# 零 关于数据库

数据库系统解决的问题：持久化存储，优化读写，保证数据的有效性。

当前使用的数据库，主要分为两类

文档型：如sqlite，就是一个文件，通过对文件的复制完成数据库的复制；

服务型：如mysql，数据存储在物理文件中，使用终端以tcp/ip协议进行数据操作。

E-R模型：当前物理的数据库都是按照E-R模型进行设计的

E表示entry，实体

R表示relationship，关系

一个实体转换为数据库中的一个表

关系描述两个实体之间的对应规则，包括

一对一

一对多

多对多

关系转换为数据库表中的一个列；

在关系型数据库中一行就是一个对象。

三范式：设计数据库的一些规范被称为范式，后一个范式，都是在前一个范式的基础上建立的。

第一范式（1NF)：列不可拆分

第二范式（2NF)：唯一标识

第三范式（3NF)：引用主键

一个数据库就是一个完整的业务单元，可以包含多张表，数据被存储在表中

在表中为了更加准确的存储数据，保证数据的正确有效，可以在创建表的时候，为表添加一些强制性的验证，包括数据字段的类型、约束。

# 一 MySQL基本使用

和大多数开源产品一样，mysql分为社区版和商业版，这两个版本又个字分四个版本发布：Alpha，Beta，RC，GA（正式版）。

5.7版本已经可以投入生产，性能更高。

## 1 字段类型

在mysql中包含的数据类型很多，这里主要列出来常用的几种：

数字 int,tinyint(8位)、smallint、mediumint、int、bigint

字符串 varchar,text

日期 datetime

布尔 bit

## 2 约束

主键 primary key 性能最高，且唯一

非空 not null

惟一 unique

默认 default

外键 foreign key

关于主键：  
 一列或者一组列，其值能位移区分表中的每一行。没有主键，更新或者删除表中特定的行很困难，因为没有安全的方法保证只涉及相关的行。虽然并不总是需要主键，但是大多数数据库管理人员都会为每个表创建一个主键，以便于以后的数据管理。

表中的任何列都可以作为主键，只要同时满足：任意两行都不具有相同主键值；每行都有一个主键值，且不为NULL。

使用多列多为主键时，上述条件必须应用到构成主键的所有列，所有列的组合必须唯一。

## 3 数据库操作

创建数据库

create database 数据库名 charset=utf8;

删除数据库

drop database 数据库名;

切换数据库

use 数据库名;

查看所有数据库

show databases;

查看当前选择的数据库

select database();

显示授权用户：

show grants;

## 4 表操作

查看当前数据库中所有表

show tables;

查看当前表中所有列：

show columns from 表名;

describe 表名; //快速写法，也可以简写为：desc 表名;

创建表

create table 表名(列及类型);

create table students(

id int auto\_increment primary key not null,

name varchar(10) not null,

gender bit default 1,

birthday datetime

);

修改表

alter table 表名 add|change|drop 列名 类型;

alter table students add birthday datetime;

删除表

drop table 表名;

更改表名称

rename table 原表名 to 新表名;

查看表的创建语句

show create table '表名';

## 5 数据操作

增加

全列插入：insert into 表名 values(...)

缺省插入：insert into 表名(列1,...) values(值1,...)

同时插入多条数据：insert into 表名 values(...),(...)...;

或insert into 表名(列1,...) values(值1,...),(值1,...)...;

删除

delete from 表名 where 条件

逻辑删除，本质就=是修改update：alter table students add isdelete bit default 0;

如果需要删除则：update students isdelete=1 where ...;

修改

update 表名 set 列1=值1,... where 条件

查询

select \* from 表名

# 二 查询入门

## 1 select

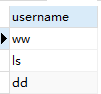
select \* from 表名;

select后面写表中的列名，如果是\*表示在结果中显示表中所有列；

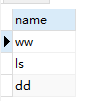
在select后面的列名部分，可以使用as为列起别名，这个别名出现在结果集中；

如果要查询多个列，之间使用逗号分隔。

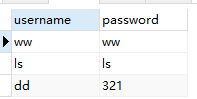
select username from user;



select username as name from user;



select username,password from user;



select distinct gender from 表名;

消除重复行：在select后面列前使用distinct可以消除重复的行

## 2 筛选：where与运算符

select \* from 表名 where 条件;

条件：使用where子句对表中的数据筛选，结果为true的行会出现在结果集中

比较运算符：= > >= < <= !=或<>

select \* from students where id>3;

select \* from students where sname!='黄蓉';

select \* from students where isdelete=0;

逻辑运算符：and or not

select \* from students where id>3 and gender=0;

