信安之路

- 信安之路
  - 第02周
  - 前言
  - 1.概念（推荐）
    - 1.1.关系型数据库
      - 1.1.1.代表
      - 1.1.2.特性
      - 1.1.3.不足
    - 1.2.非关系型数据库
      - 1.2.1.特性
        - CA
        - CP
        - AP
        - 扩展
      - 1.2.2.代表
  - 2.基础
    - 2.1.传统概念
    - 2.2.字段类型（含异同）
  - 3.建表
    - 3.1.知识概括
    - 3.2.修改SQL_Mode（MySQL系）
      - 3.2.1.默认值
      - 3.2.2.开发经验
      - 3.2.3.配置修改
    - 3.3.MySQL
      - 3.3.1.创建、删除数据库
      - 3.3.2.创建、删除表
      - 3.3.3.修改表
    - 3.4.SQLServer
      - 3.4.1.创建、删除数据库
      - 3.4.2.创建、删除表
      - 3.4.3.修改表
    - 3.5.区别
  - 4.CURD
    - 4.1.MySQL
      - 4.1.1.执行流程
      - 4.1.2.增删改
      - 4.1.3.查询

        - 4.1.4.视图
        - 附录
    - 4.2.SQLServer
        - 4.2.1.执行流程
        - 4.2.2.增删改
        - 4.2.3.查询
        - 附录2
- 5.扩展
    - 5.1.MySQL
        - other
    - 5.3.SQLServer
        - others

第02周

Code： https://github.com/lotapp/BaseCode/tree/master/safe

前言

本周需要自行研究学习的任务贴一下：

第二阶段目标：sql 基础学习

学习 web 安全，sql 注入是必学的，因为 sql 注入问题是多年来一直位居榜首的 web 漏洞，而学习 sql 注入，核心的基础是 sql，所以在接下来的一段时间以学习 sql 为主要目标。

sql 注入可以分为几个阶段，比如：检测是否存在注入、通过注入漏洞获取数据、通过注入漏洞获取权限，根据这个三个不同阶段的要求进行划分。

第二周：认识 sql 并学习数据库的基础操作

1、什么是关系型和非关系型数据库，两者都包含哪些种类的数据库（理解两者的区别）

2、选择一种关系型数据库进行学习（选择自己不熟悉的进行学习，因为不同的数据库，其特性也不同，所以可以选择不熟悉或者感兴趣进行研究学习）

3、学习数据库中的字段类型并创建库和用户表，需要包含所有字段类型（主要熟悉数据库的基本使用，可以自由创建、删除、修改数据库和表）

4、学习数据库的增删改查，记录学习过程（重点是 sql 语句的理解）并形成报告（最终结果）

1.概念（推荐）

数据库系列去年就开始陆陆续续的发文，这周任务简单带过，概念部分我更新了一下，其他部分看扩展吧~

1.1.关系型数据库

引用百科的一段 `抽象` 描述：

> “关系型数据库，是指**采用了关系模型**来组织数据的数据库，其**以行和列的形式存储数据**，以便于用户理解，关系型数据库这一系列的行和列被称为表，一组表组成了数据库。用户通过查询来检索数据库中的数据，而查询是一个用于限定数据库中某些区域的执行代码。关系模型可以简单理解为二维表格模型，而一个关系型数据库就是由二维表及其之间的关系组成的一个数据组织。”

通俗讲就是：**现实中的东西抽象成一个个关系，然后存储在一张张行列组成的表中，这些表就组成了关系型数据库**

> PS：重点就是各数据之间的 `关系` （Join）

1.1.1.代表

最经典的莫过于： `MySQL` 、 `SQLServer` 、 `PostgreSQL` 、 `SQLite` 、 `Oracle`

1.1.2.特性

先看看传统数据库的好处：

1. 通过事务保持数据一致
2. 可以Join等复杂查询
3. 社区完善（遇到问题简单搜下就ok了）

最典型的特征就是：**事物的ACID特性**

> PS：抽象的就不说了，举个例子来说明 `ACID` ：

1. A：**原子性**（Atomic）
    - 小明转账1000给小张：小明-=1000 => 小张+=1000，这个 **（事务）是一个不可分割的整体**，如果小明-1000后出现问题，那么1000得退给小明
    - PS：化学里面原子的定义是啥? ==> `化学反应不可再分的基本微粒`
2. C：**一致性**（Consistent）
    - 小明转账1000给小张，必须保证小明+小张的**总额不变**（假设不受其他转账(事务)影响）
    - PS：可以用初三化学中的 `能量守恒` 来理解
3. I：**隔离性**（Isolated）
    - 小明转账给小张的同时，小潘也转钱给了小张，需要保证他们**相互不影响**（主要是并发情况下的隔离）
    - PS：理想各个 `国家互不干涉内政`
4. D：**持久性**（Durable）
    - 小明转账给小张，银行要**留有记录**，即使以后扯皮也可以拉流水账【事务执行成功后进行的持久化（就算数据库之后挂了也能通过Log恢复）】
    - PS：理想银行每次 `交易留的底单` ，这个就是持久化的表现

1.1.3.不足

1. **修改表结构**（包括建立索引）**导致长时间不能更新数据**（上了表锁）
2. **列不固定**时新增或删除表很繁琐（同样面临表锁的问题）
    - PS：一般设计数据库不可能那么完善，都是后期越来越完善，就算自己预留了 `保留字段` 也容易因为预留名的不确定性容易出错
3. **简单查询不能快速返回**（需要先解析SQL，然后还有诸如表进行加锁解锁这些额外开销，然后才能查找）
4. **大量数据写入**比较麻烦
    1. 数据量不大还好，批量写入即可
    2. 可是本身数据量就挺大的，进行了 `主从复制` ，读数据在 `Salver` 进行到没啥事，但是大量写数据库怼到 `Master` 上去就吃不消了，必须得加主数据库了。
    3. 加完又出问题了：虽然把主数据库一分为二，但是容易发生 `数据不一致` （同样数据在两个主数据库更新成不一样的值），这时候得结合分库分表，把表分散在不同的主数据库中。
    4. 完了吗？NoNoNo，想一想表之间的Join咋办？岂不是要跨数据库和跨服务器join了？简直就是拆东墙补西墙的节奏啊，所以各种中间件就孕育而生了

扩充：现在对**修改表结构**是这么的**解决方案**：

> PS：虽然都是借助工具来操作但原理还是要了解下： `先创建新表，然后旧表中创建一个触发器（处理产生的新数据），然后把一条条数据转移到新表中，接着删除旧表，改名新表` （和旧表名一样）

1.2.非关系型数据库

引用百科一段 `抽象` 描述：

> “非关系型数据库，又被称为NoSQL（Not Only SQL），主要是指非关系型、分布式、不提供ACID的数据库设计模式”

