数据库（MySQL）：

引论

数据库管理系统（DBMS）：网络应用服务端

1. 客户端可以自己写 ：未来写代码的时候
2. 也可以用别人写好的 : 第三方的工具 数据库管理软件的公司出版的官方客户端
3. 数据库管理系统本质上也是管理一堆文件

数据库管理员（DBA）

- 主要工作：
    1. 搭建数据库服务环境
    2. 用户的创建 权限的管理
    3. 性能\语句的优化
    4. 数据库的二次开发 : 让数据库具有公司的特质

主要软件：

MySQL：适用于小公司以及互联网企业

oracle：事业单位，金融企业

sql server

sqllite

主要名词：

DB：数据库--相当于文件夹

table：表--相当于文件

data ： 一条数据--相当于一行数据

分类：

1. 关系型数据库：mysql oracle sqlserver sqllite
2. 非关系型数据库：redis mongodb memcache hbase

安装：

1. 网址：https://www.mysql.com
2. 点击download跳转https://www.mysql.com/downloads
3. 跳转https://dev.mysql.com/downloads/点击Community选项
4. 点击MySQL Community Server进入https://dev.mysql.com/downloads/mysql/页面，再点击5.6版本的数据库
5. windows操作系统 点击5.6版本之后会跳转到https://dev.mysql.com/downloads/mysql/5.6.html#downloads 网址，页面如下，确认好要下载的版本和操作系统，点击Download
6. 可以不用登陆或者注册，直接点击No thanks,just start my download就可以下载了。
7. 下载的zip文件解压，将解压之后的文件夹放到任意目录下，这个目录就是mysql的安装目录
8. 打开目录，会看到my-default.ini配置文件，复制这个配置文件可以重命名为my.ini或者my.cnf（注意每行代码后不能有空格）

```
[mysql]
# 设置mysql客户端默认字符集
default-character-set=utf8
[mysqld]
#设置3306端口
port = 3306
# 设置mysql的安装目录
basedir=C:\Program Files\mysql-5.6.39-winx64
# 设置mysql数据库的数据的存放目录
datadir=C:\Program Files\mysql-5.6.39-winx64\data
# 允许最大连接数
max_connections=200
# 服务端使用的字符集默认为8比特编码的latin1字符集
character-set-server=utf8
# 创建新表时将使用的默认存储引擎
default-storage-engine=INNODB
```

9. 配置系统环境变量：在系统变量PATH后面添加: 你的mysql bin文件夹的路径（如C:\Program Files\mysql-5.6.41-winx64\bin）
10. 以管理员身份打开cmd窗口后，将目录切换到你解压文件的bin目录，输入mysqld install回车运行
11. 以管理员身份在cmd中输入:net start mysql就可以启动MySQL

sql常用语句

1. DDL：数据库定义语言
    1. 创建，删除，查看，修改
2. DML：数据库操作语言
    1. 数据的插入数据INSERT，删除数据DELETE，更新数据UPDATE，查询数据SELECT
3. DCL：数据库控制语句
    1. 权限相关的操作GRANT，REMOVE

操作

命令：

1. mysqld install; 配置数据库
2. net start mysql;启动数据库
3. mysql -uroot -p; 以root权限启动数据库，-p之后输入密码
4. mysql -uroot -u"IP地址" -p; 设置远程连接
5. set password = password('密码'); 设置密码
6. mysqladmin -u用户名 -p旧密码 password 新密码
7. show databases; 展示所有的数据库
8. show create table 表名; 查看表结构（会查看到创建表的语句，包括约束条件，主键等）
9. use 数据库名;切换数据库名
10. show variables like '%chara%';查看当前编码
    1. 临时修改（在客户端执行）：set xxxx = utf8;
    2. 永久解决：在my.ini 添加 set xxxx = utf8；
    3. 实时解决问题：create tables 表名（） charset = utf8;
11. 创建：
    1. select user(); 查看当前使用用户

2. create database 数据库名;创建数据库
3. create table 表名(字段名1 类型（条件）); 创建表，字段名不能一样

```
create table demo(num int,username char(12),password char(32));
```

4. insert into mysql.user(Host,User,Password) values("localhost","用户名",password("密码"));创建一个localhost账户用户，该账户只能在本地登录，不能字啊另一台机器上远程登录
5. insert into mysql.user(Host,User,Password) values("%","用户名",password("密码"));创建一个在任意一台电脑上都可以登录的账户，也可以指定某台机器可以在远程登录。

