数据科学导论
Introduction to Data Science

第二章 数据入门

刘 淇
Email: qiliuql@ustc.edu.cn

课程目标

- 用科学的方法研究和应用数据
  - 文献调研报告1份（一般占30%）
    - 每人一份
    - 时间节点：第9周上课之前（**2019.10.28**）
  - 实践（需要编程）报告1份（一般占30%--40%）
    - 以小组为单位提交，每小组一份，但需要包含每个人的工作介绍
    - 时间节点：第15周上课前（**2019.12.9**）
  - 以上+上课出勤情况即是考核方式

实验题目

- 现提供以下实战题目和若干训练数据集：
  - **BDCI-19比赛题目：** 互联网新闻情感分析
  - **BDCI-19比赛题目：** 离散制造过程中典型工件的质量符合率预测
  - **BDCI-19比赛题目：** 乘用车细分市场销量预测
  - **BDCI-19比赛题目：** 互联网金融新实体发现

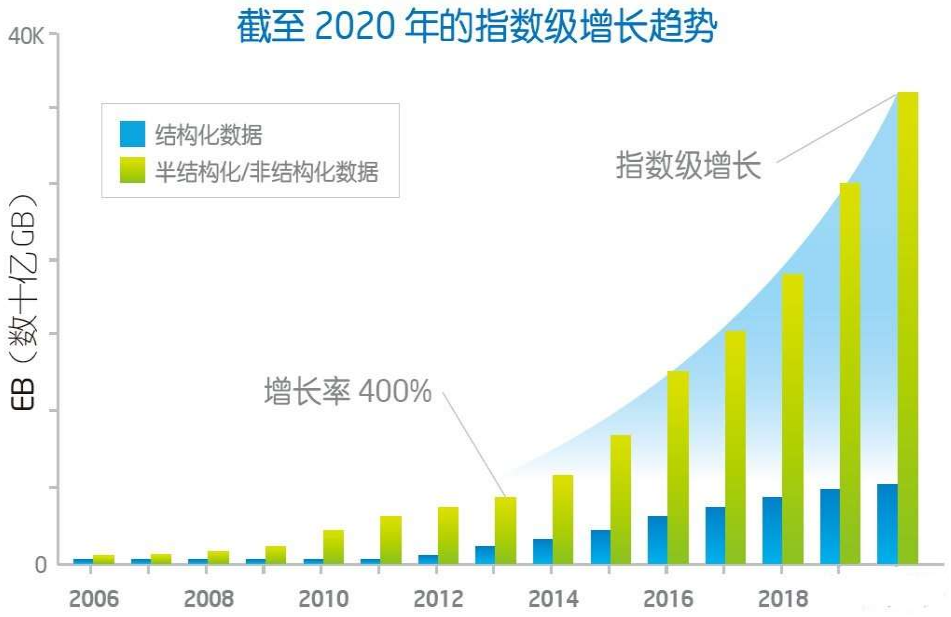
  - **推荐的训练数据集（不是本课程考核的内容）：**
    - UCI数据集：社区犯罪率预测
    - UCI数据集：森林覆盖类型预测
    - UCI数据集：个人收入预测
    - CVPR-17公开数据集：面向图像情感识别

现在任务：

9月20日前完成实验组队，并把组队信息发给助教

数据（Data）入门

- 数据采集　　Data Acquisition
- 数据预处理　　Data Preprocessing
- 特征工程　　Feature engineering



数据采集

- 获得数据的方式多种多样

网页

测量

数据库

监控

传统媒体

数据采集

- 数据检索
- 批量数据获取
  - 网络爬虫
- 数据筛选

数据采集：数据检索

- 最简单、最灵活的数据获取方式就是依靠检索
- 学会使用搜索引擎
  - 百度：适合于搜索中文信息
  - Google：更适合搜索英文信息

数据采集：数据检索

- 最简单、最灵活的数据获取方式就是依靠检索
- 学会使用搜索引擎
  - Google：更适合搜索英文信息
    - 2018.9，Google Dataset Search（Google 数据集搜索）