通俗化讲就是：**为了弥补SQL的不足而创建的**（可以逆推NoSQL的优势）

> PS：你可以理解为:NoSQL就是对原来SQL的扩展补充（ `NewSQL = SQL + NoSQL` ）

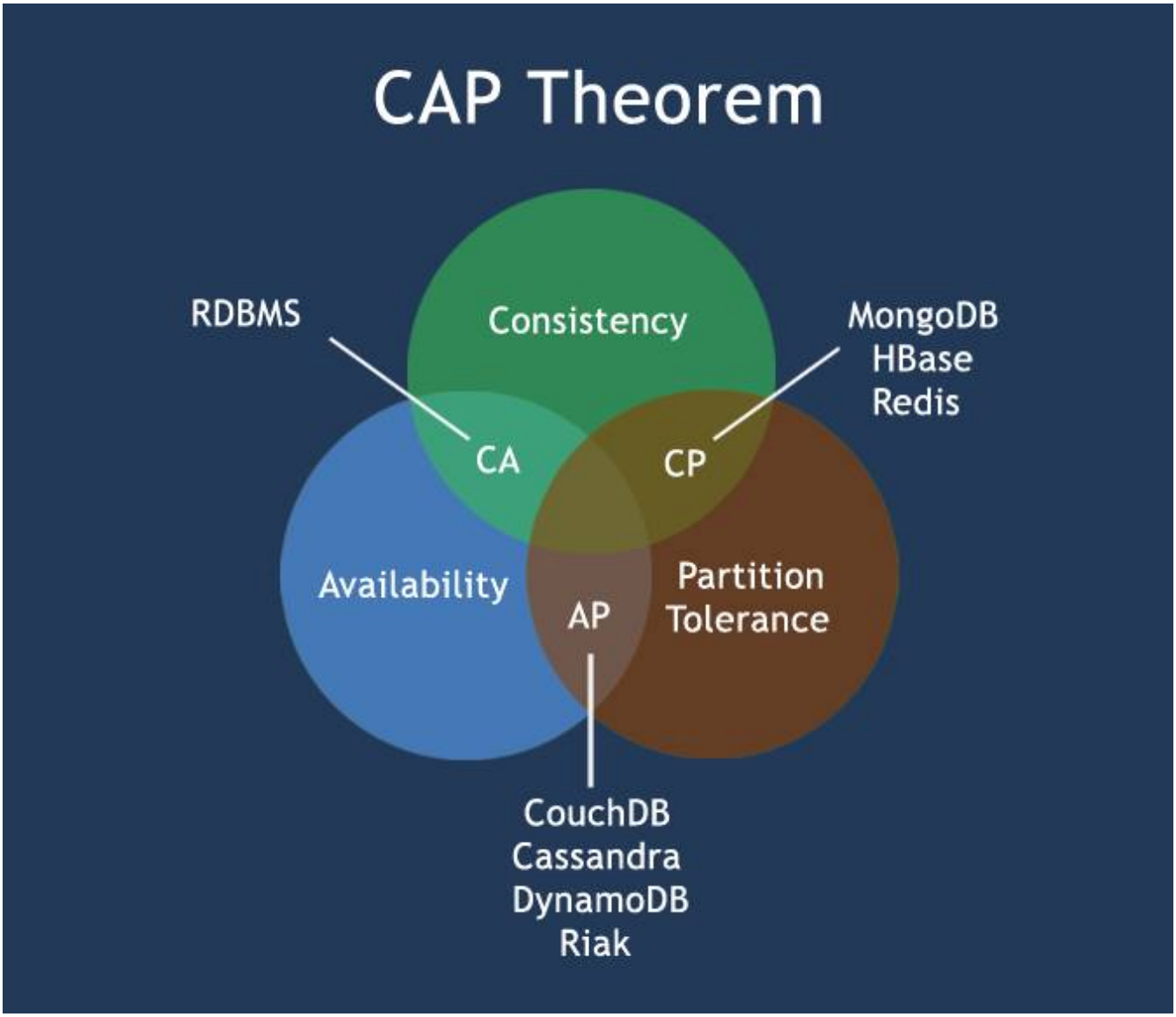
1.2.1.特性

先说说优势：

1. **易于数据分散**：NoSQL不支持Join，各个数据设计都是独立的，很容易就把数据分散到多个服务器上
2. **轻松支持大量数据写入**：没有那些乱七八糟的关系，横向扩展很轻松
3. **多样性**：NoSQL根据需求拆分成很多种类，使用非常灵活方便

NoSQL的特性就是 `CAP` ：

> PS： `CAP` 是分布式系统需要考虑的三个指标，数据共享只能满足两个而不可兼得：



1. C：一致性（**C**onsistency）
    1. 所有节点访问同一份最新的数据副本（在分布式系统中的所有数据备份，在同一时刻是否同样的值）
    2. eg：分布式系统里更新公告后，某个用户都应该读取最公告
2. A：可用性（**A**vailability）
    1. 在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求。（对数据更新具备高可用性）
    2. eg：分布式系统里每个操作总能在一定时间内返回结果（挂几个服务器也不影响）
3. P：分区容错性（**P**artition Toleranc）
    1. 以实际效果而言，分区相当于对通信的时限要求。系统如果不能在时限内达成数据一致性，就意味着发生了分区的情况，必须就当前操作在C和A之间做出选择。
    2. eg：分布式系统里，存在网络延迟的情况下依旧可以接受满足一致性和可用性的请求

CA

代表：**传统关系型数据库**

如果想避免分区容错性问题的发生，一种做法是将所有的数据（与事务相关的）都放在一台机器上。虽然无法100%保证系统不会出错，但不会碰到由分区带来的负面效果（会严重的影响系统的扩展性）

作为一个分布式系统，放弃P，即相当于放弃了分布式，一旦并发性很高，单机服务根本不能承受压力。像很多银行服务，确确实实就是舍弃了P，只用高性能的单台小型机保证服务可用性。（**所有NoSQL数据库都是假设P是存在的**）

CP

代表：`Zookeeper`、`Redis`（分布式数据库、分布式锁）

> PS：**NoSQL大部分都是CP**

相对于放弃“分区容错性”来说，其反面就是放弃可用性。`一旦遇到分区容错故障，那么受到影响的服务需要等待数据一致`（**等待数据一致性期间系统无法对外提供服务**）