12. 删除具体操作：
    1. drop user 用户名@'%'; 删除账户
    2. drop user 用户名@'localhost';删除用户权限
    3. drop database 数据库名; 删除数据库
    4. drop table 表名；删除表
13. 权限：
    1. flush privileges;刷新权限
    2. grant all privileges on 数据库名.* to 用户名@localhost identified by '密码';授权给某个用户这个数据库的所有权限
    3. 格式：grant 权限 on 数据库.* to 用户名@登录主机 identified by "密码";
    4. grant select,update on 数据库名.* to 用户名@localhost identified by '密码';
    5. grant select,delete,update,create,drop on . to 用户名@"%" identified by "密码";授权用户拥有所有数据库的某些权限
    6. @"%" 表示对所有非本地主机授权，不包括localhost。（localhost地址设为127.0.0.1
    7. 对localhost授权：加上一句grant all privileges on 数据库名.* to 用户名@localhost identified by '密码';即可
    8. show grants for 'root'@'localhost';查看数据库中具体某个用户的权限
    9. GRANT ALL ON . TO `用户名`@`127.0.0.1` WITH GRANT OPTION;修改用户权限
14. 查看数据：
    1. select database() ;查看当前所在库
    2. select * from 表名; 查看表中所有数据
    3. SELECT DISTINCT CONCAT('User: '',user,''@'',host,'';') AS query FROM mysql.user; 查看数据库中的所有用户
15. desc 表名; / describe 表名；查看表结构
16. insert into 表名 values(数据)；表里添加数据
17. update 表名 set password='alex3714' where num=1;更新数据
18. delete from 表名 where num=1;删除表中数据、

权限：

1. usage：使用权限
2. select：查看数据
3. update：更新
4. insert：写入
5. delete：清除数据
6. all：所有权限
7. on后面跟数据库中的某个表
8. （*）代表所有的表（数据库.*）

基础：

1. 数据库的操作

1. create database 数据库名；创建一个数据库名，带有具体意义的英文名字
2. show databases; 查看有多少个数据库
3. use 数据库名; 切换数据库
4. select database(); 查看当前所在的库

2. 表的操作
    1. create table 英文表名（num int , username char (12),password char(32）);
    2. show tables;查看当前有多少表
    3. desc 表名; 查看表结构
    4. describe 表名; 查看表结构
    5. alter table 表名，修改表结构
3. 数据的操作
    1. insert into 表名 values(1,'alex','123'); 必须一一对应
    2. select * from 表名; 查看表中所有数据
    3. update 表名 set 数据名='xxxxx' where num = 1; 修改数据
    4. delete from 表名 where num = 1; 删除数据

类型：

1. 数字类型：
    1. 整数：int，tinyint。约束：unsigned

类型	大小	范围（有符号）	范围（无符号）unsigned约束	用途
TINYINT	1 字节	(-128，127)	(0，255)	小整数值
SMALLINT	2 字节	(-32 768，32 767)	(0，65 535)	大整数值
MEDIUMINT	3 字节	(-8 388 608，8 388 607)	(0，16 777 215)	大整数值
INT或INTEGER	4 字节	(-2 147 483 648，2 147 483 647)	(0，4 294 967 295)	大整数值
BIGINT	8 字节	(-9 233 372 036 854 775 808，9 223 372 036 854 775 807)	(0，18 446 744 073 709 551 615)	极大整数值

```
create table t1(i1 tinyint,i2 int);
默认创建的所有数据都是有符号的

create table t2(i1 tinyint unsigned,i2 int unsigned);
unsigned表示无符号
```

    2. 小数：float，double

FLOAT	4 字节 float(255,30)	(-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 字节 double(255,30)	(-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	对DECIMAL(M,D)，如果M>D，为M+2否则为D+2 double(65,30)	依赖于M和D的值	依赖于M和D的值	小数值

```
char(25)
    'abc              ' 浪费空间、节省时间
varchar(25)
    '3abc'              节省空间、浪费时间
create table t5(c1 char(5),c2 varchar(5));
```

2. 字符串：

类型	大小	用途
CHAR	0-255字节	定长字符串
VARCHAR	0-65535 字节	变长字符串
TINYBLOB	0-255字节	不超过 255 个字符的二进制字符串
TINYTEXT	0-255字节	短文本字符串
BLOB	0-65 535字节	二进制形式的长文本数据
TEXT	0-65 535字节	长文本数据
MEDIUMBLOB	0-16 777 215字节	二进制形式的中等长度文本数据
MEDIUMTEXT	0-16 777 215字节	中等长度文本数据
LONGBLOB	0-4 294 967 295字节	二进制形式的极大文本数据
LONGTEXT	0-4 294 967 295字节	极大文本数据

