使用终端操作数据库

1.如何查看有什么数据库?

```
show databases;
```

2.如何选择数据库?

```
use databasesName;
```

3.如何查看该数据库中有哪些表?

```
show tables;
```

4.如何查询表中的数据?

```
select * from tableName;
```

5.如何退出数据库服务器?

```
exit;
```

6.如何在数据库服务器中创建自己的数据库?

```
create database databaseName;
```

7.如何创建一个数据表? 创建一个pet表

```
create TABLE pet(
                    name VARCHAR(20),
                    owner VARCHAR(20),
                    specise VARCHAR(20),
                    sex CHAR(1),
                    brith DATAE,
                    death DATE );
```

注意事项:

1:var()与varchar()的区别在于var()是定常的,哪怕存储的字符串没有达到"()"中数字的上限,var()依然会占用空格来填充空间.而varchar()则是不定长的,没有达到"()"中的上限则会自动去掉后面的空格;

2:性别不要用:sex 要用:gender 一个是性 一个是性别;

3:定义最后一个字段的时候不要加",";

4:上面的"VAR","VARCHAR","DATE"可以用小写.不过最好用大写来表示区分关键字,若不然也许写到后面你自己都不知道这个词是数据库中的关键字还是你自己自定义的一些数据,同时一定要用英文的标点符号也必须半角输入

8.如何查看数据表的架构?

```
describe tableName;
 说明:
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
 Field    :    字段的名称
  Type     :     字段的类型,可以有int    var    varchar
```

```
  Key       :     是否是关键字 如可以定义为:  primary key 或者 unique key    ...
Default: :     若是该字段没有主动设置值的时候,该字段的默认值是什么?
```

9.如何插入数据?

```
INSERT INTO pet VALUES('kk','cc','dog','1','1998-8-2',null);
```

```
+------+-------+---------+------+------------+-------+
| name | owner | specise | sex  | brith      | death |
+------+-------+---------+------+------------+-------+
| kk   | cc    | dog     | 1    | 1998-08-02 | NULL  |
+------+-------+---------+------+------------+-------+
```

注意:

NULL:代表的是空,表示该字段还没有数据.千万不要主动填写'NULL',这代表你的字段有一个值叫做'null'.

其实还有一种写法:

```
INSERT INTO pet(name,owner) VALUES ('xx','cc');
代表我只在name和owner字段上面插入的一条,其他皆为NULL/默认值的数据
```

10.mysql 常用数据类型

注意:金钱最好用int/bigint(整数,单位用分,拿出来进行*100换成元),千万不要直接用浮点,会有精度损失.

11.如何删除数据

先插入数据:

```
INSERT INTO pet VALUES('kk1','cc1','dog1','1','1998-1-2',null);
INSERT INTO pet VALUES('kk2','cc2','dog2','2','1998-2-2',null);
INSERT INTO pet VALUES('kk3','cc3','dog3','1','1998-3-2','1998-12-2');
INSERT INTO pet VALUES('kk4','cc4','dog4','2','1998-4-2',null);
```

删除语句:

```
DELETE FROM tablesName WHRER 条件;
```

修改数据:

```
UPDATE tableName SET 字段1=值1,字段2=值2 ... WHERE 条件;
```

总结:1.table的操作 2.表操作的总结

12.mysql建表中的约束

1.主键约束:

它能够**唯一确定**一张表中的一条记录,增加主键约束之后,就可以使得字段不重复而且不为空

```
create table user(
    id int PRIMARY KEY,
    name VARCHAR(20)
);
INSERT INTO user VALUES (1,'张三');

+----+------+
| id | name |
+----+------+
|  1 | 张三 |
```

```
+----+------+

运行DESCRIBE user;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
发现 id是不可以为null 而且 key的值 也变为:PRI(primary)
```

2.复合主键:

```
CREATE TABLE user2(
    id INT,
    name VARCHAR(20),
    password VARCHAR(20),
    PRIMARY key(id,name)
);

运行DESCRIBE user2;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| id       | int(11)     | NO   | PRI | NULL    |       |
| name     | varchar(20) | NO   | PRI | NULL    |       |
| password | varchar(20) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+

INSERT INTO user2 VALUES (1,'老王','123456');
INSERT INTO user2 VALUES (2,'老王','123456');

+----+------+----------+
| id | name | password |
+----+------+----------+
|  1 | 老王 | 123456   |
|  2 | 老王 | 123456   |
+----+------+----------+
```

说明了复合主键只要所有的字段都不是相同的情况下可以允许其中的字段重复:

```
INSERT INTO user2 VALUES (1,'老李','123456');

SELECT * FROM user2;
+----+------+----------+
| id | name | password |
+----+------+----------+
|  1 | 老李 | 123456   |
|  1 | 老王 | 123456   |
|  2 | 老王 | 123456   |
+----+------+----------+
场景:表中有班级号以及学生座位号,我们可以用班级号+学生的座位号可以准确的定位一个学生,如:(1班5号可以准确的确定一个学生)
```