      网址：https://toolbox.google.com/datasetsearch

目前仍处于测试阶段，支持中文搜索，但中国大陆的用户想要使用依然需要“梯子”

数据采集：公开数据

- 国内常见公开数据渠道
  - 国家相关部门统计信息
  - 中国银行业监督管理委员会
  - 中国国家统计局

- 国际公开数据集
  - 1400万的图像数据
    - http://www.image-net.org/
  - Amazon从2008年开始就为开发者提供几十TB的开发数据
    - http://aws.amazon.com/datasets
  - YouTube视频的统计与社交网络数据
    - http://netsg.cs.sfu.ca/youtubedata/

数据采集：批量数据获取

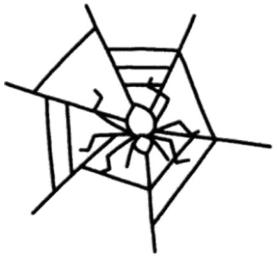
- 大量数据的获取难以手动实现，需借助爬虫程序
  - 也有可能通过交易（购买）“数据”而得
- 网络爬虫是一个自动在网上抓取数据的程序
  - 爬虫本质上就是下载特定网站网页的HTML/JSON/XML数据，并对数据进行解析、提取与存储
  - 通常先定义一组入口**URL**，根据页面中的其他URL，深度优先或广度优先的遍历访问，逐一抓取数据

数据采集：网络爬虫

- 网络爬虫是什么？
  - 网络爬虫（又被称为网页蜘蛛，网络机器人，网页追逐者），是一种按照一定的规则，自动的抓取万维网信息的程序或者脚本。
  - 爬虫的行为可以划分为：载入、解析、存储，且其中最复杂的部分为载入。

网络爬虫：载入

- 载入：将目标网站数据下载到本地
  - 网站数据主要依托于网页（html）展示
  - 爬虫程序向服务器发送网络请求，从而获取相应的网页
    - 网站常用网络协议：http，https
    - 数据常用请求方式：get，post
      - get：参数常放置在URL中
        - 例如：http://www.adc.com?p=1&q=2&r=3
        - 问号后为参数
      - post：参数常放置在一个表单（报文头（header））中
        - 在向目标URL发送请求时，将参数放置在一个网络请求的报文头中



网络爬虫：载入

- 实际操作：抓取一个静态网页步骤
  - 首先确定URL，例如：http://www.baidu.com
  - 其次确定请求的方式以及相关参数：
    - 直接用浏览器实现：chrome,firefox浏览器抓包工具，详见
      - http://jingyan.baidu.com/article/3c343ff703fee20d377963e7.html
    - 或者抓包工具：charles等，详见
      - http://blog.csdn.net/jiangwei0910410003/article/details/41620363/
  - 最后在代码中按照特定的请求方式（get，post）向URL发送参数，即可收到网页的结果

网络爬虫：载入

- 但部分页面的数据是动态加载的
  - Ajax异步请求：网页中的部分数据需要浏览器渲染或者用户的某些点击、下拉的操作触发才能获得
  - 解决方案：
    - 借助抓包工具，分析某次操作所触发的请求，通过代码实现相应的请求
      - 有技术难度，但抓取速度快。
    - 利用智能化的工具：selenium+webdriver
      - 用程序控制浏览器
      - 可以模拟实现人的所有操作
      - 操作简单，但是速度慢
        - 因为爬虫需要启动浏览器，浏览器需要渲染页面，所以速度比较慢

网络爬虫：载入

- 反爬虫：随着网络爬虫对目标网站访问频率的加大，网站禁止爬虫程序继续访问
- 常见反爬手段：
  - 出现用户登录界面，需要验证码
  - 禁止某个固定用户帐号或ip一段时间内访问网站
  - 更有甚者，直接返回错误的无用数据
- 应对措施：
  - 优化爬虫程序，尽量减少访问次数，尽量不抓取重复内容
  - 使用多个cookie（网站用来识别用户的手段，每个用户登录会生成一个cookie）
  - 使用多个ip（可以用代理实现）