1. 定长，节省时间、浪费空间 char（255）
2. 变长、节省空间，浪费时间varchar（65535）

3. 时间类型：

类型	大小(字节)	范围	格式	用途
DATE	3	1000-01-01/9999-12-31	YYYY-MM-DD	年月日
TIME	3	'-838:59:59'/'838:59:59'	HH:MM:SS	时分秒
YEAR	1	1901/2155	YYYY	年份值
DATE TIME	8	1000-01-01 00:00:00/9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS	年月日时分秒
TIME STAMP	4	1970-01-01 00:00:00/2038 结束时间是第 **2147483647** 秒，北京时间 **2038-1-19 11:14:07**，格林尼治时间 2038年1月19日 凌晨 03:14:07	YYYYMMDD HHMMSS	混合日期和时间值，时间戳

1. now()：表示当前时间
2. datetime：年月日时分秒
3. date：年月日
4. time：时分秒
5. year：年
6. timestamp：年月日时分秒,不能为空，自动写当前时间，级连更新、表示范围小，

4. enum和set：

类型	大小	用途
ENUM	对1-255个成员的枚举需要1个字节存储; 对于255-65535个成员，需要2个字节存储; 最多允许65535个成员。	单选：选择性别
SET	1-8个成员的集合，占1个字节 9-16个成员的集合，占2个字节 17-24个成员的集合，占3个字节 25-32个成员的集合，占4个字节 33-64个成员的集合，占8个字节	多选：兴趣爱好

1. enum：单选
2. set：多选

单表查询：select * from 表 where 条件 group by 分组 having 过滤 order by 排序 limit n;

1. 语法：

   select distinct 字段1，字段2... from 表名 where 条件 group by 组名 having 筛选 order by 排序 limit 限制条数

   1. 找到表:from
   2. 拿着where指定的约束条件，去文件/表中取出一条条记录

3. 将取出的一条条记录进行分组group by，如果没有group by，则整体作为一组
4. 执行select（去重）：select * from 表名;
5. 将分组的结果进行having过滤
6. 将结果按条件排序：order by
7. 限制结果的显示条数

2. 优先级：from > where > group by> select > distinct> having>order by >limit
3. 避免重复：distinct:
    - select distinct xx from 表名;
4. 通过四则运算查询:
    - select xx(+ - * /) as 新名字 from 表名;
5. 重命名:
    1. select emp_name,salary*12 as annul_salary from employee;
    2. select emp_name,salary*12 annul_salary from employee;
6. concat()：用于连接字符串
    - select concat('姓名：',emp_name),concat('年薪:',salary*12) from employee;
    - CONCAT_WS() 第一个参数为分隔符，select concat_ws('|','a','b','c')用管道符分割数据
    - 结合CASE语句：case when语句 == if条件判断句

```
SELECT
        (
            CASE
            WHEN emp_name = 'jingliyang' THEN
                emp_name
            WHEN emp_name = 'alex' THEN
                CONCAT(emp_name,'_BIGSB')
            ELSE
                concat(emp_name, 'SB')
            END
        ) as new_name
    FROM
        employee;
```

7. where 约束：select xx,xxx from 表名 where xx=="值";
    1. 比较运算符：> < >= <= <> !=
        - 格式：select xx,xxx,xxxx from employee where 条件;
    2. between 80 and 100 值在80到100之间（包括80和100）
        - 格式：select xx,xxx,xxxx from employee where xx between xx and xx;
    3. in(80,90,100) 值是80或90或100
        - SELECT 字段 FROM employee
          WHERE 字段条件 in (xxxx,xx,xxxx,xxxx) ;
    4. like 'e%'： 通配符可以是%或 _ ，%表示任意多字符，_ 表示一个字符
        - select * from 表名 where 字段名 like 'xx%';
    5. regexp 正则匹配：select * from employee where emp_name regexp '^jin'
    6. 逻辑运算符：在多个条件直接可以使用逻辑运算符 and or not
    7. 关键字IS NULL(判断某个字段是否为NULL不能用等号，需要用IS)
        - select * from 表名 where 字段 is null;
8. GROUP BY分组聚合
    1. 单独使用:

