# 昨日内容回顾

并发编程

进程

运行中的程序,是系统的最小资源单位

创建进程的两种方式

方式1

from multiprocessing import Process

class MyProcess(Process):

def \_\_init\_\_(self,n):

super().\_\_init()

self.n = n

def run(self):

pass

if \_\_name\_\_ == '\_\_main\_\_':

p = MyProcess(10)

p.start()

方式2

def f1(n):

print(n)

if \_\_name\_\_ == '\_\_main\_\_':

p =Process(target=f1,args=(10,))

p.daemon = True #守护

p.start()

p.join() #主进程等待子进程

进程通信,Queue

from multiprocessing import Process,Queue

def f1(q):

q.get()

if \_\_name\_\_ == '\_\_main\_\_':

q = Queue(5) #先进先出的队列,5表示最大队列长度

q.put(1)

p =Process(target=f1,args=(q,))

p.start()

进程三状态:就绪 运行 阻塞

并发:伪并行, 多道技术, 任务切换+保存状态 IO不等待,遇到io自动切换

并行:两个进程同时运行,需要多个cpu

任务的提交方式

同步:类似于串行

异步:只管提交,不等待它的执行

任务的执行状态

阻塞:

非阻塞:

线程

cpu的最小执行单位,指令的集合

线程的两种创建方式

方式1

from threading import Thread

class MyThread(Thread):

def \_\_init\_\_(self,n):

super().\_\_init()

self.n = n

def run(self):

pass

if \_\_name\_\_ == '\_\_main\_\_':

p = Thread(10)

p.start()

def f1(n):

pass

if \_\_name\_\_ == '\_\_main\_\_':

t = Thread(target=f1,args=(1))

t.daemon = True

t.start()

t.join()

守护进程:主进程代码运行完毕,守护进程随之结束

守护线程:所有的非守护线程运行结束才会结束

锁\互斥锁\同步锁

进程:from multiprocessing import Process,Lock

线程:from threading import Thread,Lock

num = 10

def f1():

#with loc:

loc.acquire()

global num

n = num

n += 1

time.sleep(1)

num = n

loc.release()

if \_\_name\_\_ == '\_\_main\_\_':

loc = Lock()

t1 = Thread(target=f1,args=(loc,))

t2 = Thread(target=f1,args=(loc,))

t1.start()

t2.start()

降低了效率但是保证了数据安全

线程进程的效率对比

死锁现象:两边互相拿到了对方需要的锁,互相等待,同一个线程里面,在没有释放的情况下,连续使用的同一个锁

递归锁:Rlock,计数器,互斥锁

loc =Rock()

可以连续acquire,每次acquire,计数器加1,每次release,计数器减1,直到0的时候,其他线程或者进程,才能拿到这把锁

线程队列

队列有三种

import queue

先进先出的:

q = queue.Queue(5)

q.get()

q.put()

q.get\_nowait()

q.put\_nowait()

q.empty()

q.full()

q.qsize()

先进后出,后进先出

q = queue.LifoQueue(5)

优先级

q = queue.PriorityQueue()

q.put((1,'xx')) 数字越小,优先级越高,越优先被取出来

GIL锁

全局解释器锁,Cpython解释器的.同一时间同一进程只能有一个线程会执行

线程池进程池

from concurrent.futures import ThreadPoolExecutor,ProcessPoolExecutor

def f1():

return 20

def aa(m)

print(m)

if \_\_name\_\_ == '\_\_main\_\_':

t= ThreadPoolExecutor(max\_worker=5)

t= ProcessPoolExecutor(max\_worker=5)

ret = t.submit(f1,a....)

ret.result() #阻塞程序

t.shutdonw() #等待池子里面的任务全部执行完

t.submit(f1,a....).add\_done\_callback(aa)

asyncio(建议学习)

协程

单线程的多任务并发,又称微线程,协程的任务切换+保存状态是通过咱们代码执行控制的,线程和进程的并发是操作系统做的..

from gevent import monkey;monkey.patch\_all() #帮助gevent识别任务中的所有IO,gevent是遇到IO执行切换任务,内部基于一个greenlet模块,生成器

import gevent

import time

def f1():

time.sleep(1)

print(1)

def f2():

time.sleep(2)

print(2)

g1 = gevent.spawn(f1)