网络爬虫：解析

- 解析：在载入的结果中抽取特定的数据，载入的结果主要分成三类html、json、xml
  - html
    - Java工具包：jsoup等
    - Python工具包：beautifulSoup等
  - json
    - Java工具包：json-lib、org-json、jackson等
    - Python工具包：json、demjson等
  - Xml
    - Java工具包：dom4j等
    - Java工具包：xml、libxml2等

网络爬虫：解析(对比JSON与XML)

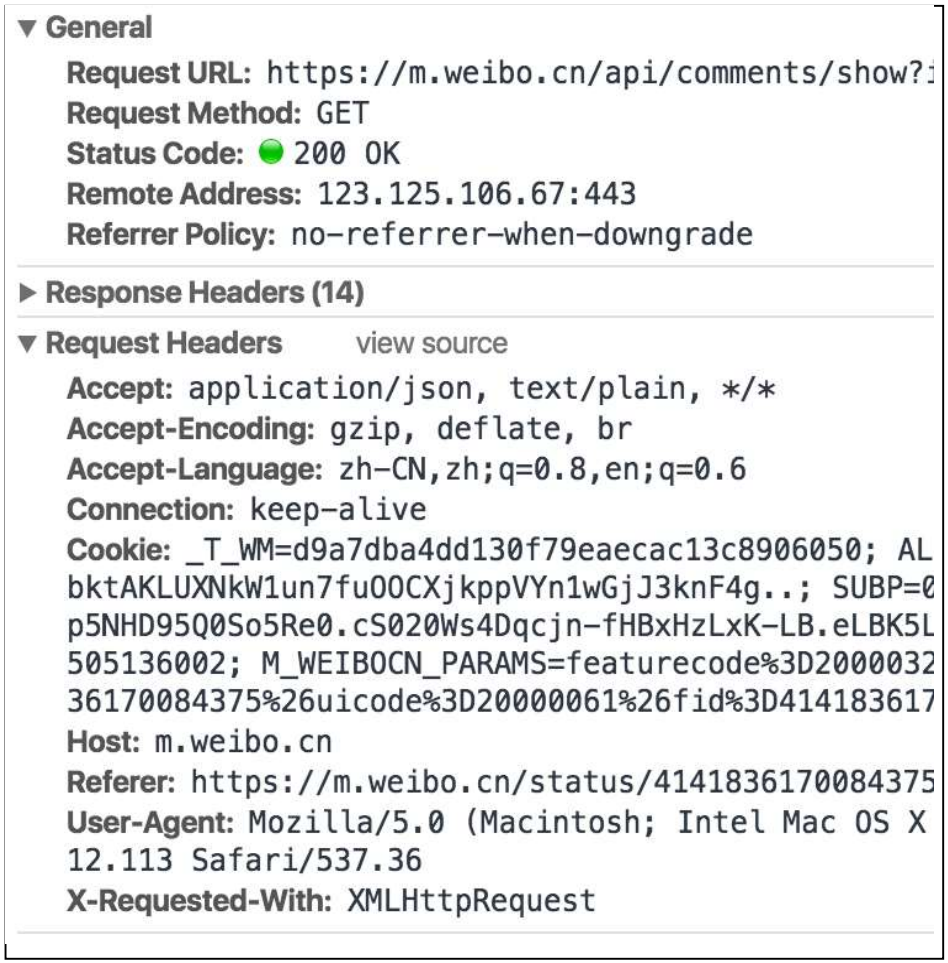

```
{
   "name": "中国",
   "province": [{
      "name": "黑龙江",
      "cities": {
         "city": ["哈尔滨", "大庆"]
      }
   },
     {
      "name": "广东",
      "cities": {
         "city": ["广州", "深圳", "珠海"]
      }
   },
   .....
   }]
```

对象，成员：键值对

```
<?xml version="1.0" encoding="utf-8"?>
<country>
   <name>中国</name>
   <province>
      <name>黑龙江</name>
      <cities>
         <city>哈尔滨</city>
         <city>大庆</city>
      </cities>
   </province>
   <province>
      <name>广东</name>
      <cities>
         <city>广州</city>
         <city>深圳</city>
         <city>珠海</city>
      </cities>
   </province>
  .........
</country>
```