AP

代表：**`DNS数据库`**（`IP和域名相互映射的分布式数据库`，联想修改IP后为什么`TTL`需要10分钟左右保证所有解析生效）

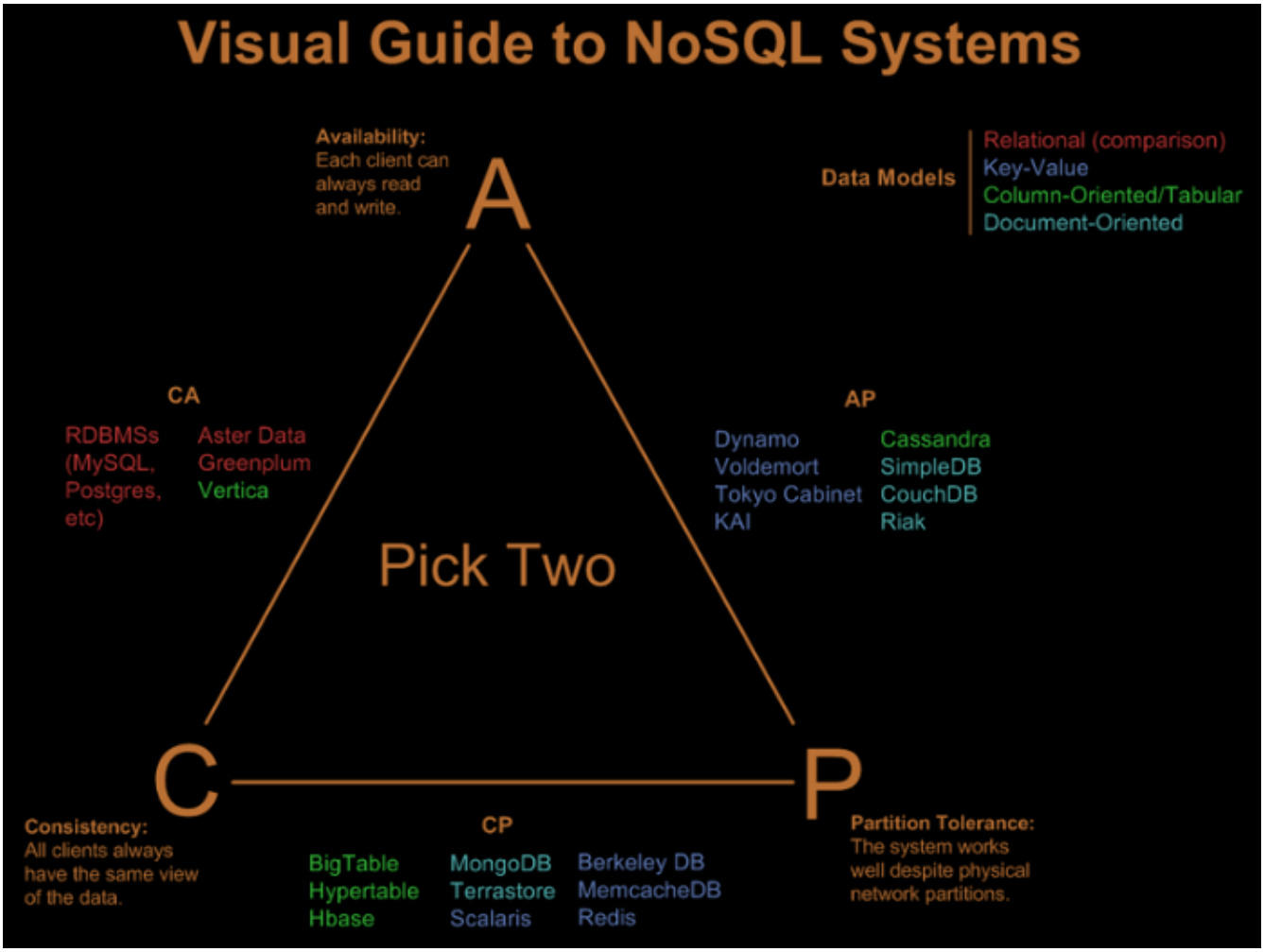
扩展

放弃强一致，保证最终一致性。所有的`NoSQL`数据库都是介于`CP`和`AP`之间，尽量往`AP`靠

> PS：**传统关系型数据库注重数据一致性，而对NoSQL来说可用性和分区容错性优先级高于数据一致性**

不同数据对一致性要求是不一样的，eg：

1. 用户评论、弹幕这些对一致性是不敏感的，很长时间不一致性都不影响用户体验
2. 像商品价格等对一致性有很高要求，容忍度铁定低于10s，就算使用了缓存在订单里面价格也是最新的
    - PS：平时注意下JD商品下面的缓存说明，JD尚且如此，其他的就不用说了



传统关系型数据库一般都是使用悲观锁的方式，但是例如秒杀这类的场景就玩不转了，这时候往往就使用乐观锁了（CAS机制），CAP不能同时满足的时候也就剩下这两个方案了：

1. **强一致性**：无论更新在哪个副本上，之后对操作都要能够获取最新数据。多副本数据就需要 **分布式事物来保证数据一致性** 了（这就是问什么项目里面经常提到的原因）
2. **最终一致性**：在这种约束下保证用户最终能读取到最新数据。举几个例子：
    1. 因果一致性：A、B、C三个独立进程，A修改了数据并通知了B，这时候B得到的是最新数据。因为A没通知C，所以C不是最新数据
    2. **会话一致性**：用户自己提交更新，他可以在会话结束前获取更新数据，会话结束后（其他用户）可能不是最新的数据（提交后JQ修改下本地值，不能保证数据最新）
    3. **读自写一致性**：和上面差不多，只是不局限在会话中了。用户更新数据后他自己获取最新数据，其他用户可能不是最新数据（有一定的延迟）
    4. 单调读一致性：用户读取某个数值，后续操作不会读取到比这个数据还早的版本（新的程度>=读取的值）
    5. **单调写一致性**（时间轴一致性）：所有数据库的所有副本按照相同顺序执行所有更新操作（有点像 `Redis` 的 `AOF` ）

1.2.2.代表

常用的已经加粗:

1. **键值数据库**： **Redis**、MemCached...
    - PS：现在 `LevelDB` 和 `RocksDB` 也是比较高效的解决方案
    - PS：360开发的 `pika` （redis的补充）可以解决Redis容量过大的问题

2. **文档数据库**：**MongoDB**、ArangoDB、CouchBase、CouchDB、RavenDB...
    - PS：Python有一款基于json存储的轻量级数据库：`tinydb`
3. 列式数据库：**Cassandra**、**HBase**、BigTable...
    - PS：小米的**Pegasus**计划取代HBase
4. **搜索引擎系**：**Elasticsearch**、Solr、Sphinx...
    - PS：这几年用Rust编写的轻量级搜索引擎`sonic`很火
5. 图形数据库：**Neo4J**、**Flockdb**、**ArangoDB**、OrientDB、Infinite Graph、InfoGrid...
    - PS：基于Redis有一款图形数据库用的也挺多：`RedisGraph`
    - PS：随着Go的兴起，这些图形数据库很火：**Cayley**、`Dgraph`、`Beam`

PS：项目中最常用的其实就是`Redis`、`Elasticsearch`、`MongoDB`

2.基础

最先接触的是SQLServer，而后才是MySQL系，下面我尽量使用通用SQL

2.1.传统概念

来说说传统概念：

1. 关系型数据库中的**关系**：表（行、列）
2. **设计范式**：
    - 第1范式：字段是原子性的
    - 第2范式：每个表需要一个主键
    - 第3范式：任何表都不应该有依赖于其他**非主键**表的字段

- **DDL**：数据定义语言(Data Defination Language)
    - `create、drop、alter`
- **DML**：数据操作语言(Data Manipulation Language)
    - `insert、delete、update、select`
- **DCL**：数据库控制语言（Data Control Language）
    - `grant`(授权)、`revoke`(回收)

PS：`CURD`（定义了用于处理数据的基本原子操作）：创建（Create）更新（Update）读取（Retrieve）删除（Delete）操作

2.2.字段类型（含异同）

官方文档：

- https://mariadb.com/kb/en/library/data-types
- https://dev.mysql.com/doc/refman/5.7/en/data-types.html

以`MariaDB`为例，简单列举下常用类型：(倾体说明和`MySQL`不一样)

1. 字符型：

1. 定长字符型：
    - `char()` ：不区分字符大小写类型的字符串，`max：255个字符`
    - binary()：区分字符大小写类型的二进制字符串
2. 变长字符型：
    - `varchar()` ： 不区分字符大小写类型的字符串
        - max：65535（2^16 - 1）个字节（`utf8编码下最多支持21843个字符`）
        - 可以理解为 `SQLServer` 的 `nvarchar`
    - varbinary()：区分字符的大小写类型的二进制字符串
3. 对象存储：
    - `text` ：不区分字符大小写的不限长字符串
        - 最大长度为65,535（2^16 - 1）个字符
        - 如果值包含多字节字符，则有效最大长度减少
    - blob：区分字符大小写的不限长二进制字符串
4. 内建类型：（不推荐使用）
    - enum：单选字符串数据类型，适合表单中的 `单选值`
    - set：多选字符串数据类型，适合表单的 `多选值`
    - **PS：`MySQL系` 独有，`SQLServer` 没有**
2. 数值型：
    1. 精确数值型：
        - 整型：int
            1. *bool：布尔类型*（MySQL没有）
                - **PS：`SQLServer` 是 `bit`**
                - **相当于 `MySQL` 的 `tinyint(1)`**
            2. `tinyint` ：微小整型（1字节，8位）
                - `[-2^7, 2^7)`（`-128 ~ 127`）
                - 无符号：`[0, 2^8)`（`0 ~ 255`）
            3. smallint（2bytes，16bit）：小整型
                - 无符号：`0 ~ 65535`
            4. mediumint（3bytes,24位）：中等整型
                - `PS：SQLServer中没这个类型`
            5. `int`（4bytes，32bit）
                - `[-2^31, 2^31)`，`[-2147483648,2147483648)`
                - 无符号：`[0, 2^32)`，`[0,4294967296)`
            6. `bigint`（8bytes，64bit）
                - `[-2^63, 2^63)`
                - 无符号：`[0, 2^64)`
    2. 浮点类型：
        - float：单精度浮点数（4字节）
        - `double` ：双精度浮点数（8字节）
            - **PS：`SQLServer` 的 `float` 类型相当于 `MySQL` 的 `double`**

