数据库

车库：存车用的
水库：存放水的

数据：数值/结论的意思（通过这几年的自身观察以及统计得出的结果，我变得更加帅气了，）数据分为很多种，最简单的就是数字，那么它还可是文字，图像，声音等
数据库（Database）：是按照数据结构来存储和管理数据的仓库。MySQL，Oracle，MongoDB...

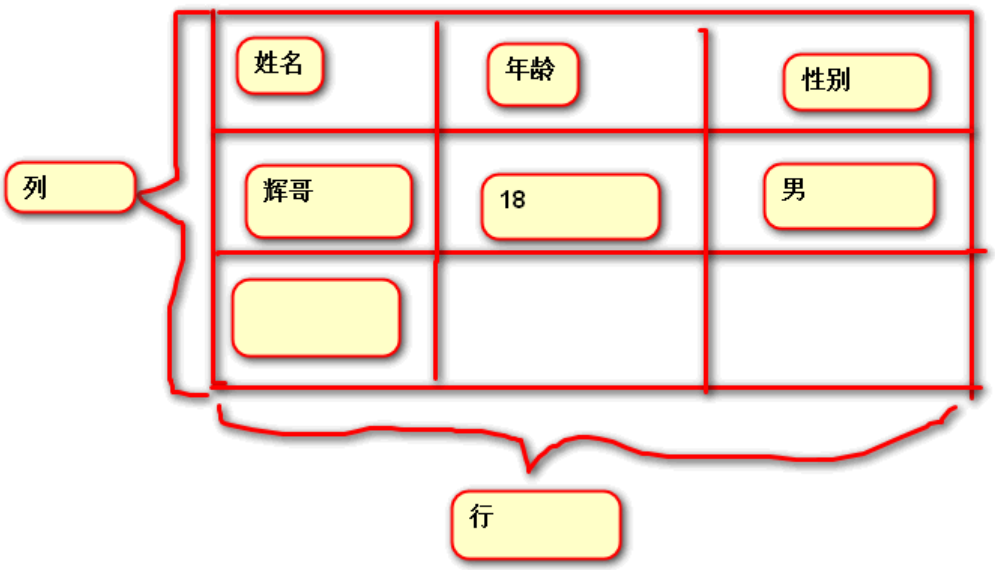
MySQL

1. MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。
2. MySQL将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。
3. MySQL是开放源码软件，可以大大降低总体成本

关系型数据库：是由多张能互相联接的二维行列表格组成的数据库。

数据库的组成

数据表：可以存储大量数据的表格，并且可以对这些数据进行操作（增删改查）。
列：每列(数据元素）包含了相同的数据，例如姓名。
行：每行是一组相关的数据，例如人，姓名，年龄，性别。



数据库的安装

测试安装是否成功及执行

测试mysql版本：cmd黑窗口---输入 mysqladmin --version

进入bin目录，按住shift键然后点击鼠标右键可以选择在该目录下打开命令窗口，或者在地址栏中输入cmd进入命令窗口。输入mysql -u root -p后回车，然后会提示输入密码，输入密码后就会进入MySQL的操作管理界面。

输入show databases；（注意末尾有分号）可以查看当前MySQL中的数据库列表，输入use test；可以进入test数据库（前提是要有此数据库），输入show tables可以查看test数据库中的所有表，输入quit可以退出MySQL的操作管理界面。