1. select 字段名 from 表名 group by 字段名;
2. select * from 表 where 条件 group by 分组：加条件

2. GROUP BY关键字和GROUP_CONCAT()函数一起使用:
   - SELECT post,GROUP_CONCAT(emp_name) FROM employee GROUP BY post;#按照岗位分组，并查看组内成员名，
   - SELECT post,GROUP_CONCAT(emp_name) as emp_members FROM employee GROUP BY post;
3. GROUP BY与聚合函数一起使用
   1. select post,count(id) as count from employee group by post;#按照岗位分组，并查看每个组有多少人

9. 聚合函数：聚合函数聚合的是组的内容，若是没有分组，则默认一组
   1. count（字段名）：计数
   2. max(字段名)：最大值
   3. min（字段名）：最小值
   4. avg（字段名）:平均值
   5. sum（字段名）：求和
   6. 格式：
      1. select count/sum/max/min/avg(字段名) from 表名；
      2. select count/sum/max/min/avg(字段名) from 表名 where 条件；
10. having过滤(group by + 聚合函数)：
    1. 执行优先级从高到低：where > group by > having
    2. Where 发生在分组group by之前，因而Where中可以有任意字段，但是绝对不能使用聚合函数。
    3. Having发生在分组group by之后，因而Having中可以使用分组的字段，无法直接取到其他字段,可以使用聚合函数

       ```
       例子：查询各岗位内包含的员工个数小于2的岗位名、岗位内包含员工名字、个数
       select post,emp_name,count(id) from employee group by post having
       count(id)<2
       ```

11. order by排序
    1. 单列排序：
       1. select * from 表名 order by 字段名;默认升序
       2. select * from 表名 order by 字段名 asc; 升序
       3. select * from 表名 order by 字段名 desc;降序
    2. 多列排序：
       1. 例子：先按照age排序，如果年纪相同，则按照薪资排序
          1. select * from 表名 order by age,salary desc;
          2. select * from employee order by age desc,salary;
12. limit:限制查询记录数
    1. select * from 表 order by 列 limit n; 取前n条
    2. select * from 表 order by 列 limit m,n; 从m+1开始,取n条
    3. select * from 表 order by 列 limit n offset m; 从m+1开始,取n条

pymysql模块

1. 安装：pip install pymysql
2. python 内连接数据库
    1. 建立连接：conn
    2. 获取游标：cur
    3. 执行sql execute(sql,(可迭代类型的参数集))
        1. sql 是查：只设计文件的读操作
            1. fetchone（）--查找一个
            2. fetchmany（n）--查找n条数据
            3. fetchall() -- 查找全部
        2. 当操作数据增删改的时候--涉及到的是文件的写操作
            1. conn.commit()： 保证修改的数据生效
    4. 关闭游标和连接
        1. cur.close()
        2. conn.close()

```
import pymysql
conn = pymysql.connect(host='ip',user='',password='',databases='')
cur = conn.cursor()

#查
cur.fetchone()/fetchmany(n)/fetchall()

#增删改
conn.commit() #
conn.rollback()

cur.close()
conn.close()
```

sql注入：-- 注释之后的代码

多表查询

1. 连表：
    1. 内连接：所有不在条件匹配内的数据们都会被剔除连表
        - select * from 表名1，表名2 where 条件;
        - select * from 表名1 inner join 表名2 on 条件;
    2. 外连接：
        1. 左外连接：left join
            - select * from 表名1 left join 表名2 on 条件;（显示表名1中的所有数据）
        2. 右外连接right join
            - select * from 表名1 right join 表名2 on 条件;（显示表名2中的所有数据）
        3. 全外连接 full join
            - select * from 表名1 left join 表名2 on 条件 union select * from 表名1 right join 表名2 on 条件;
    3. 子查询：建议分开做

表的存储引擎：

1. show engines; 存储引擎
2. 存储引擎：
    1. 表结构：存放在一个文件中（硬盘）
    2. 表数据：存放在另一个文件中（内存）
    3. 索引：为了方便查找设计的一个机制
3. 种类：
    1. innodb：索引+数据 表结构，持久化存储
        1. 支持事务begin：一致性，n条语句的执行状态是一致的

```
begin #开启事务
select ......
update/delete ......
commit; #提交事务，解锁被锁住的数据
```

        2. 支持行级锁：只对涉及到修改的行加锁，利用并发的修改，但是对于一次性大量修改效率低下（表级锁：一次性加一把锁就锁住了整张表，不利用并发的修改，但是加锁速度比行锁的效率高）
        3. 支持外键约束：被约束表中的数据不能随意的修改，删除，约束字段要根据被约束表来使用数据
    2. myisam：索引，数据，表结构，支持持久化存储
        - 支持表级锁
    3. memory：表结构
        - 支持断电消失

