数据库篇

1、数据库发展过程

- 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

- conf
- core
- lib
- log
- bin
- db
readme.md

"""
```

- 数据库程序

  关系型数据库：

  MySQL、Oracle、SQL server、DB2、Access

- 数据之间有关系或者约束
- 数据通常以表格的形式存储

name(str)	pwd	age(int)
xiaofei	123	18
zhongfei	456	19
dafei	789	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、SQL语句引入

> 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mysql自己的客户端操作数据， 也支持其它编程语言来作为客户端操作数据，**怎么解决语言不通的问题呢?**

- 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

```
-- 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

/*
注释
*/

show databases;  -- 查看所有的数据库
\s  -- 查看数据库字符编码以及其它信息的
\c  -- 结束当前语句
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 database db1;
create database db1 charset=utf8;  -- 推荐
create database if not exists db1 charset=utf8;
-- 删
drop database if exists db2;
-- 改
alter database db2 charset=utf8;
-- 查
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

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

-- 增
create table movies(id int, name char); -- 创建表（默认的字符编码，就是库的字符编码）
create table movies(id int, name char)charset=utf8;  -- 创建表
-- 删
drop table movies; -- 删除movies这张表
-- 改
alter table movies modify name char(4); -- 修改字段类型
alter table movies change name Name char(5); -- 修改字段名字和类型
-- 查
show tables; -- 查看当前库下所有的表
show create table movies; -- 查看创建表的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，这样即便不切换数据库，也可以操作数据库对应的表
create table db2.movies(id int, name char);
```

- 操作记录

```sql
-- 增
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 movies where name='三体';
-- 改
update movies set name='满江红' where id=1;
-- 查
select * from movies; -- 查movies这个表的所有数据
select name from movies;  -- 查询这张表所有数据的name字段
select id, name from movies; -- 查询这张表所有数据的id和name字段
select user, host from mysql.user; -- 查询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	数据库操纵语言，用来操作数据	insert, update, delete
DQL	数据库查询语言，用来查询数据	select
DCL	数据库控制语言，权限控制	grant, revoke, commit, rollback

6、库操作

> 参考前面笔记

语法：

```
-- 增
create database [if not exists] <库名> [charset=utf8];
-- 删
drop database [if exists] <库名>;
-- 改
alter database <库名> charset=utf8;
-- 查
show databases; -- 查所有数据库
show create database <库名>; -- 查一个，查看创建这个数据库的sql语句
```

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

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]
);
-- 宽度指的是字符个数，或者说字符串长度

-- 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=<存储引擎名字>;

-- 修改表名
alter table <表名> rename <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];
alter table <表名> change <旧字段名> <新字段名> <新字段类型>[(宽度)] [约束条件];
```

删除和复制表语法

```
-- 删除表
drop table <表名>;

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

-- 复制表结构
create table <新表名> like <旧表名>;
```

8、数据类型

- 数值
  - int

类型	大小	范围（有符号）	范围（无符号）	描述
tinyint	1 Bytes	(-128，127)	(0，255)	很小的整数
smallint	2 Bytes	(-32768，32767)	(0，65535)	较小的整数
mediumint	3 Bytes	(-8388608，8388607)	(0，16777215)	一般的整数
int	4 Bytes	(-2147483648，2147483647)	(0，4294967295)	标准的整数
bigint	8 Bytes	(-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
>
> $\pm m*2^e$
>
> 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

3.625×2^{-1} -> 3.625 x 0.5 = 1.8125

11.101 -> 1.10100000000000000000000 尾数：10100000000000000000000

$1.8125 \times 2^1 = 3.625$ 指数：1 -> 127+1=128 -> 10000000

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

$1.1101 \rightarrow 1 * 2^0 + 1 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3} + 1 * 2^{-4} = 1 + 0.5 + 0.25 + 0 + 0.0625 = 1.8125 * 2^1 = 3.625$

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	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
    >
    > 缺点：浪费空间
    >
    > 优点：存去速度快
    >
    > 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
>
> 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

- 时间日期
  - year
  - date
  - time
  - datetime

    > 8字节，1000～9999

  - timestamp

    > 4字节，1970～2038

```
use db3;
create table user(
    id int,
    name varchar(16),
    born year,
    birth 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 t10(
    id int,
    name varchar(16),
    gender enum('male', 'female', 'other')
);
insert into t10 values(1, 'fei', 'male');
```

- 集合

> set：多选

