可恢复 |

|  |
| --- |
| Delete from 表名  Delete from t\_user where id = 1  Drop 表名  Drop table t\_user  Turncate 表名  Turncate t\_user |

### 5.3更改update、alter、rename

|  |  |
| --- | --- |
| 关键字 | 解释 |
| Update | 更改表内数据 |
| Alter table | 更改表结构 |
| Rename table | 更改表名字 |

|  |
| --- |
| Update 表 set 字段名字1=值 , 字段名字2 = 值  Update t\_user set username = ‘name\_new’ , password = ‘pass\_new’  Alter table 表名 add|change|drop 列名 类型  Alter table t\_user add timestamp timestamp  Alter table t\_user change username username2 varchar(16)  Rename table 原表名 to 新表名  Rename table t\_user to t\_user\_new |

## 六 DQL（data query language）查询

### 6.1外部操作

|  |  |
| --- | --- |
| 操作 | 命令 |
| 查看当前数据库所有的表 | Show tables |
| 查看表结构 | Desc 表名 |
| 查看表的创建语句 | Show create table 【表名】 |

### 6.2运算符

|  |  |
| --- | --- |
| 比较运算符 | 说明 |
| = | 等于 |
| > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |
| != <> | 不等于 |

|  |  |
| --- | --- |
| 逻辑运算符 | 说明 |
| And | 与 |
| Or | 或 |
| Not | 非 |

|  |  |
| --- | --- |
| 空判断 | Is null |

### 6.3 select

|  |
| --- |
| 查找表中的所有数据 |
| select \* from 【表名】 |
| 指定字段查找 |
| select 字段名字1,字段名字2 from 【表名】 |

### 6.4分组group by

|  |
| --- |
| 格式：  group by 字段1，字段2，字段3… |

|  |
| --- |
| Select \* from t\_user group by username |

如果有多个username，则在上面的语句中只会显示第一个，可以用count(\*)

### 6.5 where和having

|  |
| --- |
| Select \* from t\_user where id >1 and id < 5  Select \* from t\_user group by datetime having datetime > ‘2018-11-10 11:15:00’ |

对比where和having的区别:

where是对from后面指定的表进行筛选，属于对原始数据的筛选

having是对group by的结果进行筛选

### 6.6排序order by

|  |  |
| --- | --- |
| 命令 | 说明 |
| Order by … asc | 升序排列 |
| Order by … desc | 降序排列 |

|  |
| --- |
| 格式：  Order by 字段 1 asc|desc ，字段2 adc|desc… |

|  |
| --- |
| Select id,username from t\_user order by datetime |

### 6.7分页limit

|  |
| --- |
| 格式：  Limit 起始索引，数据条数 |

### 6.8总结分组、条件、排序、分页的先后顺序

|  |
| --- |
| Select… from表  Where 条件  Group by 字段 1，字段2  Having 条件  Order by 字段 asc|desc  Limit by 起始索引，数据条数 |

### 6.9模糊查询like

|  |  |
| --- | --- |
| 关键字 | 解释 |
| % | 任意多的字符 |
| \_ | 一个字符 |

|  |
| --- |
| Select \* from t\_user where username like ‘n\_m%’ |

### 6.10范围查询in

|  |
| --- |
| 格式：  In (数据1，数据2，数据3) |

|  |
| --- |
| Select \* from t\_user where username in (‘name1’,name2’’) |

### 6.11常用函数

|  |  |
| --- | --- |
| 聚合函数 | 解释 |
| Count(…) | 总数 |
| Max(…) | 最大值 |
| Min(…) | 最小值 |
| Sum(…) | 求和 |
| Avg(…) | 平均值 |

|  |  |
| --- | --- |
| 字符和字符串操作 | 函数 |
| 字符的ascii码，空字符串返回0 | Ascii(字符) |
| 查看ascii对应的字符 | Char(数字) |
| 拼接字符串 | Concat(str1,str2) |
| 得到字符串的长度 | Length(str) |
| 截取字符串左面len个字符 | Left(str,len) |
| 截取字符串右面len个字符 | Right(str,len) |
| 截取字符串pos位置后的len个字符 | Substring(str,pos,len) |
| 去掉空格 | Trim(str) |
| 替换字符串 | Replace(str,from\_str,to\_str) |
| 转小写 | Lower(str) |
| 转大写 | Upper(str) |