环境变量--计算机--系统属性---高级系统程序设计--环境变量--下面那个环境点击Path

- 执行MySQL

  黑窗口输入：cls回车--表示清屏

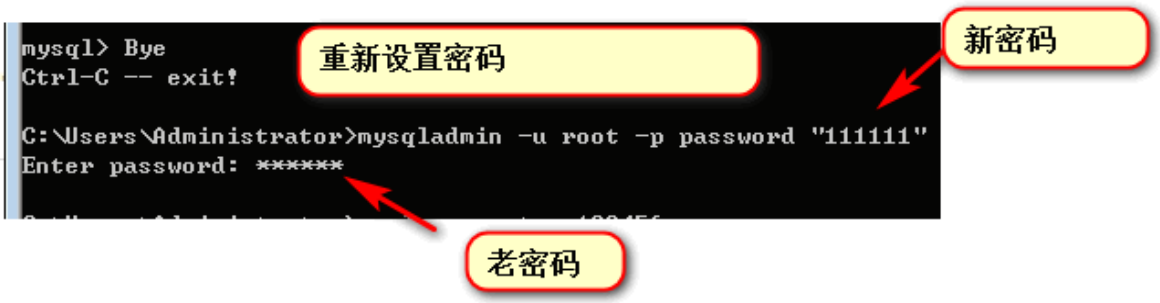
  1. cmd---输入：mysql -u root -p123456

     u:user的缩写　p：password的缩写 密码要挨着p输入

  2. 重新设置密码

     mysqladmin -u root -p password "123456"; #123456=新密码

     #之后需要输入旧密码



基本指令

显示所有数据库：

```
mysql>show databases;
```

选择数据库：

```
mysql>use test;
```

查看数据库中的所有表：

```
mysql>show tables ;
```

查看表中的数据：

```
mysql>select * from 表名;
```

查看当前时间：

```
mysql>select now();
```

防止乱码

```
1. 到Mysql安装目录下，找到my.ini
2. 查找 latinl（iso-8859-1），全部改为utf8
-------------------------------------------------
3. 防止在客户端连接时（黑窗口），中文乱码
    黑窗口是win的黑窗口，中文的黑窗口，他所支持的是gbk（国标码）
        1.查看编码：show variables like 'char%';
        2.由于mysql编码设定为utf8，而win的cmd默认是gbk，所以会显示乱码
        3.解决方法：mysql>set names gbk; --临时修改查询输出格式，仅对当前连接有效
```

数据库操作

```
创建数据库：
    mysql>create database 数据库名（英文） charset=utf8；
删除数据库：
    mysql>drop database 数据库名；
选择/切换数据库：
    mysql>use 数据库名
查看当前选择的数据库：
    mysql>select database()；
```

简单的建表操作

```
1. 建表语法：
        create table 表名英文(列名 数据类型 约束，.....)   约束指的是限制比如姓名这列不能为空等等
2. 数据类型：
        int：整数
        varchar(n):字符串，n表示长度
        datetime：日期
3. 约束：
        auto_increment ：自动增长
        primary key ：主键 -- 比如身份证，根据身份证可以直接查到，便于查询每个表必须有个主键，一般是id
        unique ： 唯一  --只能有一个不能重复
        not null ：非空 --  就是不能是空的
4. 建表：
        create table t_user(
        id  int  auto_increment  primary key,
        name varchar(20) not null unique,
        age  int,
        birth datetime
        )ENGINE=InnoDB DEFAULT CHARSET=utf8;
       desc  t_user; ---查看表的设计
       insert into 表名(name,age,birth) values('zhj',18,'2018-06-11'); ---插入数据
       select * from t_user; ---查询数据
```

- 安装Navicat

```
双击安装就行
```

数据库之增删改查（SQL语句）

结构化查询语言(Structured Query Language)简称SQL，是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统。
它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以具有完全不同底层结构的不同数据库系统，可以使用相同的结构化查询语言作为数据输入与管理的接口

查询

- 查询表中的列

```
语法：select 列名1，列名2等 from 表名
    select * from 表名    --- *表示所有列，但不建议使用，影响效率和可读性
```

- 列的操作

```
1. 列查询支持运算：+，-，*，/
        select id+1,age*2 from 表名
        select id+age,id from 表名
        ---------------------------------------
        注意：在运算中如果出现null，结果也会是null
        解决办法：select id+ifnull(age,1) from 表名
                如果age为null让age=1，不为空还是age
2. 列拼接 : CONCAT(str,str,...)
        SELECT CONCAT(id,18) FROM t_user            #把id和18拼接在一起
        SELECT CONCAT(ID,NAME) FROM t_user          #把id和name拼接在一起
        SELECT CONCAT(18,'ab','cd') FROM t_user     #把18和ab和cd拼接在一起
        注意：数据库中，字符串必须用单引号标识

3. 列去重
        SELECT distinct name from t_user           #选择所有用户名，并消除重复
        SELECT distinct age，id,name from t_user   #逻辑扭曲查询去重后的age，但是还要查询所有
id，name
4. 列别名
        SELECT 列名 as 别名，...  FROM ...
        select id+1 as 'idd',age+2 as 'hehe' from user125;
        SELECT concat(name,'-',age,'-',id) as 全名 FROM t_user;
    注意：as可以省略
```

- where子句：限制查询的数据行

```
1. 比较运算符： =  !=  >  <  <=  >=  <>:不等于
        select id,name,birth from user125 where birth>'2018-06-20'/now()
2. 逻辑运算符： AND  OR
        select id,name,birth from user125 where id>1 and birth<'2018-06-20'/now()
3. 范围：IN(x,x,..) , BETWEEN 起始范围 and 结束范围  ---[开始，结束]
        select id，name from user125 where id in(18,19,20)
        select id，name from user125 where age between 18 and 20
```

- where子句：空值-null判断

```
SELECT .. FROM 表名 WHERE 列名 is null and 列名 is not null
    select id,name from s1 where age is null and name is not null;
```