        - `decimal`：精确小数类型（比double更精确）
            - 钱相关用的比较多：`decimal(位数,小数点位数)`
            - eg：`decimal(3,2)` => `x.xx`
3. **日期和时间类型**：（和 `MySQL` 一样）
    1. date：日期（3bytes）
    2. time：时间（3bytes）
    3. year：年
        - eg：`year(2)`：`00~99 (1bytes)`
        - eg：`year(4)`：`0000~9999 (1bytes)`
        - **PS：`SQLServer没有这个类型`**
    4. `datetime`：既有时间又有日期（8bytes）
    5. `timestamp`：时间戳（4bytes）【精度更高】
4. 修饰符：
    - 所有类型都适用：
        - 是否为null：`null` | `not null`
        - 默认值：`default xxx_value`
        - 主键：`primary key`
        - 唯一键：`unique key`
    - 数值类型适用：
        - **无符号：`unsigned`**（MySQL系独有）
        - 自增长：`auto_increment`（一般只用于整型，MSSQL是 `identity`）
            - 获取ID：`last_insert_id()`
    - PS：**多列设置**：
        1. 主键：`primary key(xx,...)`
        2. 唯一键：`unique key(xx,...)`
        3. 索引：`index index_name (xx,...)`

PS：现在新版本数据库兼容了SQLServer的 `nvarchar` 写法（`执行成功后数据类型变成varchar`）【不推荐使用】

课后拓展：

MySQL：char、varchar、text的区别

- https://dev.mysql.com/doc/refman/5.7/en/char.html
- https://blog.csdn.net/brycegao321/article/details/78038272

3.建表

3.1.知识概括

1. 创建数据库：
    - `create database [if not exists] db_name;`

2. 删除数据库：
    - `drop database [if exists] db_name;`
3. 创建表：
    - `create table [if not exists] tb_name(列名1,数据类型 修饰符,列名2,数据类型 修饰符);`
4. 删除表：
    - `drop table [if exists] db_name.tb_name;`
5. 修改表：
    1. 字段
        - 添加字段：add
            - `alter table tb_name add 列名 数据类型 修饰符 [first | after 列名];`
            - **PS：SQLServer没有`[first | after 列名]`**
        - 修改字段：alter、change、modify
            - 修改字段名：`alter table tb_name change 旧列名 新列名 类型 类型修饰符`
            - 修改字段类型：`alter table tb_name modify 列名 类型 类型修饰符`
            - 添加默认值：`alter table tb_name alter 列名 set default df_value`
        - 删除字段：drop
            - `alter table tb_name drop 字段名`
    2. 索引
        - 添加索引：add（常用：**`create index index_name on tb_name(列名,...)`**；）
            - `alter table tb_name add index [ix_name] (列名,...);`
            - 添加唯一键：`alter table tb_name add unique [uq_name] (列名,列名2...);`
            - **PS：不指定索引名字，默认就是第一个字段名**
        - 删除索引：drop（常用：**`drop index index_name on tb_name`**）
            - `alter table tb_name drop index index_name;`
            - 删除唯一键：`alter table tb_name drop index uq_name;`
            - **PS：唯一键的索引名就是第一个列名**
        - **PS：一般在经常用做查询条件的列设置索引**

3.2.修改SQL_Mode（MySQL系）

通俗讲：SQL_Mode就是设置SQL语法检测的级别

3.2.1.默认值

MariaDB 5.x的SQLMode一般是空（没什么约束）

> PS：`select version();`可以查看DB的版本，`select @@sql_mode;`可以查看SQL_Mode的值

MySQL 5.7.27的SQLMode默认是：
`ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION`

简单解析一下：

1. `ONLY_FULL_GROUP_BY`：使用`group by`时查询的字段必须是`group by`子句的列

    - eg：`select count(url),name from file_records group by url;`
    - 使用了name字段，name不是聚合函数，那必须在group by中写一下（不然就报错）
2. `STRICT_TRANS_TABLES`：对所有支持事物类型的表做严格约束（线上版本一般都是这个）
3. `NO_ZERO_IN_DATE` 和 `NO_ZERO_DATE`：防止 `0000-00-00` 插入日期中
4. `ERROR_FOR_DIVISION_BY_ZERO`：分母为0则报错（默认是警告）
5. `NO_AUTO_CREATE_USER`：创建用户的时候必须 `identified by 'pass'` 来指定密码
    - eg：`create user bryan@'%' identified by '含大小写字母+数字的密码';`
6. `NO_ENGINE_SUBSTITUTION`：建表的时候指定不可用存储引擎会报错

3.2.2.开发经验

PS：项目开发一般设置为 `traditional`（使用传统模型，不允许对非法值做插入操作），线上一般使用 `strict_trans_tables` 来提高并发（阿里云默认值）

3.2.3.配置修改

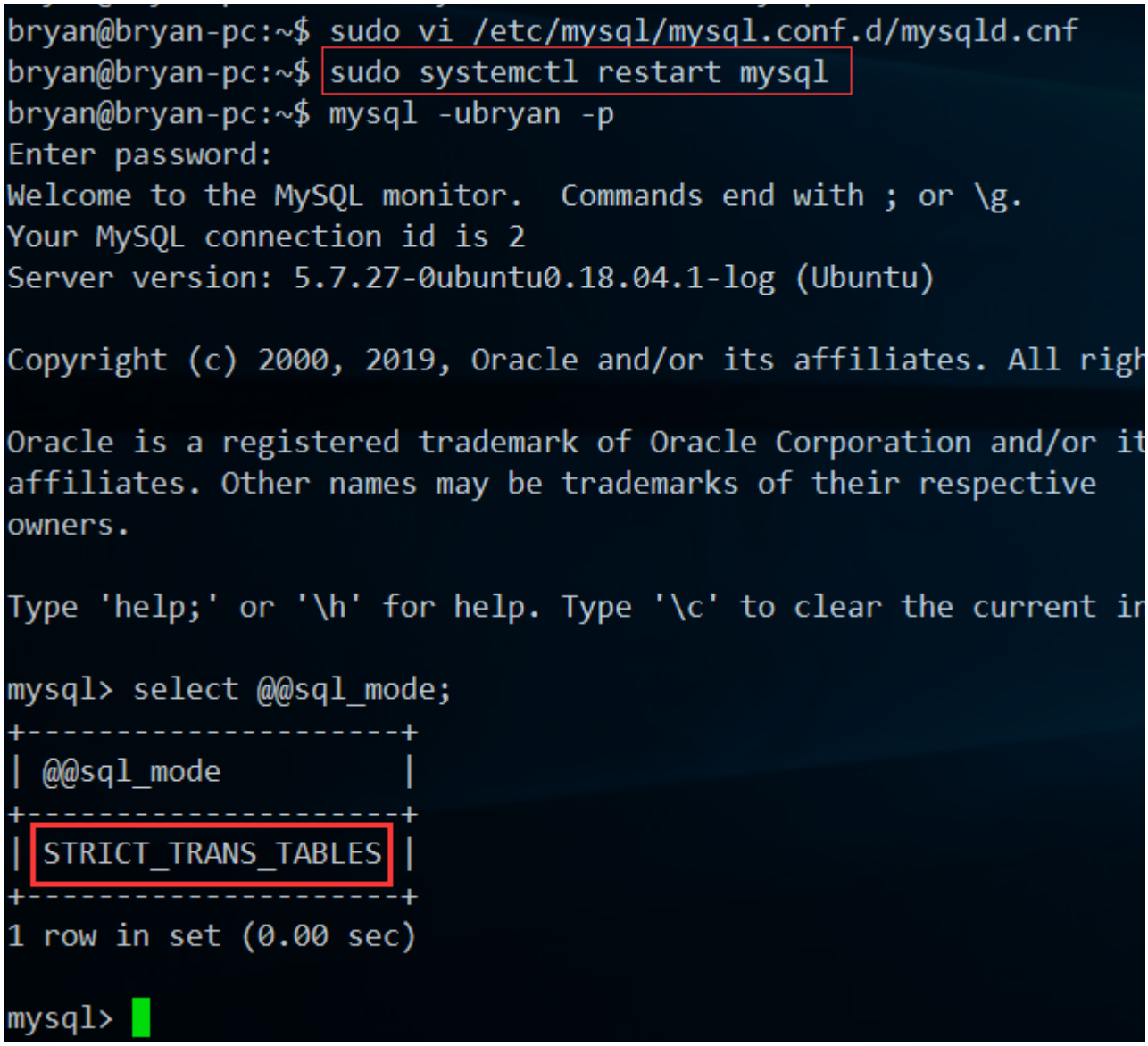
以Ubuntu为例：`sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf`（`sql_mode='strict_trans_tables'`）