## 3 模糊查询：like与通配符

使用like进行模糊查询： % 表示任意多个任意字符， \_ 表示一个任意字符

select \* from students where name like '黄%'; 查询姓黄的学生

select \* from students where name like '黄\_'; 查询姓黄且名字是一个字的学生

select \* from students where name like '黄%' or name like '%靖%'; 查询姓黄或叫靖的

注意：不要过度使用通配符，尽量使用别的方式达到相同目的（为了提升性能）；

尽量不要把通配符置于搜索模式的开始处，因为这样搜索最慢；

## 5 范围查询：in与between and

in表示在一个非连续的范围内，between ... and ...表示在一个连续的范围内

select \* from students where id in(1,3,8); 查询编号是1或3或8的学生

select \* from students where id between 3 and 8;查询学生是3至8的学生

select \* from students where id between 3 and 8 and gender=1;查询学生是3至8的男生

## 4 空判断：is null

判空is null，判非空is not null，注意：null与''是不同的

select \* from students where hometown is null; 没有填写地址的学生

select \* from students where hometown is not null; 填写了地址的学生

select \* from students where hometown is not null and gender=0; 填写了地址的女生

## 6 优先级

小括号，not，比较运算符，逻辑运算符

and比or先运算，如果同时出现并希望先算or，需要结合()使用

# 三 查询详解

## 1 排序：order by

将行数据按照列1进行排序，如果某些行列1的值相同时，则按照列2排序...

select \* from 表名 order by 列1 asc,列2desc,...

asc从小到大排列，即升序，不写asc为默认升序

desc从大到小排序，即降序

查询未删除男生学生信息，按学号降序

select \* from students

where gender=1 and isdelete=0

order by id desc;

## 2 五个聚合函数·

聚合的目的是为了统计，但是聚合后只有结果没有以前的数据集合了。

count(\*)表示计算总行数，括号中写星与列名，结果是相同的

select count(\*) from students; 查询学生总数

max(列)表示求此列的最大值

select max(id) from students where gender=0; 查询女生的编号最大值

min(列)表示求此列的最小值

select min(id) from students where isdelete=0; 查询未删除的学生最小编号

sum(列)表示求此列的和

select sum(id) from students where gender=1; 查询男生的编号之和

avg(列)表示求此列的平均值

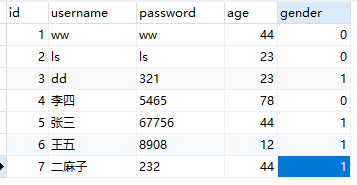
select avg(id) from students where isdelete=0 and gender=0;查询未删女生的编号平均值

## 3 分组：group by

按照字段分组，表示此字段相同的数据会被放到一个组中。分组后，只能查询出相同的数据列，一般会对分组后的数据进行统计，做聚合运算。分组仍然是为了聚合，额外提供了聚合后的集合。

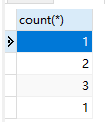
select 列1,列2,聚合... from 表名 group by 列1,列2,列3...

现在有如下一张user表：



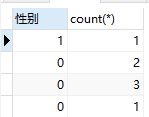
现在我们要按照年龄分组：年龄相同的分为一组

select count(\*) from user group by age;



可以看到统计信息：分成了四组，但没有显示结果集。按照性别查询，新的查询如下：

select gender as 性别,count(\*) from user group by age;查询结果如下：



group by指示mysql进行分组，然后对每个组而不是整个结果集进行聚集。

## 4 分组后筛选：having

select 列1,列2,聚合... from 表名 group by 列1,列2,列3... having 列1,...聚合...

查询男生总人数

方案一

select count(\*)

from students

where gender=1;

-----------------------------------

方案二：

select gender as 性别,count(\*)

from students

group by gender

having gender=1;

对比where与having

where是对from后面指定的表进行数据筛选，属于对原始数据的筛选

having是对group by的结果进行筛选

## 5 获取部分行-分页

从start（第几条数据）开始，获取count（每页展示多少条）条数据，start一般从0开始：

select \* from 表名 limit start,count

示例：SELECT \* FROM users LIMIT 4,2; 从第五行开始取出2行数据。

分页：每页显示m条数据，当前显示第n页

select \* from students

where isdelete=0

limit (n-1)\*m,m

# 四 完整的select语句

select distinct \*

from 表名

where ....

group by ... having ...

order by ...

limit star,count

执行顺序为：

from 表名

where ....

group by ...

select distinct \*

having ...

order by ...

limit star,count

实际使用中，只是语句中某些部分的组合，而不是全部

# 五 SQL与正则

查找username包含 d 的用户：

select username from user where username regexp 'd' order by age;

