数据库篇

1.数据发展过程：

- 文件操作

```
"""
小飞----123---18
中飞-456-19
大飞|789|20
"""
```

- 软件开发目录规范

```
"""
流星蝴蝶剑

- config
- core
- lib
- log
- bin
- db
readme.md
"""
```

- 数据库程序

  关系型数据库：

  Mysql、Oracle、SQL server、DB2、Access

  - 数据之间有关系或约束
  - 数据通常以表格的形式储存

name（str）	pwd	age（int）
xiaofei	123	18
xiaoming	258	19
xiaojun	368	20

  非关系型数据库：

  MongoDB、Redis、Memcache

  - 存储数据通常都是key、value的形式

2.Mysql引入

> mysql就是一款基于网络通信的应用程序，底层一定是用的socket

mysql服务端支持自己的客户端操作数据、也支持其他编程语言，**怎么解决语言不通的问题?**

- 让服务端精通所有编程语言
- 统一语言（sql语句）

3.MySQL安装

参考mysql安装文档：https://active.clewm.net/FrcyFA

4.SQL语句初体验

```
-- 启动和连接数据库
net start mysql -- 启动数据库服务(需要管理员权限)
net stop mysql -- 关闭数据库服务(需要管理员权限)
mysql -u root -p -- 连接到数据库

-- 每一条sql语句都是以分号结尾的

-- 库 --> 文件夹
-- 表 --> 文件
-- 记录 --> 文件里面的一行行数据
-- 杠杠是单行注释
/*
多行注释
*/

show databases; --查看所有数据库
\s -- 查看数据库字符编码以及其他信息的
\c -- 结束当前语句，或者用ctrl+c
exit; -- 退出连接
quit; -- 退出连接
help <命令> -- 查看命令的帮助信息
```

- 操作库

```
-- 增（create）
create database db1;
create database db1 charset=utf8;  -- 推荐
-- 删（drop）
drop database db2; -- 删除库
-- 改（alter）
alter database db2 charset=utf8; -- 改数据库的字符编码(当前mysql5.1版本之后只能改数据库的字符编码，不能改数据库名称了)
-- 查（show、describe(desc)）
show databases; -- 查所有数据库
show create database db1; -- 查一个，查看创建这个数据库的sql语句
```

- 操作表

```
select database(); -- 查看当前所在数据库
use db1; -- 切换数据库
-- 增（create）
```

```sql
create table movies(id int,name char); -- 创建表（默认字符编码，就是库的字符编码）
create table movies(id int,name char) charset=utf8; -- 创建表
-- 删（drop）
drop table movies; -- 删除当前库中的表
-- 改（alter）
alter table movies rename to MOVIES; -- 修改表名
alter table movies modify name char(4); -- 修改表字段类型
alter table movies change name NAME char(5); -- 修改表字段和类型
-- 查（show、describe(desc)）
show tables; -- 查看当前库下所有的表
show create table movies; -- 查看创建这个数据表的sql语句
describe movies; -- 查看一张表的结构，简写：desc movies;

-- 所有对表的操作都可以使用绝对路径的方式，这样即便不操作数据库，也可操作对应数据库的表
create table db.movies(id int,name char(4)) charset=utf8; -- 在指定的库中创建表
drop table db.movies; -- 删除指定库中的表
alter table db.movies change name NAME char(16); -- 修改指定库中表字段和类型
show create table db.movies; -- 查看指定库中的数据库表的sql语句
describe db.movies; -- 查看指定库的一张表的结构，简写：desc db.movies;
```

- 操作记录

```sql
-- 增 (insert into <表名> values())
insert into movies values(1,'流浪地球'); -- 插入一条记录
insert into movies values(1,'流浪地球'),(2,'三体'); -- 插入多条记录
-- 删 (delete from <表名> where)
delete from movies where name='三体';
-- 改 (update xx set)
update movies set name='满江红' where id=1;
-- 查 (select <字段> from <表名>)
select * from db1.movies; -- 查一个表的所有数据
select id,name from db1.movies; -- 查询这张表所有数据的id,name字段
select user,host from mysql.user; -- 查看user表里面所有用户的user字段和host字段
```

5.SQl语句分类

类型	描述	关键字
DDL	数据库定义语言，用来定义和管理数据库或者数据表	create,alter,drop
DML	数据操纵语言，用来操作数据	insert,update,delete
DQL	数据库查询语言，用来查询数据库	select
DCL	数据库控制语言，权限控制	grant,revoke,commit,rollback

6.库的操作

> 参考之前的笔记

```
-- 增（create）
create database [if not exists] <库名> [charset=utf8];
-- 删（drop）
drop database [if not exists] <库名>;
-- 改（alter）
alter database <库名> [charset=utf8];
-- 查（show、describe(desc)）
show databases;
show create database <库名>;
```