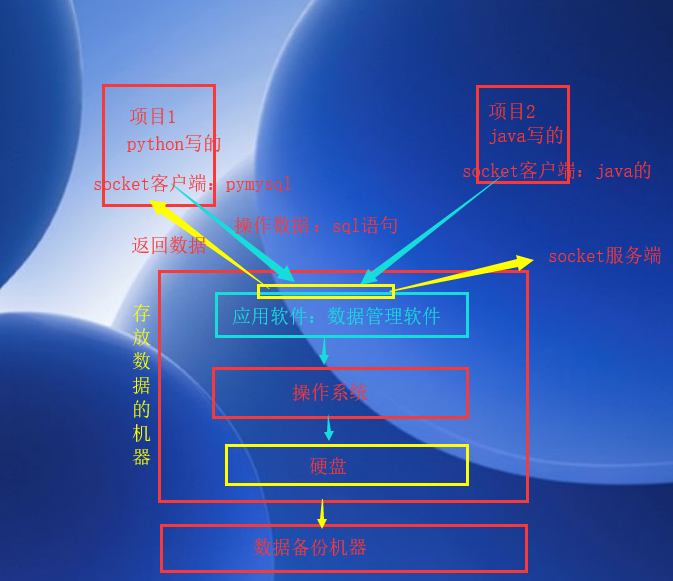
g2 = gevent.spawn(f2)

#g1.join()

gevent.joinall([g1,g2])

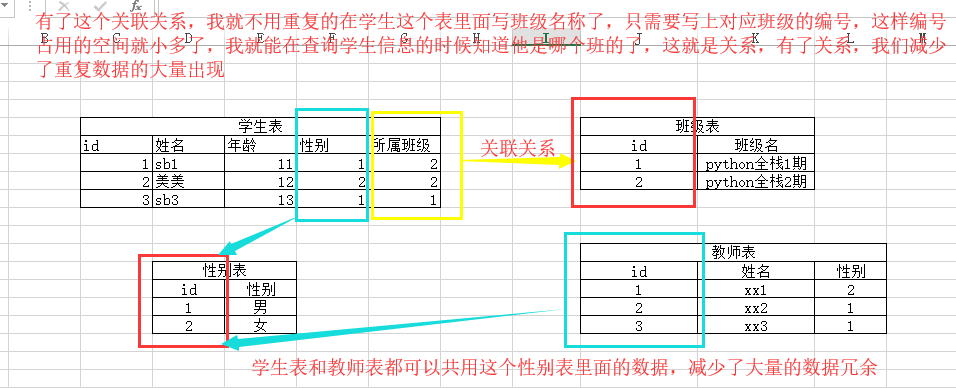
# 今日内容

## 介绍



RDBMS

## 关系型数据库和非关系型数据库



关系型数据库

oracle数据库

MySQL

MariaDB数据库

SQL Server数据库

不常见：DB2，PostgreSQL，Informix，Sybase等

非关系型数据库

键值（Key-Value）存储数据库

列存储

面向文档

图形（Graph）数据库

memcached（key-value）

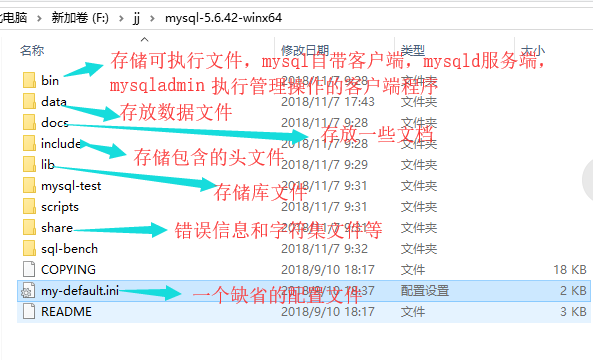
Redis（key-value）

MongoDB（Document-oriented

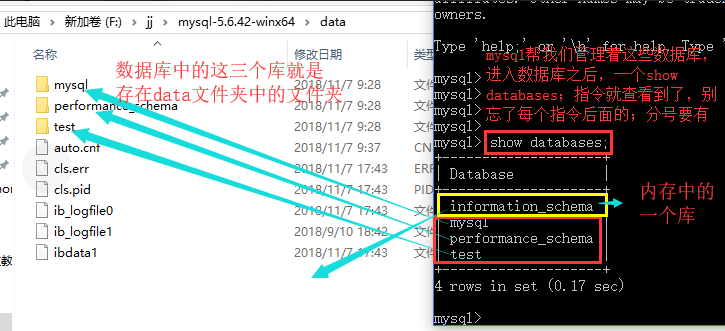
Apache Cassndra是一套开源分布式Key-Value存储系统。它最初由Facebook开发，用于存储特别大的数据。Facebook目前在使用此系统。

操作数据库的指令叫做sql,

## Mysql安装目录介绍



Mysql的内置库介绍：



[https://www.cnblogs.com/clschao/articles/9928223.html](https://www.cnblogs.com/clschao/articles/9928223.html" \t "_blank)