正则与like的区别：

like匹配整个列，即使要查找的数据在列值中出现，也无法找到；

正则匹配的是列值，只要值内包含要查找的对象就会返回结果集；

正则使用定位符 ^ 与 $ 可以实现与 like 一样的特性。

匹配模式：

’.000’ 特殊符号 . 表示匹配任意字符

‘1000|2000’ 匹配到1000 or 2000

‘[123] ton’ 匹配到1或者2或者3，比如1ton，2ton都会返回结果集，

写为 1|2| ton 也是正确的。

‘[^123] ton’ 匹配除了1或者2或者3之外的数据

‘[a-z]’ 匹配任意字母

‘[1-3]’ 匹配1 2 3

‘\\-’ 匹配带 - 的数据，由于 - 是特殊字符，需要 \\ 转义

mysql中常见预定义的匹配：

[:alnum:] 任意字母和数字，同：[a-zA-Z0-9]

[:alpha:] 任意字符，同：[a-zA-Z]

[:digit:] 任意数字，同：[0-9]

[:lower:] 任意小写字母，同：[a-z]

[:uper:] 任意大写字母

元字符：

\* 0个或多个匹配

+ 1个或多个匹配，同 1,{}

？ 0或1个匹配，同{0,1}

{n} 指定数目的匹配

{n,} 不少于指定数目的匹配

{n,m} 匹配数目的范围，m不能超过255

^ 文本的开始

$ 文本的结尾

[[:<:]] 词的开始

[[:>:]] 词的结束

案例：

‘\\([0-9] sticks?)’ 匹配？前面的s有或者没有

注意：可以在不使用数据库的情况下测试正则，如：

select ‘hello’ regexp ‘[0-9]’;

# 六 数据处理函数

注意：虽然函数的出现让sql的书写变得简便，且有助于提升性能，但是却造成了数据在不同数据库管理软件之间迁移的困难，因为不同的数据库软件支持的函数不尽一致，所以在书写函数时，尽量给出注释。

## 1 拼接函数 concat()

select concat(username,'(', age, ')') from user;



注意：大多数数据库软件使用的是 + 或者 || 来实现拼接，mysql使用的是 concat函数！

## 2 去除空格函数 trim()

select concat(trim(username)) from user;

此外还有，去除左边空格：ltrim()，去除右边空格rtrim()。

## 3 常见函数

upper() 转换为大写

lower() 转换为小写

left() 返回串左边的字符

rigt()

length() 返回串长度

locate() 找出串的第一个子串

subString() 返回串的子串

## 4 日期和时间处理函数

日期和时间采用响应的数据类型和特殊的格式存储，以便能快速的排序、过滤，并且节省物理存储空间。

一般应用程序不使用用来存储日期和时间的格式，因此日期和时间处理函数通常用来读取、统计、处理这些值。

常见日期和时间处理函数：

NOW() 返回当前的日期和时间

CURDATE() 返回当前的日期

CURTIME() 返回当前的时间

DATE() 提取日期或日期/时间表达式的日期部分

EXTRACT() 返回日期/时间按的单独部分

DATE\_ADD() 给日期添加指定的时间间隔

DATE\_SUB() 从日期减去指定的时间间隔

DATEDIFF() 返回两个日期之间的天数

DATE\_FORMAT() 用不同的格式显示日期/时间

注意：mysql推荐日期写为：yyyy-mm-dd

案例：

select cust\_id,order\_num from orders where order\_date=’2018-01-01’;

这种写法看似没有什么问题，也能查出日期为2018-01-01的订单，但是order\_date的存储格式为datatime，这些值全部具有00:00:00。如果下单的时间具有具体的时间，那么上述的数据就无法查出。

解决办法是仅仅将给出的日期与列中的日期部分进行比较，下列为更可靠的sql语句：

select cust\_id,order\_num from orders where Date(order\_date) = ‘2018-01-01’;

即使存储的只是日期，也推荐使用上述格式查询，当然也有相应的事件函数time()，用法一致。

如何检索出某个时间断下的所有订单？

select cust\_id,order\_num from orders where Date(order\_date) between ‘2005-09-01’ and ‘2005-09-30’;

或者：

select cust\_id,order\_num from orders where Year(order\_date) = 2005 and month(order\_date) = 9

## 5 数值处理函数

abs() 返回绝对值

cos() 返回一个角度的余弦，相应的还有正弦sin() 正切tan()

exp() 返回一个数的指数值

mod() 返回除操作的余数

pi() 返回圆周率

rand() 返回一个随机数

sqrt() 返回一个数的平方根