|  |  |
| --- | --- |
| 计算操作 | 函数 |
| 绝对值 | Abs(int) |
| 计算m/n的余数 | Mod(m/n) m%n |
| 不大于n的最大整数 | Floor(n) |
| 不小于n的最大整数 | Ceiling(n) |
| 四舍五入（d默认是0，表示小数位） | Round(n,d) |
| 求m的n次幂 | Pow(m,n) |
| 圆周率 | PI() |
| 随机生成0--1.0的浮点数 | Rand() |
| 当前日期 | Current\_date() |
| 当前时间 | Current\_time() |
| 当前日期时间 | Current\_now() |

## 七 事务

### 7.1基本命令

|  |  |
| --- | --- |
| 操作 | 命令 |
| 开启事务 | begin |
| 提交 | commit |
| 回滚 | rollback |

### 7.2事务的特性

原子性：指事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。

一致性：事务必须使数据库从一个一致性状态变换到另外一个一致性状态。转账前和转账后的总金额不变。

隔离性：事务的隔离性是多个用户并发访问数据库时，数据库为每一个用户开启的事务，不能被其他事务的操作数据所干扰，多个并发事务之间要相互隔离。

持久性：指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来即使数据库发生故障也不应该对其有任何影响。

### 7.3 数据库的隔离级别

赃读：指一个事务读取了另一个事务未提交的数据。

不可重复读：在一个事务内读取表中的某一行数据，多次读取结果不同。一个事务读到了另一个事务提交后的数据。（update）

虚读（幻读）：是指在一个事务内读取到了别的事务插入的数据，导致前后读取不一致。 (insert)

数据库通过设置事务的隔离级别防止以上情况的发生：

1、READ UNCOMMITTED: 赃读、不可重复读、虚读都有可能发生。

2、READ COMMITTED: 避免赃读。不可重复读、虚读都有可能发生。（oracle默认的）

4、REPEATABLE READ:避免赃读、不可重复读。虚读有可能发生。（mysql默认）

8、SERIALIZABLE: 避免赃读、不可重复读、虚读。

级别越高，性能越低，数据越安全

注：单纯的update和insert不是事务，会发生类似’不可重复读’的可能

### 7.4 操作命令：

|  |  |
| --- | --- |
| 操作 | 命令 |
| 查看数据库安全级别 | SELECT @@tx\_isolation |
| 设置数据库安全级别 | set session transaction isolation level 设置事务隔离级别 |

## 八 外键和级联操作

### 8.1外键

|  |  |
| --- | --- |
| 关键字 | 解释 |
| Foreign key | 外键 |
| Constraint | 约束 |
| References | 标记 |

|  |
| --- |
| 格式：  创建表2引用表1的外键  Foreign key(表2的一个键) references 表1名字(表1被引用的外键)  修改表的外键  Alter table 表1 add constraint foreign key (表1的键) references  表2(表2倍引用的  外键) |

|  |
| --- |
| Create table t\_user(  Id int primary key auto\_increment,  Username varchar(16) not null unique,  Password varchar(16) not null,  datetime datetime,  timestamp timestamp  )  Create table t\_user\_comment(  Id int primary key auto\_increment,  Uid int,  Foreign key(uid) references t\_user(id)  ) |

### 8.2级联操作

|  |  |
| --- | --- |
| 级联类型 | 解释 |
| on delete restrict | 默认值，抛异常 |
| on delete cascade | 如果主表被引用的外键删除，相关联的表的记录也会被删除 |
| on delete set null | 如果主表被引用的外键删除，相关联的表的外键设置为空 |
| on delete no action | 什么也不做 |

|  |
| --- |
| 格式：  On delete restrict|cascade|set null| no action |

|  |
| --- |
| CREATE TABLE forum(  id INT PRIMARY KEY AUTO\_INCREMENT,  topic VARCHAR(100)  );  CREATE TABLE content\_t(  id INT PRIMARY KEY AUTO\_INCREMENT,  content VARCHAR(200),  fid INT,  CONSTRAINT fk1 FOREIGN KEY(fid) REFERENCES forum(id) ON DELETE RESTRICT  );  # 删除外键  ALTER TABLE content\_t DROP FOREIGN KEY fk1;  # 添加外键  ALTER TABLE content\_t ADD CONSTRAINT fk1 FOREIGN KEY(fid) REFERENCES forum(id) ON DELETE CASCADE; |