> 然后重启mysql使配置生效：`sudo systemctl restart mysql`

```
[mysqld]
#
# * Basic Settings
#
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3333
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking

# 独立表空间：每个表都有一个.frm表描述文件和一个.ibd文件
innodb_file_per_table=on
# 不对连接进行DNS解析（提高解析速度）
skip_name_resolve=on
# 修改SQL_Mode（开发使用traditional，线上使用strict_trans_tables）
#sql_mode='traditional'
sql_mode='strict_trans_tables'
#
```

效果：



PS：从MySQL8开始,可通过 `set persist` 命令将全局变量的修改持久化到(`/var/lib/mysql/mysqld-auto.cnf`)配置文件中

> eg: `set persist sql_mode='strict_trans_tables';`

3.3.MySQL

3.3.1.创建、删除数据库

```
-- 如果存在就删除数据库
drop database if exists safe_db;

-- 创建数据库
create database if not exists safe_db;
```

3.3.2.创建、删除表

```
-- 如果存在就删除表
```

```
drop table if exists safe_db.users;

-- mysql> help create table (低版本的默认值不支持函数)
-- 创建表 create table users(字段名 类型 修饰符,...)
create table if not exists safe_db.users
(
    id         int unsigned auto_increment,             -- 主键，自增长【获取ID: last_insert_id()】
    username   varchar(20) not null,
    password   char(40)    not null,                    -- sha1: 40
    email      varchar(50) not null,
    ucode      char(36)    not null,                    -- uuid
    createtime datetime    not null, -- default now(),
    updatetime datetime    not null, -- default now(),
    datastatus tinyint     not null default 0,          -- 默认值为0
    primary key (id),                                   -- 主键可多列
    unique uq_users_email (email),
    index ix_users_createtime_updatetime (createtime, updatetime) -- 索引，不指定名字默认就是字段名
)
--   表选项
--   engine = 'innodb', -- 引擎
--   character set utf8, -- 字符集
--   collate utf8_general_ci, -- 排序规则
;
```

3.3.3.修改表

```
-- 修改表 mysql> help alter table

-- 3.1.添加一列 alter table tb_name add 列名 数据类型 修饰符 [first | after 列名]
alter table safe_db.users
    add uid bigint not null unique first; -- MSSQL没有[first | after 列名]

-- 在email后面添加手机号码列
-- 手机号不会用来做数学运算，varchar可以模糊查询(eg: like '138%')
-- 牵扯到国家代号时，可能出现+、-、()等字符，eg: +86
alter table safe_db.users
    add tel varchar(20) not null after email;

-- 3.2.删除一列 alter table tb_name drop 字段名
alter table safe_db.users
    drop uid;

-- 3.3.添加索引 alter table tb_name add index [ix_name] (列名,...)
alter table safe_db.users
    add index ix_users_ucode (ucode); -- 不指定名字默认就是字段名
-- add index (ucode, tel); -- 不指定索引名字，默认就是第一个字段名

-- 添加唯一键 alter table tb_name add unique [uq_name] (列名,列名2...)
alter table safe_db.users
    add unique uq_users_tel_ucode (tel, ucode);
-- add unique (tel, ucode);-- 不指定索引名字，默认就是第一个字段名

-- 3.4.删除索引 alter table tb_name drop index ix_name
```

```sql
alter table safe_db.users
    drop index ix_users_ucode;

-- 删除索引（唯一键）alter table tb_name drop index uq_name
alter table safe_db.users
    drop index uq_users_tel_ucode;
-- drop index tel; -- 唯一键的索引名就是第一个列名

-- 3.5.修改字段
-- 1.修改字段名：`alter table tb_name change 旧列名 新列名 类型 类型修饰符`
-- 此时一定要重新指定该列的类型和修饰符
alter table safe_db.users
    change ucode usercode char(36);

-- 2.修改字段类型
alter table safe_db.users
    modify username varchar(25) not null;

-- 3.添加默认值：`alter table tb_name alter 列名 set default df_value`
alter table safe_db.users
    alter password set default '7c4a8d09ca3762af61e59520943dc26494f8941b';
```

```sql
desc safe_db.users;
```

Output | Result 6

9 rows

	Field	Type	Null	Key	Default	Extra
1	id	int(10) unsigned	NO	PRI	<null>	auto_increment
2	username	varchar(25)	NO		<null>	
3	password	char(40)	NO		7c4a8d09ca3762af61e59520943dc26494f8…	
4	email	varchar(50)	NO	UNI	<null>	
5	tel	varchar(20)	NO		<null>	
6	usercode	char(36)	YES		<null>	
7	createtime	datetime	NO	MUL	CURRENT_TIMESTAMP	
8	updatetime	datetime	NO		CURRENT_TIMESTAMP	
9	datastatus	tinyint(4)	NO		0	

```
show create table safe_db.users;
```

Output | Result 16

1 row

	Table	Create Table

1	users	CREATE TABLE `users` (
		  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
		  `username` varchar(25) NOT NULL,
		  `password` char(40) NOT NULL DEFAULT '7c4a8d09ca3762af61e59520943dc26494f8941b',
		  `email` varchar(50) NOT NULL,
		  `tel` varchar(20) NOT NULL,
		  `usercode` char(36) DEFAULT NULL,
		  `createtime` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
		  `updatetime` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
		  `datastatus` tinyint(4) NOT NULL DEFAULT '0',
		  PRIMARY KEY (`id`),
		  UNIQUE KEY `uq_users_email` (`email`),
		  KEY `ix_users_createtime_updatetime` (`createtime`,`updatetime`)
		) ENGINE=InnoDB DEFAULT CHARSET=utf8

3.4.SQLServer

示例服务器：`SQLServer 2014`

3.4.1.创建、删除数据库

```
use master

--存在就删除
if exists(select *
          from sysdatabases
          where Name = N'safe_db')
  begin
    drop database safe_db
  end

--创建数据库（简化版：create database safe_db)
create database safe_db
  on primary --数据库文件，主文件组
  (
    name ='safe_db_Data', --逻辑名
    size =10 mb, --初始大小
    filegrowth =10%, --文件增长
    maxsize =1024 mb, --最大值
    filename =N'D:\Works\SQL\safe_db_data.mdf'--存放路径（包含文件后缀名）
    )
  log on --日记
  (
    name ='safe_db_Log',
    size =5 mb,
    filegrowth =5%,
    filename =N'D:\Works\SQL\safe_db_log.ldf'
    );
```

```
-- 切换数据库
use safe_db;
```

3.4.2.创建、删除表

```
--存在就删除表
if exists(select *
          from sysobjects
          where name = N'users')
  begin
    drop table users
  end

-- safe_db.dbo.users
create table users
(
  id         int identity,                                        -- 主键，自增长
  username   nvarchar(20) not null,
  email      varchar(50)  not null,
  password   char(40)     not null,                               -- sha1
  ucode      char(36)     not null default newid(),               -- guid
  createtime datetime     not null default getdate(),
  updatetime datetime     not null default getdate(),
  datastatus tinyint      not null default 0,                     -- 默认值为0
  primary key (id),                                               -- 主键可多列
  unique (email),
  index ix_users_createtime_updatetime (createtime, updatetime) -- 索引
);
```