3.自增约束:

```
CREATE TABLE user3(
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(20)
);

运行DESCRIBE user3;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
```

```
| id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| name  | varchar(20) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+

INSERT INTO user3(name) VALUES('张三');
INSERT INTO user3(name) VALUES('李四');
+----+------+
| id | name |
+----+------+
|  1 | 张三 |
|  2 | 李四 |
+----+------+
没有自定义id值 但是自动生成了id
```

4.唯一约束:

```
CREATE TABLE user5(
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(20)
);
运行 DESCRIBE user5;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| name  | varchar(20) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+

新增name为唯一约束:
ALTER TABLE user5 ADD UNIQUE(name);
运行 DESCRIBE user5;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| name  | varchar(20) | YES  | UNI | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
测试:插入数据
INSERT INTO user5(name) VALUES ('cc');
运行 SELECT * FROM user5; 查看结果:
+----+------+
| id | name |
+----+------+
|  1 | cc   |
+----+------+
再次插入INSERT INTO user5(name) VALUES ('cc');
出现:ERROR 1062 (23000): Duplicate entry 'cc' for key 'name'

换个试试 INSERT INTO user5(name) VALUES ('aa');
运行 SELECT * FROM user5; 查看结果:
+----+------+
| id | name |
+----+------+
|  3 | aa   |
|  1 | cc   |
+----+------+
总结一下:
    主键约束(primary key)中包含了唯一约束
场景:业务需求:设计一张用户注册表,用户姓名必须要用手机号来注册,而且手机号和用户名称都不能为空,那么:
CREATE TABLE user_test(
    id INT PRIMARY KEY AUTO_INCREMENT COMMENT'主键id',
    name VARCHAR(20) NOT NULL COMMENT'用户姓名,不能为空',
    phone_number VARCHAR(20) UNIQUE NOT NULL COMMENT'用户手机,不能重复且不能为空'
);
```

```
运行 DESCRIBE user_test;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id           | int(11)     | NO   | PRI | NULL    | auto_increment |
| name         | varchar(20) | NO   |     | NULL    |                |
| phone_number | int(11)     | NO   | UNI | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
这样的话就达到了每一个手机号都只能出现一次,达到了每个手机号只能被注册一次.
用户姓名可以重复,但是手机号码却不能重复,复合正常的逻辑需求
```

5.非空约束:

在上面的蓝字中已经添加了非空约束: NOT NULL;

```
name和phone_number都设置了非空,先只设置name参数不设置phone_number参数试一试
INSERT INTO user_test (name) VALUES ('张三');
会出现Field 'phone_number' doesn't have a default value

两个非空参数一起设置:
INSERT INTO user_test (name,phone_number) VALUES ('张三','12345678901');
+----+------+--------------+
| id | name | phone_number |
+----+------+--------------+
|  1 | 张三 | 12345678901  |
+----+------+--------------+
```

6.默认约束

```
CREATE TABLE user6(
   id int PRIMARY KEY AUTO_INCREMENT COMMENT'主键id',
    name VARCHAR(20) NOT NULL COMMENT'用户姓名不能为空',
    phone_number VARCHAR(20) NOT NULL COMMENT'用户手机号,不能为空',
    status INT DEFAULT 0 COMMENT'用户状态0:启用 1:禁封 默认:0'
);
运行DESCRIBE user6;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id           | int(11)     | NO   | PRI | NULL    | auto_increment |
| name         | varchar(20) | NO   |     | NULL    |                |
| phone_number | varchar(20) | NO   |     | NULL    |                |
| status       | int(11)     | YES  |     | 0       |                |
+--------------+-------------+------+-----+---------+----------------+
插入数据:
INSERT INTO user6(name,phone_number) VALUES ('aa','123');
INSERT INTO user6(name,phone_number) VALUES('bb','1234');
INSERT INTO user6(name,phone_number) VALUES('cc','1263456');

查看数据:SELECT * FROM user6;
+----+------+--------------+--------+
| id | name | phone_number | status |
+----+------+--------------+--------+
|  1 | aa   | 123          |      0 |
|  2 | bb   | 1234         |      0 |
|  3 | cc   | 1263456      |      0 |
+----+------+--------------+--------+
我们没有设置status的值,但是给我们创建了默认值 0.

应用场景:
业务需求:找正常的用户,对这些正常用户进行发放优惠卷或者积分之类的东西,而被禁封的用户我们不让其参加多动.
我们想要封用户只要将status的值从0改为1就行了,当然我们取用户的时候必须要先判断status是否是0.若是1.说明该用户已经被禁封.
先封手机号为'1234'的用户:
```

```
UPDATE user6 SET status = 1 WHERE phone_number= '1234';
SELECT * FROM user6;
+----+------+--------------+--------+
| id | name | phone_number | status |
+----+------+--------------+--------+
|  1 | aa   | 123          |      0 |
|  2 | bb   | 1234         |      1 |
|  3 | cc   | 1263456      |      0 |
+----+------+--------------+--------+
status为1,说明用户已经被封,该用户不可以参加活动

我们取用户的时候加上status的判断,如:
SELECT * FROM user6 WHERE status = 0;
+----+------+--------------+--------+
| id | name | phone_number | status |
+----+------+--------------+--------+
|  1 | aa   | 123          |      0 |
|  3 | cc   | 1263456      |      0 |
+----+------+--------------+--------+
```