## 九 逻辑删除

isdelete（逻辑删除字段名称任意）,不真正删除表中的数据

select \* from 表名 where isdelete != 0

## 十 表关系

### 10.1 一对一

|  |
| --- |
|  |

|  |
| --- |
|  |

### 10.2 一对多

|  |
| --- |
|  |

|  |
| --- |
|  |

### 10.3 多对多

|  |
| --- |
|  |

|  |
| --- |
|  |

## 十一 连接查询

### 11.1 内外连接

#### 11.1.1创建表结构

|  |
| --- |
| CREATE TABLE t\_forum (  id INT PRIMARY KEY AUTO\_INCREMENT COMMENT '主键',  topic VARCHAR(100) COMMENT '帖子名称',  username VARCHAR(16) COMMENT '发帖人',  createDatetime DATETIME COMMENT '发帖时间',  isdelete TINYINT DEFAULT 0 COMMENT '逻辑删除，0：未删除，1：删除');  CREATE TABLE t\_content(  id INT PRIMARY KEY AUTO\_INCREMENT COMMENT '主键',  content VARCHAR (200) COMMENT '回复的内容',  username VARCHAR(16) COMMENT '回复的人',  createDatetime DATETIME COMMENT '回复的时间',  fid INT COMMENT '外键约束，连接forum的id',  isdelete TINYINT DEFAULT 0 COMMENT '逻辑删除',  CONSTRAINT fid\_ FOREIGN KEY(fid) REFERENCES t\_forum(id) ON DELETE CASCADE  );  INSERT INTO t\_forum (topic,username,createDatetime) VALUES  ('python好学吗','小深','2019-01-09 00:00:00'),  ('python好吗','学委','2019-01-09 10:10:00'),  ('没人回复的帖子','xxx','2019-01-09 10:10:00');  INSERT INTO t\_content (content,username,createDatetime,fid) VALUES  ('好学','班长','2019-01-09 01:00:00',1),  ('相当好学','生委','2019-01-09 02:00:00',1),  ('非常好学','组长','2019-01-09 03:00:00',1),  ('好','学委','2019-01-09 11:00:00',2),  ('很好','小深','2019-01-09 12:00:00',2); |

|  |
| --- |
|  |

#### 11.1.2内连接

|  |
| --- |
| 关键字：（inner可以省略不写） |
| inner join …… on …… |

|  |
| --- |
| SELECT \* FROM t\_forum f INNER JOIN t\_content c ON f.id = c.fid; |

|  |
| --- |
|  |

#### 11.1.3外链接

|  |
| --- |
| 关键字：（outer可以不写） |
| left/right outer join …… on …… |

|  |
| --- |
| SELECT \* FROM t\_forum f LEFT OUTER JOIN t\_content c ON f.id = c.fid;  SELECT \* FROM t\_forum f RIGHT OUTER JOIN t\_content c ON f.id = c.fid; |

|  |
| --- |
|  |

#### 11.1.4 内连接和外链接的区别

1.内连接只显示满足on条件的内容

2.外链接根据是左外链接还是右外连接，先把这一侧的表内容显示出来，另一测不满足on条件的，字段用null填充

### 11.2 自然连接

#### 11.2.1关键字

|  |
| --- |
| 关键字：（outer可以省略，inner必须省略不写） |
| natural join 表  natural left outer join 表  natural right outer join 表 |

#### 11.2.2创建表结构

|  |
| --- |
| CREATE TABLE user\_1(  id INT PRIMARY KEY AUTO\_INCREMENT,  NAME VARCHAR(10));  CREATE TABLE user\_2(  id INT PRIMARY KEY AUTO\_INCREMENT,  age INT);  INSERT INTO user\_1 (NAME) VALUES ('name1'),('name2'),('name3');  INSERT INTO user\_2 (age) VALUES (10),(15); |

|  |
| --- |
|  |

#### 11.2.3内连接的结果

SELECT \* FROM user\_1 INNER JOIN user\_2;

|  |
| --- |
|  |

#### 11.2.4自然内连接

SELECT \* FROM user\_1 NATURAL JOIN user\_2;

|  |
| --- |
|  |

#### 11.2.5自然外连接

SELECT \* FROM user\_1 NATURAL LEFT OUTER JOIN user\_2;