7.表操作

存储引擎

> 表的类型，就是存储引擎

```
show engines; -- 查看所有的存储引擎

'InnoDB':支持事务、行级锁定和外键
'MRG_MYISAM':相同的MyISAM表的集合
'MEMORY':基于哈希，存储在内存中，适用于临时表
'BLACKHOLE':/dev/null存储引擎（您写入其中的任何内容都会消失）
'MyISAM':MyISAM存储引擎
'CSV': CSV存储引擎
'ARCHIVE':归档存储引擎
'PERFORMANCE_SCHEMA': 性能模式
'FEDERATED':联合MySQL存储引擎

Create table t1(id int,name char)engine=innodb;
Create table t2(id int,name char)engine=myisam;
Create table t3(id int,name char)engine=memory;
Create table t4(id int,name char)engine=blackhole;

insert into t1 values(1,'a');
insert into t2 values(1,'a');
insert into t3 values(1,'a');
insert into t4 values(1,'a');
```

创建表的语法

```
create table <表名>(
    <字段名1><字段类型>[(宽度)] [约束条件],
    <字段名2><字段类型>[(宽度)] [约束条件],
    <字段名3><字段类型>[(宽度)] [约束条件],
    <字段名4><字段类型>[(宽度)] [约束条件1,约束条件2]
)engine=innodb charset=utf8;
-- 宽度指的是字符个数，或者说字符串长度，关闭严格模式之后，即便宽度超了，也可以插进去（截取字符），5.6版本之后默认开启严格模式

-- 约束条件,注意顺序
[unsigned][zerofill][not null]
-- unsigned 无符号
-- zerofill 0填充
-- not null 非空
```

修改表的语法

```
-- 修改存储引擎
alter table <表名> engine=<存储引擎名称>;

-- 修改表名
alter table <表名> rename to <新表名>;

-- 增加字段
alter table <表名> add <字段名> <字段类型>[(宽度)] [约束条件] [first|after <字段名>];

-- 删除字段
alter table <表名> drop <字段名>;

-- 修改字段
alter table <表名> modify <字段名> <新字段类型>[(宽度)] [约束条件]; -- 修改表字段类型
alter table <表名> change <旧字段名> <新字段名> <新字段类型>[(宽度)] [约束条件]; -- 修改表字段类型和名称
```

删除和复制表语法

```
-- 删除表
drop table <表名>;

-- 复制创建表
create table <新表名> select * from <旧表名> [条件];

-- 复制创建表结构
create table <新表名> like <旧表名>;
```

8.数据类型

- 数值
    - int(整形)

-- 整形创建的时候设置的宽度是其实是显示宽度，所以在创建时候无需对其设置宽度（这里指的是数据大小），只有以下几种大小，用unsigned约束条件设置为无符号范围，不设置约束条件时默认为有符号范围

类型	大小	范围（有符号）	范围（无符号）	描述
tinyint	1Bytes	(-128，127)	(0,255)	很小的整数
smallint	2Bytes	(-32768，32767)	(0,65535)	较小的整数
mediumint	3Bytes	(-8388608，8388607)	(0,16777215)	一般的整数
int	4Bytes	(-2147483648，2147483647)	(0,4294967295)	标准的整数
bigint	8Bytes	(-9223372036854775808，9223372036854775807)	(0,18446744073709551615)	极大的整数

- float

  ==乘二取整==

  3.625

  3 -> 11

  0.625 -> 0.625x2 = ==1==.25 0.25x2 = ==0==.5 0.5x2 = ==1== -> 0.101

  3.625 -> 11.101

  > 浮点数的表现形式：==符号.尾数==.基数.==指数==
  >
  > ±m*2^e^
  >
  > 单精度浮点数（32位）：符号（1位）+指数（8位）+尾数（23位）
  >
  > 双精度浮点数（64位）：符号（1位）+指数（11位）+尾数（52位）

  符号：1 -> 负，0 -> 正或0

  指数：

  10000001 -> 129 -> 2

10000000 -> 128 -> 1

01111111 -> 127-> 0

01111110 -> 126 -> -1

01111101 -> 125 -> -2

尾数：把小数点前的二进制数，固定为1的规则

3.625 x 2^-1^ -> 3.625 x 0.5 = 1.8125

11.101 -> 1.10100000000000000000000 尾数：==10100000000000000000000==

1.8125 x 2^1^ = 3.625 指数：==^1^== -> 127+1=128 -> ==10000000==

3.625 -> 11.101的存储方式：0 10000000 10100000000000000000000

1.1101 -> 1 * 2^0^ + 1 * 2^-1^ + 1 * 2^-2^ + 0 * 2^-3^ + 1 * 2^-4^ = 1 + 0.5 + 0.25 + 0 + 0.0625 = 1.8125 * 2^1^ = 3.625

3.1

3 -> 11

0.1 -> 0.1x2 = 0.2 0.2x2 = 0.4 0.4x2 = 0.8 0.8x2 = 1.6 0.6x2 = 1.2

3.1 -> 11.0 0011 0011 0011 0011.....