```
create table t11(
    id int,
    name varchar(16),
    hobby set('tea', 'cola', 'beer')
);
insert into t11 values(1, 'fei', 'tea');
```

9、约束条件

> 注意顺序unsigned zerofill需要放前面

- unsigned：无符号
- zerofill：0填充
- not null：非空
- default：默认值
- unique：唯一

```
```

```
create table user(id int, name varchar(16) unique);

insert into user values(1, 'fei');

-- 单列唯一
create table user(
    id int unique,
    name varchar(16) unique
);

create table user(
    id int,
    name varchar(16),
    unique(id),
    unique(name)
);

-- 联合唯一
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, '192.168.3.1', 9000),
(3, '192.168.3.2', 3306);

insert into app values
(4, '192.168.3.2', 3306);
```

- primary key：主键（约束特性：不为空，且唯一）

```
-- 单列主键
create table t10(
```

```
    id int primary key,
    name varchar(16)
);

create table t11(
    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, '192.168.3.1', 9000),
(3, '192.168.3.2', 3306);
```

- auto_increment：自动递增

```
create database db4;
use db4;
create table t1(
    id int primary key auto_increment,
    name varchar(16)
);
insert into t1(name) values
('fei'),
('jack'),
('rose');

-- 清空表
delete from t1; -- 只清空数据，不清空自增值
```

```
truncate t1; -- 同时清空数据和自增值
```

- foreign key：外键约束

```
drop table dep;

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);
insert into emp values
(2, '大仙', 'female', '150', 2),
(3, 'tom', 'female', '139', 2),
(4, 'jack', 'male', '135', 3),
(5, 'rose', 'male', '137', 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;
```

10、关系

- 多对一

```
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
);
```

- 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
);


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
);


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);
```

11、记录的查询语法

```
select distinct <字段.....> from <库名>.<表名>
    where <过滤条件>
    group by <分组条件>
    having <过滤条件>
    order by <排序字段> {ASC | DESC} -- 默认升序ASC
    limit n;
```

例子

```sql
create database db5;
use db5;
create table emp(
    id int primary key auto_increment,
    name varchar(16) not null,
    gender enum('male', 'female') not null,
    age int not null,
    salary float(10, 2),
    dep varchar(32),
    notes varchar(64)
);

insert into emp(name, gender, age, salary, dep) values
('关羽', 'male', 20, 8000, '技术部'),
('张飞', 'male', 25, 12000, '技术部'),
('赵云', 'male', 19, 6800, '技术部'),
('马超', 'male', 26, 11000, '技术部'),
('黄忠', 'female', 48, 15000, '技术部'),
('夏侯惇', 'male', 36, 34000, '技术部'),
('典韦', 'male', 19, 6500, '技术部'),
('吕布', 'female', 20, 9000, '技术部'),
('周瑜', 'female',32, 36000, '技术部'),
('文丑', 'male', 27, 24000, '技术部'),

('刘备', 'male', 32, 4000, '市场部'),
('诸葛亮', 'male', 27, 2700, '市场部'),
('庞统', 'male', 37, 4200, '市场部'),
('徐庶', 'male', 36, 4000, '市场部'),
('荀彧', 'male', 25, 2400, '市场部'),
('荀攸', 'male', 25, 2400, '市场部'),
('鲁肃', 'male', 43, 4300, '市场部'),
('司马懿', 'female', 44, 5000, '市场部'),
('杨修', 'male', 19, 800, '市场部'),
('丁仪', 'male', 49, 3500, '市场部'),

('宋江', 'male', 30, 4000, '人事部'),
('吴用', 'male', 38, 3000, '人事部'),
('扈三娘', 'female', 42, 2500, '人事部'),
```

```
('顾大嫂', 'female', 38, 3300, '人事部'),
('孙二娘', 'female', 32, 2400, '人事部'),
('丁得孙', 'male', 32, 2800, '人事部'),


('柴进', 'male', 30, 4200, '财务部'),
('卢俊义', 'male', 44, 4000, '财务部');

-- 四则运算
select name,salary*12 from emp;
select name,salary*12 as yearly_salary from emp;
select name,salary*12 yearly_salary from emp;

-- 设置显示格式
-- 姓名: 关羽  年龄: 20 薪资: 8000
-- 姓名: 张飞  年龄: 25 薪资: 12000
-- concat()
select concat('姓名: ', name, '    年龄: ', age) as '姓名 年龄',
concat('薪资: ', salary) as '薪资' from emp;
select concat(name, '-', age, '-', salary) as '姓名-年龄-薪资' from
emp;
select concat_ws('-', name, age, salary) as '姓名-年龄-薪资' from
emp;
```