7.外键约束

```
CREATE TABLE classes(
    id INT PRIMARY KEY AUTO_INCREMENT COMMENT'班级表id',
    name VARCHAR(20) COMMENT'班级名称'
);
运行DESCRIBE classes;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| name  | varchar(20) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+

CREATE TABLE student(
   id INT PRIMARY KEY AUTO_INCREMENT COMMENT'学生表id',
   name VARCHAR(20) COMMENT'学生姓名',
    class_id int COMMENT'教室id,这张表中的class_id是classes表中id的值',
    FOREIGN KEY (class_id) REFERENCES classes(id)
);
//FOREIGN :外来  REFERENCES:应用,参考
运行DESCRIBE student;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(20) | YES  |     | NULL    |                |
| class_id | int(11)     | YES  | MUL | NULL    |                |
+----------+-------------+------+-----+---------+----------------+

班级插入数据:
INSERT INTO CLASSES (name) VALUES ('一班');
INSERT INTO CLASSES (name) VALUES ('二班');
INSERT INTO CLASSES (name) VALUES ('三班');
INSERT INTO CLASSES (name) VALUES ('四班');
查看数据 SELECT * FROM classes;
+----+------+
| id | name |
+----+------+
|  1 | 一班 |
|  2 | 二班 |
|  3 | 三班 |
```

```
|  4 | 四班 |
+----+------+

学生插入数据:
INSERT INTO student (name,class_id) VALUES ('小赵',1);
INSERT INTO student (name,class_id) VALUES ('小钱',2);
INSERT INTO student (name,class_id) VALUES ('小孙',3);
INSERT INTO student (name,class_id) VALUES ('小李',4);
查看数据 SELECT * FROM student;
+----+------+----------+
| id | name | class_id |
+----+------+----------+
|  1 | 小赵 |        1 |
|  2 | 小钱 |        2 |
|  3 | 小孙 |        3 |
|  4 | 小李 |        4 |
+----+------+----------+
若是像插入班级为5的数据 如:
INSERT INTO student (name,class_id) VALUES ('小周',5);
报错: Cannot add or update a child row

我们删除正在被学生表引用的'四班'试试:
DELETE classes WHERE name = '四班';
出现:Cannot delete or update a parent row:不能删除主表中的行

我们先删除学生表中的 '小李'从而解除班级中'四班'的外键约束,再来删除'四班'(因为小李引用了四班)
DELETE FROM student WHERE name = '小李';
再次删除classes表中的'四班';
DELETE FROM classes WHERE name = '四班';
最后: SELECT * FROM classes;
+----+------+
| id | name |
+----+------+
|  1 | 一班 |
|  2 | 二班 |
|  3 | 三班 |
+----+------+
'四班'被成功删除!

总结:
1.主表中没有的数据,在附表中,是不可以使用的.
2.主表中记录的数据现在正在被附表所引用,那么主表中正在被引用的数据不可以被删除
3.若要想删除,先将附表中的数据删除在删除主表数据
4.对于外键约束大家可以联想 省,市 来进行联想 (市必须要依赖于省,只要省还有一个市在引用,那么就不可以删除省,要不然市就没有省了. 那么我们想删除省,必须要将该省下所有的市全部删除之后,才可以删除这个省)
```

8.如何建表之后添加主键约束

```
CREATE TABLE user4(
    id INT,
    name VARCHAR(20)
);
运行DESCRIBE user4;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+

加入主键约束:
ALTER TABLE user4 add PRIMARY KEY(id);
再次运行DESCRIBE user4;
```

```
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+

删除主键约束:
ALERT TABLE user4 DROP PRIMARY KEY;
运行DESCRIBE user4查看表结构:
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+

使用modify 修改字段.添加约束:
ALTER TABLE user4 MODIFY id INT PRIMARY key;
使用DESCRIBE user4 查看表结构:
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+

给主键设置自增长:
ALTER TABLE user4 MODIFY id INT AUTO_INCREMENT;
运行 DESCRIBE user4 查看表结构:
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| name  | varchar(20) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
```