## 配置启动服务

**将bin目录配置环境变量**

**mysqld --install**

**启动指令：net start mysql**

关闭指令：net stop mysql

cmd下移除服务命令为：mysqld remove

完整连接指令mysql -h 127.0.0.1 -P 3306 -u root -p

root修改密码：

方式1

格式：mysql> set password for 用户名@localhost = password('新密码');   
例子：mysql> set password for root@localhost = password('123');

方式2

格式：mysqladmin -u用户名 -p旧密码 password 新密码   
例子：mysqladmin -uroot -p123456 password 123

方式3

mysql> use mysql;   use mysql的意思是切换到mysql这个库，这个库是所有的用户表和权限相关的表都在这个库里面，我们进入到这个库才能修改这个库里面的表。  
mysql> update user set password=password('123') where user='root' and host='localhost';   其中password=password('123') 前面的password是变量，后面的password是mysql提供的给密码加密用的，我们最好不要明文的存密码，对吧，其中user是一个表，存着所有的mysql用户的信息。

　　mysql> flush privileges;  刷新权限，让其生效，否则不生效，修改不成功。

## 修改字符集编码

[mysqld]

character\_set\_server=utf8  
collation-server=utf8\_general\_ci

[client]

default-character-set=utf8

[mysql]

user=root  
password=666  
default-character-set=utf8

**1、创建数据库**

**1.1 语法**

CREATE DATABASE 数据库名 charset utf8;

**1.2 数据库命名规则**

　　　　　　可以由字母、数字、下划线、＠、＃、＄

　　　　　　区分大小写

　　　　　　唯一性

　　　　　　不能使用关键字如 create select

　　　　　　不能单独使用数字

　　　　　　最长128位

　　　　　　# 基本上跟python或者js的命名规则一样

库相关操作

1 查看数据库

show databases;

show create database db1;

select database();

2 选择数据库

USE 数据库名

3 删除数据库

DROP DATABASE 数据库名;

4 修改数据库

alter database db1 charset utf8

flush privileges;

引擎：[https://www.cnblogs.com/clschao/articles/9953550.html](https://www.cnblogs.com/clschao/articles/9953550.html" \t "_blank)

创建表：

#语法：

create table 表名(

字段名1 类型[(宽度) 约束条件],

字段名2 类型[(宽度) 约束条件],

字段名3 类型[(宽度) 约束条件]

);

#注意：

1. 在同一张表中，字段名是不能相同

2. 宽度和约束条件可选、非必须，宽度指的就是字段长度约束，例如：char(10)里面的10

3. 字段名和类型是必须的

查看表结构

mysql> describe t1; #查看表结构，可简写为：desc 表名

mysql> show create table t1\G; #查看表详细结构，可加\G

# 二 数值类型

**1、整数类型**

　　　　整数类型：TINYINT，SMALLINT，MEDIUMINT，INT，BIGINT

2、浮点型

mysql> create table t1(x float(256,31));

mysql> create table t2(x double(255,30)); #建表成功

mysql> create table t3(x decimal(65,30)); #建表成功

mysql> insert into t1 values(1.1111111111111111111111111111111); #小数点后31个1

Query OK, 1 row affected (0.01 sec)

mysql> insert into t2 values(1.1111111111111111111111111111111);

Query OK, 1 row affected (0.00 sec)

mysql> insert into t3 values(1.1111111111111111111111111111111);

Query OK, 1 row affected, 1 warning (0.01 sec)

日期类型 DATE TIME DATETIME

mysql> create table t11(d date,t time,dt datetime);

mysql> desc t11;

测试：

mysql> insert into t11 values(now(),now(),now());

mysql> select \* from t11;

字符串

char类型 varchar类型



枚举和集合

枚举类型（enum）

An ENUM column can have a maximum of 65,535 distinct elements. (The practical limit is less than 3000.)

示例：

CREATE TABLE shirts (

name VARCHAR(40),

size ENUM('x-small', 'small', 'medium', 'large', 'x-large')

);

INSERT INTO shirts (name, size) VALUES ('dress shirt','large'), ('t-shirt','medium'),('polo shirt','small');

集合类型（set）

A SET column can have a maximum of 64 distinct members.

示例：

CREATE TABLE myset (col SET('a', 'b', 'c', 'd'));

INSERT INTO myset (col) VALUES ('a,d'), ('d,a'), ('a,d,a'), ('a,d,d'), ('d,a,d');