- where子句：模糊查询（like）

```
select ... from 表名 where 列名 like '%._'
        %:表示任意多个字符
        _：表示一个字符
    like '%abc%' 含有abc
    like "%abc"   abc结尾
    like "abc%"   abc开头
    like “ab%_”    ab开头，ab后至少一个字符
    like '%____%'   4个字符
 select id,name from s1 where name like '%h%';
 select id,name from s1 where name like '__';
```

- 设定查询记录数：分页查询（limit ）

```
select .. from 表名 limit 0,2----从第一条开始查询，共查询2条
select .. from 表名 limit 2,3----从第三条开始，共查询3条
select ... from 表 limit (m-1)*n,n----每页显示n条，查询第m页

select id,name from s1 where id>=0 LIMIT 0,2;
select id,name from s1 where id>=0 LIMIT 2,2;
select id,name from s1 where id>=0 LIMIT 4,2;
```

- order by 子句：排序

```
ASC : 升序(默认)       DESC : 降序
-----------------------------------
1. select ..from 表名 ORDER BY id;---不加where
        select id,name from s1 ORDER BY id;
        select id,name from s1 ORDER BY id desc;
2. SELECT .. FROM 表名   WHERE ...
                                ORDER BY  列名1  DESC
                                   ORDER BY  列名1  ASC
                                   ORDER BY  列名1  DESC,列名2 ASC
```

根据id升序排列 - 升序是默认排序
    SELECT id,name,age FROM t_user WHERE age>18 ORDERBY ID
根据id降序排列
    SELECT id,name,age FROM t_user WHERE age>18 ORDERBY ID DESC
先根据id降序排列，id重复的再根据name升序排列 【联合排序】
    SELECT id,name,age FROM t_user WHERE age>18 ORDERBY ID DESC,NAME ASC

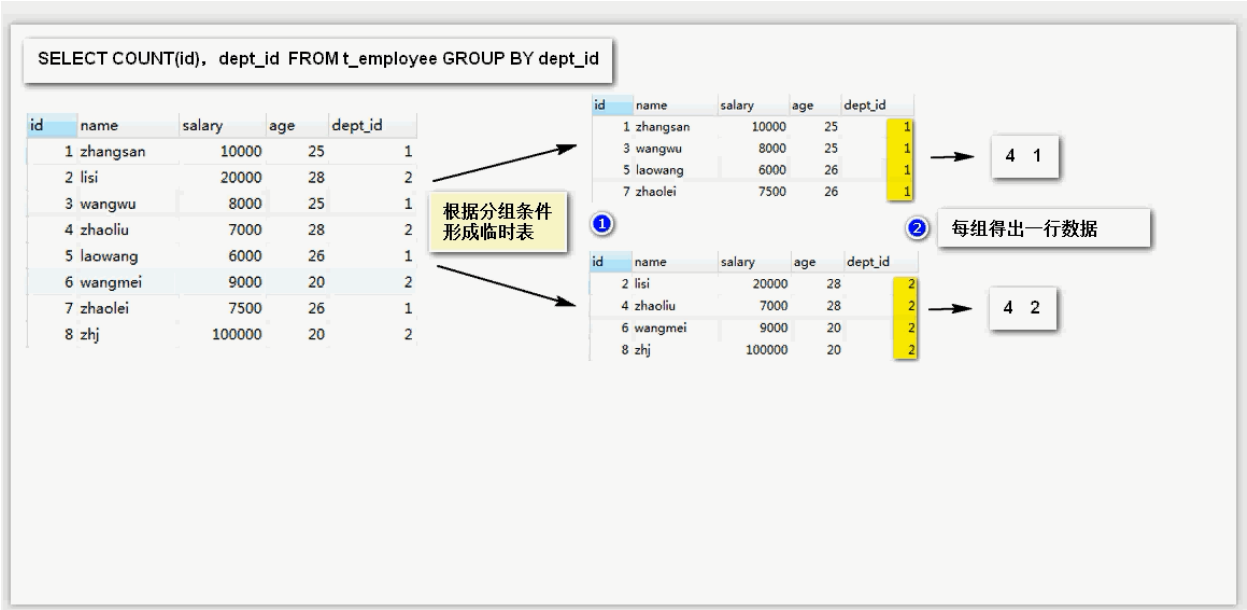
- 聚合操作：组函数

语法：SELECT max(id),avg(age) FROM t_user
1. MAX():最大值
2. MIN()；最小值
3. SUM()：和
4. AVG():平均值
5. COUNT():总个数
    查询最大id和年龄的平均值
        SELECT max(id),avg(age) FROM t_user
    查询共有多少id==共有多少行，NULL不能计数
        SELECT count(id) FROM t_user
    查询所有id的累加的和，所有用户的平均年龄..
        SELECT sum(id),avg(age) FROM t_user
注意：1.select max(age),name,id from t_user;逻辑扭曲，得到的结果是错误的---逻辑错误：又要查最大的年龄，又要查询所有的用户姓名，id，逻辑错误