表的约束

1. 非空约束：not null,默认不写的时候自动插入
2. unsigned ： 无符号的数字
3. 唯一约束：unique
    1. 联合唯一约束：unique(需要联合的列)
4. 外键约束：foreign key(字段名) references
    - 级联更新：on update cascade ,相关的数据会跟随变化
5. 设置默认值：default xx
6. 主键：primary key
    - 第一个被设置了非空+唯一约束的字段会被定义成主键 primary key
7. 对某一字段 自增：auto_increment
    - (truncate table 表名; 清空表且重置auto_increment),
    - (delete from 表名;清空表数据但不能重置auto_increment)
    - alter table 表名 auto_increment=数值

表的修改 alter table

1. 修改表明：alter table 表名 rename 表名；
2. 修改编码：alter table 表名 charset 编码;
3. 修改自增：alter table 表名 auto_increment 自增的位置；
4. 添加字段约束：alter table 表名 add 约束条件；
5. 修改约束：alter table 表名 add 字段名 类型（长度） 约束；
6. 修改字段名：alter table 表名 drop 字段名；
7. 修改字段名以及类型和约束条件：alter table 表名 change 字段名 新名字 类型（长度）约束

8. 修改表中的字段：alter table 表名 modify 字段名 新类型（新长度）约束
9. 修改字段名的位置：alter table 表名 add 字段名 类型（长度） 约束 first（备注：移动到第一个）/after 字段名（备注：移动到某个字段名之后）
10. 修改原字段的类型：after table 表名 change 字段名1 字段名1 类型（长度） 约束;

表与表之间的关系

1. 一对一：

```
create table class(id int primary key,cname char(26));

create table student(id int primary key,sname char(16),gid int
unique,foreign key(gid) references guest(id));
```

2. 一对多：foreign key

```
create table class(id int primary key,cname char(16));

create table student(id int primary key,sname char(16),cid int,foreign
key(cid) references class(id));
```

3. 多对多：

```
create table class(id int primary key,cname char(16));

create table teacher(id int primary key,tname char(16));

create table teacher_cls(id int,cid int,tid int,foreign key(cid) references
class(id),foreign key(tid) references teacher(id));
```

索引原理

1. 磁盘预读性原理
    1. linux 中一个block块大小是4096个字节
2. 树：根节点root，分支节点branch，叶子节点：leaf
    1. b树：balance 树
        1. 数据存储在分支节点和叶子节点上
        2. 导致了树的高度增加，找到一个数据的时间不稳定
        3. 在查找范围的时候不够便捷
    2. b+树：mysql中innodb存储引擎的所有的索引树都是b+树，是为了更好的处理范围问题在b树的基础上有所优化
        1. 数据不再存储在分支节点了，而是存储在叶子节点上（树的高度降低，找到所有数据的时间稳定）
        2. 在叶子节点与叶子节点之间添加的双向指针提高了再查找范围的效率
3. 索引的两种存储方式：
    1. 聚集（簇）索引：叶子节点会存储整行数据--innodb的主键中才会有（主键只可以创建一个的原因）
    2. 非聚集索引（辅助索引）：除了主键之外的普通索引都是辅助索引，一个索引没办法查询到整行数据，需要回聚集索引再查一次，俗称回表，数据不直接存储在索引的叶子节点
4. 索引优缺点：

    1. 优点： 加快查询速度
    2. 缺点：降低写的效率，占用更多的磁盘空间
5. 索引的创建：create index 索引名 on 表名（字段名）；
6. 索引的删除：drop index 索引名 on 表名;
7. 正确的使用索引：
    1. 只有对创建了索引的列进行条件筛选的时候效率才可以提高
    2. 索引对应的列做条件不能参与运算，不能使用函数
    3. 当某一列的区分度非常小（重复率高），不适合创建索引
    4. 当范围作为条件的时候，查询结果的范围越大越慢，越小越快。
    5. like关键字：如果使用%/ _ 开头都无法命中索引
    6. 多个条件：如果只有一部分创建了索引，条件用and相连，那么可以提高查询效率，如果用or相连，不能提高效率
    7. 联合索引：
        1. 条件不能用or
        2. 要服从最左前缀原则
        3. 不能从使用了范围的条件开始之后的索引都失效
8. 基础概念：
    1. explain ： 执行计划
        1. explain
    2. 覆盖索引 using index：在查询的过程中不需要回表

    3. 索引合并：