完整性约束：

PRIMARY KEY (PK) 标识该字段为该表的主键，可以唯一的标识记录

FOREIGN KEY (FK) 标识该字段为该表的外键

NOT NULL 标识该字段不能为空

UNIQUE KEY (UK) 标识该字段的值是唯一的

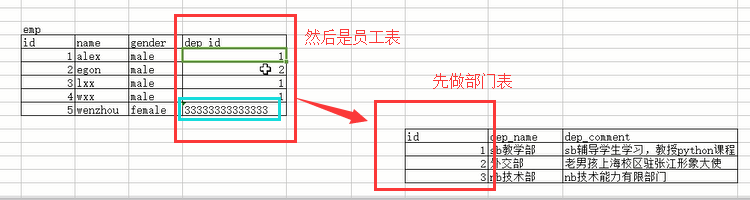
AUTO\_INCREMENT 标识该字段的值自动增长（整数类型，而且为主键）

DEFAULT 为该字段设置默认值

UNSIGNED 无符号

ZEROFILL 使用0填充

Foreigin key



　　　　1、先要建立被关联的表才能建立关联表

　　　　2、在插入数据记录的时候，要先想被关联表中插入数据，才能往关联表里面插入数据

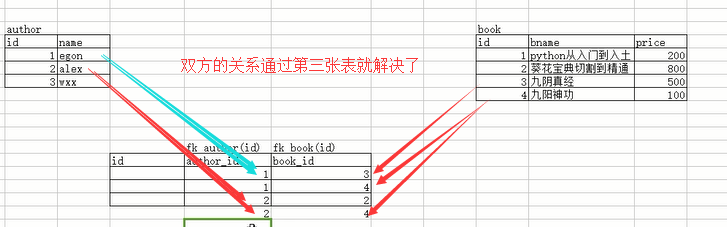
　　　　3、更新或者删除数据的时候，都需要考虑关联表和被关联表的关系

　　　　　　解决方案：

　　　　　　　　a.删除表的时候，先删除关联表，再删除被关联表

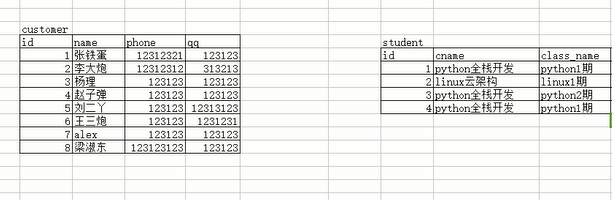
　　　　　　　　b.重建表的时候，在加外键关联的时候加上这两句：on delete cascade 和 on update cascade

多对多关系



一对一

客户表和学生表



mysql> alter table e3 drop ee\_id; #直接删除外键字段是不可以的

ERROR 1553 (HY000): Cannot drop index 'ee\_id': needed in a foreign key constraint

mysql> alter table e3 drop foreign key e3\_ibfk\_1; #通过上面的方法找到这个表的外键字段，然后先解除外键字段的关系，才能删除外键字段

mysql> alter table e3 drop ee\_id;

Query OK, 0 rows affected (0.65 sec)

mysql> desc e3;

+-------+----------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+----------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| xx | char(11) | YES | | NULL | |

+-------+----------+------+-----+---------+----------------+

2 rows in set (0.10 sec)  
  
#看添加外键字段和外键关联：  
首先创建一个e2表，包含一个id字段，别忘了id字段最少也要是unique属性，primary key当然最好啦

  mysql> alter table e3 add ee\_id int;  
  Query OK, 0 rows affected (0.64 sec)  
  Records: 0 Duplicates: 0 Warnings: 0

  mysql> alter table e3 add foreign key(ee\_id) references e2(id);  
  Query OK, 0 rows affected (0.83 sec)  
  Records: 0 Duplicates: 0 Warnings: 0

  #添加关联删除和关联更新的操作：当删除主表数据的时候，从表中有关的数据都跟着删除，当主表的关系字段修改的时候，从表对应的关系字段的值也更着更新。

　　alter table 从表 add foreign key(从表字段) references 主表(主表字段) on delete cascade on update cascade;

  #另外，能够作为主表(也就是多对一关系的那个一表的被关联的那个字段)的关系字段的约束最少要是唯一的unique属性。

## 单表查询

## 多表查询

索引

添加主键索引:

创建的时候添加: 添加索引的时候要注意,给字段里面数据大小比较小的字段添加,给字段里面的数据区分度高的字段添加.

聚集索引的添加方式

创建的是添加

Create table t1(

Id int primary key,

)

Create table t1(

Id int,

Primary key(id)

)

表创建完了之后添加

Alter table 表名 add primary key(id)

删除主键索引:

Alter table 表名 drop primary key;

唯一索引:

Create table t1(

Id int unique,

)

Create table t1(

Id int,

Unique key uni\_name (id)

)

表创建好之后添加唯一索引:

alter table s1 add unique key u\_name(id);

删除:

Alter table s1 drop index u\_name;

普通索引:

创建:

Create table t1(

Id int,

Index index\_name(id)

)

Alter table s1 add index index\_name(id);

Create index index\_name on s1(id);

删除:

Alter table s1 drop index u\_name;

DROP INDEX 索引名 ON 表名字;