注意：2.聚合操作，只返回一行结果。 组函数忽略NULL值，把空的值忽略掉，例如：有5个age，一个age是空值，那么使用组函数时按照四个计算，也就是说直接把null值的删除了

- group by：分组统计

    员工:employee 部门：department--dept_id
语法：SELECT .. FROM WHERE ... GROUP BY 列名
        在WHERE之后执行，即对WHERE筛选出的数据进行分组
1. 查询每个部门的最高工资,最低工资，最大年龄，部门id
        SELECT MAX(salary),MIN(salary),MAX(age),dept_id FROM t_employee GROUP BY dept_id
2. 查询每个部门的平均工资，最低工资，工资总和
        SELECT AVG(salary),MIN(salary),SUM(salary) FROM t_employee GROUP BY dept_id
3. 查询每个部门的员工数量，部门编号
        SELECT COUNT(id)，dept_id FROM t_employee GROUP BY dept_id
注意：分组查询中除了组函数外，只能查询作为分组条件的字段（列名）
-
指的是group by后面的列名，才可以放到select后面查询，如果在查个name的话，逻辑扭曲了（结果不正确）



- having子句：对分组（group by）查询的限制

```
语法：SELECT .. FROM  WHERE ... GROUP BY 列名   HAVING ...  order by
        在GROUP形成临时表之后，最终确定结果之前执行，即对每个临时表做筛选，满足having条件的临时表才可以将一条数据放入最终结果
eg：查询平均工资大于10000的部门的员工平均年龄和最高工资
    SELECT MAX(salary),AVG(age) FROM t_employee GROUP BY dept_id HAVING AVG(salary)>10000
eg：查询最小年龄大于20的部门的员工总数和部门id
    SELECT COUNT(id),dept_id FROM t_employee GROUP BY dept_id HAVING MIN(age)>20
注意：having和where的区别：where先执行，用来对总表做数据筛选，筛选完的结果在进行分组，having在where之后的group之后执行，对分组过程中的每个临时表做筛选（对group by执行后结果进行筛选）----------如果一个需求，where和having都可以实现，建议使用where，效率更高
------------------------
```

- 子句执行的顺序

```
执行顺序：from > where > group by > having > select > order by
语法顺序：select ..  from .. where .. group by .. having .. order by ..
```

- case子句

```
语法：
        case
            when bool表达式  then  结果1
            when bool表达式  then  结果2
            ...
            else 默认结果
        end as 别名（给上面的when取别名）
样例：
select
    id,name,salary,
    case
        when salary<10000  then '白领'
```

```
        when salary>10000 and salary<20000 then '银铃'
        when salary between 20000 and 50000 then '蓝领'
        else '金领'   #else可以省略
    end as level,
    id,   #id，name也可以放在这
    name
from
    t_person
```

- 子查询

一个查询作为另一个查询的一部分，称为子查询

1. 查询工资大于平均工资的员工信息（子中只有一个值=一行一列）

```
        select id,name from t_employee where salary > (select avg(salary) from t_employee)
```

2. 查询工资小于平均值的员工数量

```
        select count(id) from t_employee where salary < (select avg(salary) from
t_employee)
```

3. 查询平均工资大于10000的部门的员工信息(子中是多个值=单列多行)

```
        select id,name,age,salary from t_employee where dept_id in
        (select dept_id from t_employee group by dept_id having avg(salary)>10000)
```

补充：子查询中有多行多列（了解）

```
        select 别名.id,别名.name from (select * from t_employee) 别名
        select e.id,e.name from (select id,name from t_employee) e
```

注意:子查询的执行效率，很低

-

常用函数：

1. length():获取长度

```
        select name,length(name) from t_user
```

2. lcase():转为小写字符abcd

```
        select name,lcase(name) from t_user
```

3. ucase() :转为大写字符ABCD

```
        select name,ucase(name) from t_user
```

4. trim() :去除头部和尾部的空格

```
        select name,trim(name) from t_user
```

5. now()   :当前时间和日期

```
        select name,now() from t_user
```

6. curdate():当前日期
7. curtime():当前时间
8. date_format(now(),'%Y/%m%d %H:%i:%s')--日期格式化'：'也可以换成'-'

```
        select name,date_format(now(),'%Y/%m/%d %H:%i:%s') from t_user
```

9. database():当前数据库
10. user():当前用户名

```
        select user(),version()
```

11. version():当前服务器版本