精度问题：如果小数最后一位不是5的话，不精确，延长循环让精度更高

无法保存极小的小数：最小非零值

类型	大小（m表示所有位数，d表示小数位数）	范围（有符号）	范围（无符号）	描述
float	4 Bytes float(m, d) m最多255，d最多30	(-3.402 823 466 E+38，-1.175 494 351 E-38)，0，(1.175 494 351 E-38，3.402 823 466 351 E+38)	0，(1.175 494 351 E-38，3.402 823 466 E+38)	单精度浮点数值
double	8 Bytes double(m, d) m最多255，d最多30	(-1.797 693 134 862 315 7 E+308，-2.225 073 858 507 201 4 E-308)，0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	双精度浮点数值
decimal	decimal(m,d) m最大65，d最大30	依赖于m和d的值	依赖于m和d的值	小数值

- 字符
  - char（定长字符串）

> char(10) 最多能存10个字符，如果超过10个字符，会直接报错，如果不足10个字符，空格补齐(会补充在数据尾部，但查询的时候会自动把尾部的空格去掉，所以当尾部数据有空格时，不要用char)
>
> 缺点：浪费空间
>
> 优点：存取速度快
>
> 范围：0-255字符

  - varchar（变长字符串）

> varchar(10) 最多能存10个字符，如果超过10个字符，会直接报错，如果不足10个字符，直接存
>
> 优点：节省空间
>
> 缺点：存取速度慢　头hello头fei头heiheihei
>
> 范围：0-65535字节，一行的最大字节数。（注：数据库一行数据的最大字节不可超过65535）
>
> 最大字符数 =（65535-行其它字段总字节数-null标志字节数-长度标志字节数）/ 字符集单字符最大字节数。（注：utf8字符集一个字符占三个字节，gbk字符集一个字符占三个字节）

- 时间日期
  - year(年)
  - date(日期：年-月-日)
  - time(时间：时-分-秒)
  - datetime(日期时间：年-月-日 时-分-秒)

> 8字节，1000～9999

  - timestamp

> 4字节，1970～2038

```
create table user(
    id int,
    name varchar(16),
    born year,
    brith date,
    active time,
    reg_time datetime);
insert into user values(1,'fei',now(),now(),now(),now());
insert into user values(2,'jack','2025','2025-10-10','12:00:00','2025-10-10 12:00:00');
```

- 枚举

> enum：单选

```
create table t14(
    id int,
    name varchar(16),
    gender enum('male','female','other')
    );
insert into t14 values(1,'fei','male');
-- 注：只能在给定的范围内插入一条
```

- 集合

> set：多选

```
create table t15(
    id int,
    name varchar(16),
    hobby set('tea','cola','beer')
    );
insert into t15 values(1,'fei','tea');
insert into t15 values(1,'fei','tea,cola');
-- 注：可在给定的范围内插入多条，用逗号隔开
```

9.约束条件

-- 约束条件,注意顺序([unsigned] [zerfill]放前面)
-- unsigned zerofill not null default unique

- unsigned 无符号
- zerfill 0填充
- not null 非空
- default 设置默认值
- unique 唯一

```
create table t16(id int unsigned zerofill not null);-- unsigned、zerofill属于类型约束，not null属于字段约束，所以约束条件需要先写类型约束后写字段约束
| Field | Type                        | Null | Key | Default | Extra |
+-------+-----------------------------+------+-----+---------+-------+
| id    | int(10) unsigned zerofill | NO   |     | NULL    |       |

-- 单列唯一:一个字段单独唯一
create table user(id int unique, name vachar(16) unique);

create table user(id int, name vachar(16),
                  unique(id),
                  unique(name)); -- 可把unique写在字段类型后，或者写为unique(字段名称)
-- 联合唯一：多个字段一起唯一
create table app(id int,
                 host varchar(15),
                 port int,
                 unique(host,port),
                 unique(id));
```

```
insert into app values
(1,'192.168.3.1',3306),
(2,'192.168.3.2',9000),
(3,'192.168.3.2',3306);

insert into app values
(4,'192.168.3.1',3306);
```