3.4.3.修改表

```
-- 3.1.添加一列 alter table tb_name add 列名 数据类型 修饰符
-- 在email后面添加手机号码列
alter table users
  add tel varchar(20) not null;

-- 3.1.1.添加含唯一键的列
-- 先添加列
alter table users
  add uid bigint not null
-- 再添加约束 alter table tb_name add constraint uq_name
alter table users
  add constraint uq_users_uid unique (uid); -- 自定义名称

-- 3.1.2.定义和约束一步走（系统设置名字）
-- alter table users
--   add uid bigint not null unique; -- 默认名称

-- 3.2.含唯一键的列
-- 3.2.1.删除约束 alter table tb_name drop constraint uq_name
if exists(select *
          from sysobjects
```

```
          where name = 'uq_users_uid')
alter table users
  drop constraint uq_users_uid;

-- 3.2.2.删除列 alter table tb_name drop column 字段名
alter table users
  drop column uid;

-- 3.3.修改字段
-- 3.3.1.修改列名：exec sp_rename '表名.旧列名','新列名';
exec sp_rename 'users.ucode', 'usercode';

-- 3.3.2.修改字段类型
alter table users
    alter column username varchar(25) not null;

-- 3.3.3.添加默认值：`alter table tb_name alter 列名 set default df_value`
alter table users
  add default '7c4a8d09ca3762af61e59520943dc26494f8941b' for password;
```

PS：SQL Server默认端口为TCP 1433

3.5.区别

简单列举下上面的区别（欢迎补充）：

1. **MySQL自增长是 `auto_increment` ，MSSQL是 `identity`**
2. **MySQL可以设置无符号 `unsigned` ，MSSQL不可以直接设置无符号整型，需要通过约束之类的来限制**
3. **`alter table` 的时候，MSSQL没有 `[first | after 列名]` ，而且语法差别也挺大**

4.CURD

4.1.MySQL

4.1.1.执行流程

select语句执行流程：

1. `from 表`
2. `[inner|left|right] join 表 on 条件`
3. `where 条件`
   - **对select的结果进行过滤**
4. `group by 字段`
   - **根据指定条件把查询结果进行 `分组` ，以用做 `聚合` 运算**
5. `having 条件`
   - **对分组聚合运算(`group by`)后的结果进行过滤**
6. `order by 字段 [asc|desc]`

    - 根据指定字段对查询结果进行排序（默认升序 `asc` ）
7. `select 字段`
8. `limit [偏移量,]显示数量`
    - 显示多少条数据|分页显示

4.1.2.增删改

```
-- 4.1.插入 help insert
-- 自增长主键和默认值的字段可以不写
insert into safe_db.users(username, password, email, tel, usercode, createtime, updatetime,
datastatus)
values ('test', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'test@qq.com', '18738002038', uuid(),
now(), now(), 1);

-- 批量插入
insert into safe_db.users(username, password, email, tel, usercode, createtime, updatetime,
datastatus)
values('xxx', '7c4a8d09ca3762af61e59520943dc26494f8942b', 'xxx@qq.com', '13738002038', uuid(),
now(), now(), 0),('mmd', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'mmd@qq.com', '13718002038',
uuid(), now(), now(), 1),('小张', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'zhang@qq.com',
'13728002038', uuid(), now(), now(), 1);

-- 4.2.修改 help update
update safe_db.users
set datastatus=99,
    updatetime = now()
where username = 'mmd'; -- 一定要有where条件！开发中一般都是先写where条件再写update

-- 4.3.删除
-- 删除数据（自增长不重置）help delete;
delete
from safe_db.users
where datastatus = 0;

-- 删除全部数据（自增长重置）help truncate;
truncate table safe_db.users;
```

4.1.3.查询

`file_records` 的建表语句我放在附录了，这边只谈语法

> PS：查询相关的帮助文档：`help select`

1.**查询来源url**

```
use safe_db;

-- 查询来源url（去重后）
select distinct url
from file_records;

-- 查询来源url（分组方式）
select url
from file_records
group by url;
```

	url
1	http://360.cn
2	http://baidu.com
3	http://cnblogs.com
4	http://qq.com

2.**统计url出现的次数**

```
-- 分别统计一下url出现的次数（分组+聚合）
-- 分组一般都和聚合函数一起使用
select url, count(*) as count
from file_records
group by url;

-- 分别统计一下url出现的次数大于1次的url（删除的不统计）
select url, count(*) as count
from file_records
where datastatus = 1 -- 99代表删除的数据
group by url
having count > 1; -- 在group by的结果上筛选
```

	url	count
1	http://360.cn	1
2	http://baidu.com	2
3	http://cnblogs.com	1
4	http://qq.com	2

	url	count
1	http://qq.com	2

3.**统计一下url出现的次数并查出对应的id**

```
-- 分别统计一下url出现的次数并查出对应的id
select group_concat(id) as ids, url
from file_records
group by url;
```

	ids	url
1	3	http://360.cn
2	1,6	http://baidu.com
3	4	http://cnblogs.com
4	2,5	http://qq.com

4.查询上传文件的用户详细信息，并按文件名降序排序

```
-- 内连接查询 innet join tb_name on 关联条件
select file_records.id,
       users.id                    as uid,
       users.username,
       users.email,
       file_records.file_name,
       file_records.md5,
       inet_ntoa(file_records.ip) as ip,
       file_records.url
from users
        inner join file_records on file_records.user_id = users.id -- 连接条件
where users.datastatus = 1
  and file_records.datastatus = 1
order by file_records.file_name desc; -- 文件名降序排序
```

	id	uid	username	email	file_name	md5	ip	url
1	4	4	小张	zhang@qq.com	9.png	7afbb1602613ec52b265d7a54ad27330	103.3.152.3	http://cnblogs.com
2	1	1	test	test@qq.com	2.zip	3aa2db9c1c058f25ba577518b018ed5b	43.226.128.3	http://baidu.com

5.查询前五条文件上传信息

```
-- MySQL没有`select top n`语法，可以使用 limit来实现，eg: top 5
select *
from file_records
limit 5; -- limit 0,5
```

id	file_name	md5	meta_type	user_id	ip	url	createtime	datastatus
1	2.zip	3aa2db9c1c058f25ba577518b018ed5b	2	1	736264195	http://baidu.com	2019-08-09 11:43:47	1
2	3.rar	6f401841afd127018dad402d17542b2c	3	3	736103427	http://qq.com	2019-08-09 11:43:47	1
3	7.jpg	fe5df232cafa4c4e0f1a0294418e5660	4	5	978522371	http://360.cn	2019-08-09 11:43:47	1
4	9.png	7afbb1602613ec52b265d7a54ad27330	5	4	1728288771	http://cnblogs.com	2019-08-09 11:43:47	1
5	1.gif	b5e9b4f86ce43ca65bd79c894c4a924c	6	3	1914437635	http://qq.com	2019-08-09 11:43:47	1