SELECT \* FROM user\_1 NATURAL RIGHT OUTER JOIN user\_2;

|  |
| --- |
|  |

#### 11.2.6 说明

natural关键字配置内外连接，如果两个表中有同名字段，则只会显示同名字段中存在相同值的行

### 11.3 子查询

在结果中查结果：例如，在t\_content表中查找未被逻辑删除的数据的创建时间大于所有被逻辑删除了的数据中的创建时间的username

更新两个被逻辑删除的数据

|  |
| --- |
| UPDATE t\_content SET isdelete = 1 WHERE id = 2 OR id = 4; |

#### 11.3.1老版本查询

|  |
| --- |
| 1. 查询isdelete=1的数据   SELECT createDatetime FROM t\_content WHERE isdelete = 1;   1. 查询isdelete = 0的数据中的createDatetime > 上次查询的createDatetime的最大值   SELECT username FROM t\_content WHERE createDatetime > '2019-01-09 11:00:00' AND isdelete = 0; |

#### 11.3.2子查询的关键字

|  |  |
| --- | --- |
| 关键字 | 解释 |
| all | 所有 |
| any | 任意 |

#### 11.3.3子查询语句

all

|  |
| --- |
| SELECT \* FROM t\_content WHERE createDatetime > ALL (SELECT createDatetime FROM t\_content WHERE isdelete = 1) AND isdelete = 0; |

any

|  |
| --- |
| SELECT \* FROM t\_content WHERE createDatetime > ANY (SELECT createDatetime FROM t\_content WHERE isdelete = 1) AND isdelete = 0; |

### 11.4 自连接

自己连接自己，不再演示

### 11.5 视图

某些查询结果经常被使用，就可以把这些结果封装到一个视图中。

|  |
| --- |
| 关键字： |
| create view 视图名字 as 查询语句 🡪创建视图  drop view 视图名字 🡪删除视图 |

例如：

|  |
| --- |
| create view v1 as select \* from t\_content where isdelete = 1; |

## 十二 python操作mysql

### 12.1 安装pymysql

|  |
| --- |
| pip install pymysql |

### 12.2 创建连接connect对象

|  |
| --- |
| import pymysql  conn = pymysql.connect(  host = 'localhost',  port = 3306,  user = 'root',  passwd = '12345',  db = 'database\_mysql',  charset = 'utf8',  ) |

### 12.2 增删改

|  |
| --- |
| # 1.创建光标  cs = conn.cursor()  # 2.向mysql发送sql语句，但不执行  succeed\_num = cs.execute(‘sql语句’) # 返回需要执行的行数  # 3.写入数据库  conn.commit()  # 4.关闭  cs.close()  conn.close() |

### 12.3 事务

|  |
| --- |
| # 1.创建光标  cs = conn.cursor()  # try-except块  try:  # 发送sql语句  cs.execute(‘sql语句’)  cs.execute(‘sql语句’)  # 执行sql语句  conn.commit()  except:  # 回滚  conn.roll\_back()  finally:  cs.close()  conn.close() |

### 12.4 查询

关键字：

|  |  |
| --- | --- |
| 命令 | 说明 |
| fetchall() | 取所有 |
| fetchmany(num) | 取指定数目 |
| fetchone() | 取一个 |

例如：

|  |
| --- |
| # 创建cursor,默认返回元组形式的结果，这里采用字典的形式返回  cs = conn.cursor(cursor = pymysql.cursor.DictCursor)  # fetchall:  cs.execute(‘sql语句’)  ret = cs.fetchall()  # fetchmany:  cs.execute(‘sql语句’)  ret = cs.fetchmany(2)  # fetchone：  cs.execute(‘sql语句’)  ret = cs.fetchone() |

## 十三 终端操作乱码解决

set names gbk