- primary key 主键（约束特性：不为空，且唯一）

```
-- 单列主键
create table t19(
    id int primary key,
    name varchar(16));

insert into t19 values(1,'fei'),(2,'jack');
insert into t19 values(1,'rose'); -- 插入相同的id报错
insert into t19(name) values('rose'); -- 插入空id报错

-- 创建表时不指定主键的话，会默认把唯一且不为空的字段指定为组件
create table t20(
    name varchar(16),
    age int unique not null);

-- 复合主键
create table app1(id int,
                  host varchar(15),
                  port int,
                  primary key(host,port),
                  unique(id));

insert into app1 values
(1,'192.168.3.1',3306),
(2,'192.168.3.2',9000),
(3,'192.168.3.2',3306);

insert into app values
(4,'192.168.3.1',3306);
```

- auto_increment：自动递增

```
create table db4;
use db4;
create table t1(id int primary key auto_increment,
                name varchar(16)
               );

insert into t1(name) values('fei'),('jack'),('rose');-- 无需传id，自动递增（会在最后插入的id基础上递增）

-- 清空表(删除表数据)
delete from t1; -- 清空所有数据，不清空自增值
delete from t1 where id=1; -- 删除id为1的数据
truncate t1; -- 清空所有数据，并且清空自增值
```

- foreign key：外键约束

```
create table dep(
    id int primary key,
    name varchar(16),
    `desc` varchar(64)
);

create table emp(
    id int primary key,
    name varchar(16),
    gender enum('male','female'),
    mobile varchar(11),
    dep_id int,
    foreign key(dep_id) references dep(id)
    on delete cascade
    on update cascade
); -- 把emp表的dep_id字段关联到dep表的id字段
-- on delete cascade on update cascade的作用：删除与更新同步，使得存在外键的表不需要去改或者删除外键表的数据，可之间修改本数据表。当本数据表中引用的外键数据没有了时自动删除外键表中的该条数据，当本数据表中的外键数据更新时自动更新外键表

insert into dep values(1,'研发部','造火箭的'),(2,'销售部','卖火箭的'),(3,'人事部','裁员的');
insert into emp values(1,'fei','male','131',1),
    (2,'大仙','male','137',2),
    (3,'tom','male','138',2),
    (4,'jack','male','135',3),
    (5,'rose','female','139',3);

delete from emp where dep_id=1;
delete from dep where id=1;

update dep set name = '市场部' where id=2;
update dep set id = 999 where id=2;

-- 大型项目不建议用外键，小型项目可用
```

10.关系

- 多对一或一对多

```
create table dep(
    id int primary key,
    name varchar(16),
    `desc` varchar(64)
);

create table emp(
    id int primary key,
    name varchar(16),
    gender enum('male','female'),
    mobile varchar(11),
    dep_id int,
    foreign key(dep_id) references dep(id)
    on delete cascade
    on update cascade
);

insert into dep values(1,'研发部','造火箭的'),(2,'销售部','卖火箭的'),(3,'人事部','裁员的');
insert into emp values(1,'fei','male','131',1),
    (2,'大仙','male','137',2),
    (3,'tom','male','138',2),
    (4,'jack','male','135',3),
    (5,'rose','female','139',3);
```

- 多对多

```
create table song(
    id int primary key auto_increment,
    name varchar(16) not null
);
create table singer(
    id int primary key auto_increment,
    name varchar(16) not null
);
create table song2singer(
    id int primary key auto_increment,
    singer_id int not null,
    song_id int not null,
    foreign key(singer_id) references singer(id) on delete cascade on update cascade,
    foreign key(song_id) references song(id) on delete cascade on update cascade
); -- 这张表是用来建立上面两张表的多对多关系的，即将两张表的外键都建立在另外一张表内

insert into song(name) values
('以父之名'),
('夜的第七章'),
('止战之殇'),
('夜曲'),
('北京欢迎你');
```

```
insert into singer(name) values
('周杰伦'),
('刘欢'),
('韩红'),
('成龙');
insert into song2singer(singer_id, song_id) values
(1,1),
(1,2),
(1,3),
(1,4),
(2,5),
(3,5),
(4,5);
```

- 一对一

```
create table customer(
    id int primary key,
    name varchar(16),
    gender enum('male', 'female'),
    mobile varchar(11),
);

create table owner(
    id int primary key,
    room_number varchar(16),
    area int,
    is_loan enum(true,false),
    customer_id int unique,
    foreign key(customer_id) references customer(id) on delete cascade on update
cascade
); -- 保证customer_id的唯一性即可保证数据表的一对一性质


insert into customer values
(1, 'fei', 'male', '131'),
(2, '大仙', 'female', '150'),
(3, 'tom', 'female', '139'),
(4, 'jack', 'male', '135'),
(5, 'rose', 'male', '137');

insert into owner values
(1, '688', 300, false, 2),
(1, '233', 180, true, 4);
```