6.分页查询相关

```
-- 分页查询
-- page:1,count=5 ==> 0,5 ==> (1-1)*5,5
-- page:2,count=5 ==> 5,5 ==> (2-1)*5,5
-- page:3,count=5 ==> 10,5 ==> (3-1)*5,5
-- 推理: limit (page-1)*count,count
select file_records.id,
       users.id                    as uid,
       users.username,
       users.email,
       file_records.file_name,
       file_records.md5,
       inet_ntoa(file_records.ip) as ip,
       file_records.url
from file_records
```

```
        inner join users on file_records.user_id = users.id
limit 0,5;

-- limit后面跟表达式就会报错
select file_records.id,
       users.id                   as uid,
       users.username,
       users.email,
       file_records.file_name,
       file_records.md5,
       inet_ntoa(file_records.ip) as ip,
       file_records.url
from file_records
        inner join users on file_records.user_id = users.id
limit 5,5;
-- limit (2-1)*5,5; -- limit错误写法

-- limit要放在最后
select file_records.id,
       users.id                   as uid,
       users.username,
       users.email,
       file_records.file_name,
       file_records.md5,
       inet_ntoa(file_records.ip) as ip,
       file_records.url
from file_records
        inner join users on file_records.user_id = users.id
order by username desc, file_name desc
limit 10,5; -- 先order by排完序，然后再取第三页的5个数据
```

	id	uid	username	email	file_name	md5	ip	url
1	1	1	test	test@qq.com	2.zip	3aa2db9c1c058f25ba577518b018ed5b	43.226.128.3	http://baidu.com
2	2	3	mmd	mmd@qq.com	3.rar	6f401841afd127018dad402d17542b2c	43.224.12.3	http://qq.com
3	4	4	小张	zhang@qq.com	9.png	7afbb1602613ec52b265d7a54ad27330	103.3.152.3	http://cnblogs.com
4	5	3	mmd	mmd@qq.com	1.gif	b5e9b4f86ce43ca65bd79c894c4a924c	114.28.0.3	http://qq.com
5	6	4	小张	zhang@qq.com	大马.jsp	abbed9dcc76a02f08539b4d852bd26ba	220.181.108.178	http://baidu.com

7.连接查询

```
-- 查找一下从来没上传过文件的用户
-- right join: 以右边表 (users) 为基准连接
select file_records.id             as fid,
       users.id                    as uid,
       users.username,
       users.email,
       file_records.file_name,
       file_records.md5,
       inet_ntoa(file_records.ip) as ip,
       file_records.url
from file_records
        right join users on file_records.user_id = users.id
where users.datastatus = 1
  and file_records.id is null
order by username desc, file_name desc;
```

```
-- 自连接案例:
-- 二级联动 p: province, c: city, a: area
-- 前端一般都会显示省级信息，用户选择后可以获得对应的二三级信息
select c.name, a.name
from city_infos as c
        inner join city_infos as a on a.pcode = c.code
where c.pcode = '320000'; -- pcode设置为索引

-- 通过省名称查询
select p.name, c.name, a.name
from city_infos as c
        inner join city_infos as p on c.pcode = p.code
        inner join city_infos as a on a.pcode = c.code
where p.name = '江苏省';
```

4.1.4.视图

```
-- 简单提一下视图:
-- 创建视图
create view view_userinfo as
select id, username, password, email, tel, datastatus
from safe_db.users;

-- 查询视图
select id, username, password, email, tel, datastatus
from safe_db.view_userinfo;

-- 删除视图
drop view if exists view_userinfo;
```

id	username	password	email	tel	datastatus
1	test	7c4a8d09ca3762af61e59520943dc26494f8941b	test@qq.com	18738002038	1
3	mmd	7c4a8d09ca3762af61e59520943dc26494f8941b	mmd@qq.com	13718002038	99
4	小张	7c4a8d09ca3762af61e59520943dc26494f8941b	zhang@qq.com	13728002038	1

附录

知识点:

```
-- 把ip转换成int
select inet_aton('43.226.128.3'); -- inet6_aton()
-- 把int转换成ip
select inet_ntoa('736264195'); -- inet6_ntoa() ipv6

-- 将多个字符串连接成一个字符串
select concat(user_id, ',', file_name, ',', ip, ',', url) as concat_str
from file_records;

-- 将多个字符串连接成一个字符串+可以一次性指定分隔符
select concat_ws(',', user_id, file_name, ip, url) as concat_str
from file_records;

-- 在有group by的查询语句中，select指定的字段要么就包含在group by语句的后面，作为分组的依据，要么就包含在聚合函数中
-- group_concat(): 将group by产生的同一个分组中的值连接起来，返回一个字符串结果
```

```
select group_concat(file_name) as file_name, url, count(*)
from file_records
group by url;

-- having一般对group by的结果进行筛选，where是对原表进行筛选
select group_concat(file_name) as file_name, group_concat(url) as url, count(*) as count
from file_records
group by url
having count >= 3;

-- 四舍五入到指定位数
select round(3.12345, 4);
-- 存小数数据为了不损伤精读一般都是转成整数，eg：3.1415 ==> 整数：31415，倍数：10000
```

数据构造：

```
-- 编号，文件名，文件MD5，Meta(媒体类型)，当前用户，请求IP，来源地址，请求时间，数据状态
drop table if exists safe_db.file_records;
create table if not exists safe_db.file_records
(
    id         int unsigned auto_increment primary key,
    file_name  varchar(100)     not null,
    md5        char(32)         not null,
    meta_type  tinyint unsigned not null default 1,
    user_id    int unsigned     not null,
    ip         int unsigned     not null,
    url        varchar(200)     not null default '/',
    createtime datetime         not null, -- default now(),
    datastatus tinyint          not null default 0
);

-- 可以插入2~3次（方便下面演示）
insert into safe_db.file_records(file_name, md5, meta_type, user_id, ip, url, createtime,
datastatus)
values ('2.zip', '3aa2db9c1c058f25ba577518b018ed5b', 2, 1, inet_aton('43.226.128.3'),
'http://baidu.com', now(), 1),
       ('3.rar', '6f401841afd127018dad402d17542b2c', 3, 3, inet_aton('43.224.12.3'),
'http://qq.com', now(), 1),
       ('7.jpg', 'fe5df232cafa4c4e0f1a0294418e5660', 4, 5, inet_aton('58.83.17.3'),
'http://360.cn', now(), 1),
       ('9.png', '7afbb1602613ec52b265d7a54ad27330', 5, 4, inet_aton('103.3.152.3'),
'http://cnblogs.com', now(), 1),
       ('1.gif', 'b5e9b4f86ce43ca65bd79c894c4a924c', 6, 3, inet_aton('114.28.0.3'),
'http://qq.com', now(), 1),
       ('大马.jsp', 'abbed9dcc76a02f08539b4d852bd26ba', 9, 4, inet_aton('220.181.108.178'),
'http://baidu.com', now(),
        99);
```

4.2.SQLServer

4.2.1.执行流程

select语句执行流程：

1. `from 表`
2. `join类型 join 表 on 条件`
3. `where 条件`
    - **对select的结果进行过滤**
4. `group by 字段`
    - **根据指定条件把查询结果进行 `分组` ，以用做 `聚合` 运算**
5. `having 条件`
    - **对分组聚合运算(`group by`)后的结果进行过滤**
6. `select distinct 字段`
7. `order by 字段 [asc|desc]`
    - 根据指定字段对查询结果进行排序（默认升序 `asc` ）
8. `top 多少行`
    - 类比 `limit`

4.2.2.增删改

```
-- 4.1.插入 help insert
-- 自增长主键和默认值的字段可以不写
insert into safe_db.dbo.users(username, password, email, tel, usercode, createtime, updatetime,
datastatus)
values ('mmd', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'mmd@qq.com', '18738002038', newid(),
getdate(), getdate(),
        1);

-- 批量插入 SQLServer一次批量插入最多1000行左右
insert into safe_db.dbo.users(username, password, email, tel, usercode, createtime, updatetime,
datastatus)
values ('xxx', '7c4a8d09ca3762af61e59520943dc26494f8942b', 'xxx@qq.com', '13738002038', newid(),
getdate(), getdate(), 0),
       ('mmd', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'mmd@qq.com', '13738002038', newid(),
getdate(), getdate(), 1),
       ('小明', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'xiaoming@qq.com', '13718002038',
newid(), getdate(), getdate(), 1),
       ('小张', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'zhang@qq.com', '13728002038', newid(),
getdate(), getdate(), 1),
       ('小潘', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'pan@qq.com', '13748002038', newid(),
getdate(), getdate(), 1),
       ('小周', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'zhou@qq.com', '13758002038', newid(),
getdate(), getdate(), 1),
       ('小罗', '7c4a8d09ca3762af61e59520943dc26494f8941b', 'luo@qq.com', '13768002038', newid(),
getdate(), getdate(), 1);

-- 4.2.修改 help update
update safe_db.dbo.users
set datastatus=99,
    updatetime = getdate()
where username = 'mmd'; -- 一定要有where条件！开发中一般都是先写where条件再写update

-- 4.3.删除
```

```
-- 删除数据（自增长不重置）help delete;
delete
from safe_db.dbo.users
where datastatus = 0;

-- 删除全部数据（自增长重置）help truncate;
truncate table safe_db.dbo.users;
```

4.2.3.查询

```
-- 查询来源url（去重后）
select distinct url
from file_records;

-- 查询来源url（分组方式）
select url
from file_records
group by url;

-- 分别统计一下url出现的次数（分组+聚合）
-- 分组一般都和聚合函数一起使用
select url, count(*) as count
from file_records
group by url;

-- 分别统计一下url出现的次数，已经删除的文件不算进去
select url, count(*) as count
from file_records
group by url
having count(*) > 3; -- 在group by的结果上筛选，★写成count就不行了★

-- 分别统计一下url出现的次数并查出对应的id
-- SQLServer2017新增string_agg
select ids =(select stuff((select ',' + cast(id as varchar(20)) from file_records as f
 where f.url = file_records.url for xml path ('')), 1, 1, '')),url from file_records
group by url;

-- 内连接查询 innet join tb_name on 关联条件
select file_records.id,
       users.id                   as uid,
       users.username,
       users.email,
       file_records.file_name,
       file_records.md5,
       file_records.ip,
       file_records.url
from users
         inner join file_records on file_records.user_id = users.id -- 连接条件
where users.datastatus = 1
  and file_records.datastatus = 1
order by file_records.file_name desc; -- 文件名降序排序

-- 显示前5个数据
```

```
select top 5 * from file_records;

-- 分页查询 第3页，每页5条
select *
from (select row_number() over (order by username desc, file_name desc) as id,
             file_records.id                                             as fid,
             users.id                                                    as uid,
             users.username,
             users.email,
             file_records.file_name,
             file_records.md5,
             file_records.ip,
             file_records.url
      from file_records
               inner join users on file_records.user_id = users.id) as temp
where id > (3 - 1) * 5 and id <= 3 * 5;

-- 简单提一下视图:
-- 存在就删除
if exists(select *
          from sysobjects
          where name = N'view_userinfo')
    begin
        drop view view_userinfo
    end
-- 创建视图
create view view_userinfo as
select id, username, password, email, tel, datastatus
from users;

-- 查询视图
select id, username, password, email, tel, datastatus
from view_userinfo;
```

附录2

知识点:

```
select getdate() as datatime, newid() as uuid;

-- 类似于concat的效果
select cast(id as varchar(20)) + ','
from file_records for xml path ('');

-- 移除多余的字符
-- STUFF (<character_expression>, <开始>, <长度>, <character_expression>)
-- 将字符串插入到另一个字符串中。它会删除开始位置第一个字符串中的指定长度的字符，然后将第二个字符串插入到开始位置的第一个字符串中
select stuff((select ',' + cast(id as varchar(20))
              from file_records for xml path ('')), 1, 1, '');
```

数据构造:

```
--存在就删除表
if exists(select *
          from sysobjects
          where name = N'file_records')
    begin
        drop table file_records
    end
-- 因为SQLServer的int没有unsigned，所以推荐使用bigint
create table file_records
(
    id         bigint identity (1,1) primary key,
    file_name  varchar(100) not null,
    md5        char(32)     not null,
    meta_type  tinyint      not null default 1,
    user_id    int          not null,
    ip         bigint       not null, -- 在程序中自行转换
    url        varchar(200) not null default '/',
    createtime datetime     not null default getdate(),
    datastatus tinyint      not null default 0
);

-- 可以插入3次（方便下面演示）
insert into file_records(file_name, md5, meta_type, user_id, ip, url, createtime, datastatus)
values ('2.zip', '3aa2db9c1c058f25ba577518b018ed5b', 2, 1, 736264195, 'http://baidu.com',
getdate(), 1),
       ('3.rar', '6f401841afd127018dad402d17542b2c', 3, 3, 736103427, 'http://qq.com', getdate(),
1),
       ('7.jpg', 'fe5df232cafa4c4e0f1a0294418e5660', 4, 5, 978522371, 'http://360.cn', getdate(),
1),
       ('9.png', '7afbb1602613ec52b265d7a54ad27330', 5, 4, 1728288771, 'http://cnblogs.com',
getdate(), 1),
       ('1.gif', 'b5e9b4f86ce43ca65bd79c894c4a924c', 6, 3, 1914437635, 'http://qq.com', getdate(),
1),
       ('大马.jsp', 'abbed9dcc76a02f08539b4d852bd26ba', 9, 4, 3702877362, 'http://baidu.com',
getdate(), 99);
```

5.扩展

5.1.MySQL

聊聊数据库~1.开篇（主要是NoSQL）

> https://www.cnblogs.com/dotnetcrazy/p/9690466.html

聊聊数据库~2.SQL环境篇：

> https://www.cnblogs.com/dotnetcrazy/p/9887708.html

聊聊数据库~3.SQL基础篇：

> https://www.cnblogs.com/dotnetcrazy/p/10399838.html

聊聊数据库~4.SQL优化篇：

> https://www.cnblogs.com/dotnetcrazy/p/10416523.html

聊聊数据库~5.SQL运维上篇：

> https://www.cnblogs.com/dotnetcrazy/p/10810798.html

聊聊数据库~6.SQL运维中篇：

> https://www.cnblogs.com/dotnetcrazy/p/11029323.html

other

CentOS7安装MySQL8.0小计：

> https://www.cnblogs.com/dotnetcrazy/p/10871352.html

MySQL的SQL_Mode修改小计： https://www.cnblogs.com/dotnetcrazy/p/10374091.html

MySQL5.7.26 忘记Root密码小计：

> https://www.cnblogs.com/dotnetcrazy/p/11027732.html

小计：协同办公衍生出的需求：

> https://www.cnblogs.com/dotnetcrazy/p/10481483.html

IDE相关：**JetBrains全家桶破解思路**（最新更新：2019-08-01）：

> https://www.cnblogs.com/dotnetcrazy/p/9711763.html

5.3.SQLServer

SQLServer性能优化之----强大的文件组----分盘存储（水平分库）

> https://www.cnblogs.com/dunitian/p/5276431.html

SQLServer性能优化之---水平分库扩展：

> https://www.cnblogs.com/dunitian/p/6078512.html

SQLServer性能优化之---分表分库技术（同义词+链接服务器）

> https://www.cnblogs.com/dunitian/p/6041745.html

SQLServer性能优化之---数据库级日记监控（XEVENT）

> https://www.cnblogs.com/dotnetcrazy/p/11166516.html

PS：逆天以前整理的**SQLServer脚本**：

> https://github.com/lotapp/BaseCode/tree/master/database/SQL/SQLServer

others

我为NET狂官方面试题-数据库篇答案：

> https://www.cnblogs.com/dunitian/p/6041323.html

牛逼的OSQL----大数据导入：

> https://www.cnblogs.com/dunitian/p/5276449.html

数据库改名系列（数据库名，逻辑名，物理文件名）

> https://www.cnblogs.com/dunitian/p/6165998.html

SQLServer文件收缩-图形化+命令：

> https://www.cnblogs.com/dunitian/p/6047709.html

数据库分离附加（附日记丢失的处理）

> https://www.cnblogs.com/dunitian/p/6165945.html

数据库备份相关：

> https://www.cnblogs.com/dunitian/p/6260481.html

利用SQLServer数据库发送邮件：

> https://www.cnblogs.com/dunitian/p/6022826.html

【恢复挂起的解决方案】附加文件时候的提示“无法重新生成日志，原因是数据库关闭时存在打开的事务/用户，该数据库没有检查点或者该数据库是只读的。”【数据库恢复】

> https://www.cnblogs.com/dunitian/p/6197051.html