网络爬虫：抓取微博评论

抓包工具
获取请求



网络爬虫：抓取微博评论

获得评论的json格式

京ICP备15025187号-1 邮箱：service@json.cn

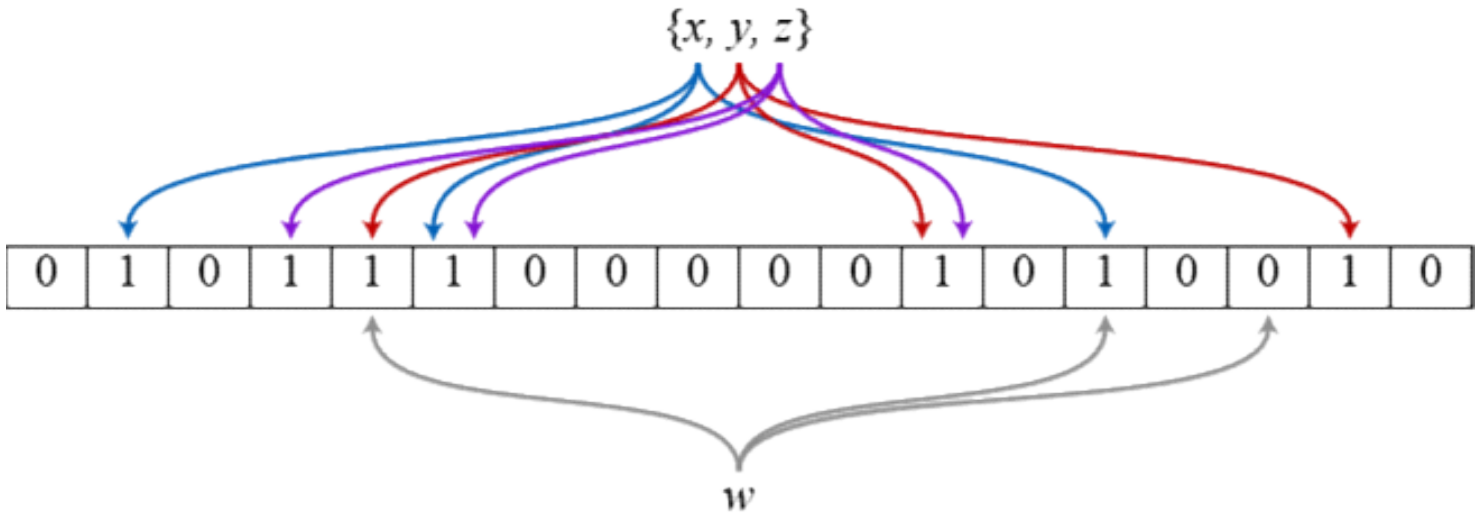
```
{
    "mod_type":"mod/pagelist",
    "previous_cursor":"",
    "next_cursor":"",
    "card_group":[
        {
            "id":4142016554789113,
            "created_at":"08-18 08:46",
            "source":"柔光自拍vivo X7",
            "user":Object{...},
            "text":"回复<a
href="/n/%E9%82%93%E8%B6%85">@邓超</a>:不管是谁,
请大家记住陈赫的话，他们很好，感情都很好。恳请各家粉丝不要
戏太多就好<i class="face face_1 icon_1">[微笑]</i>
没准你们那么嫌弃骂的那么难听，人家正主还是感情好的时不时去
吃火锅呢，你们不累吗？别用自己对他的爱去给他造成困扰",
            "reply_id":4142015488402958,
            "reply_text":"<a
href='/u/'5187664653>@邓超</a>：我也不知道<i
class="face face_1 icon_20">[doge]</i>",
            "like_counts":10811,
            "liked":false,
            "mod_type":"mod/single/infobox"
```


解析出需要的字段

用户id	时间	内容
陈赫	08-18	天霸
邓超	08-18	我们都很好，谢谢大家
邓超	08-18	我也不知道
贼亮zl	08-17	迪丽热巴
……	……	……

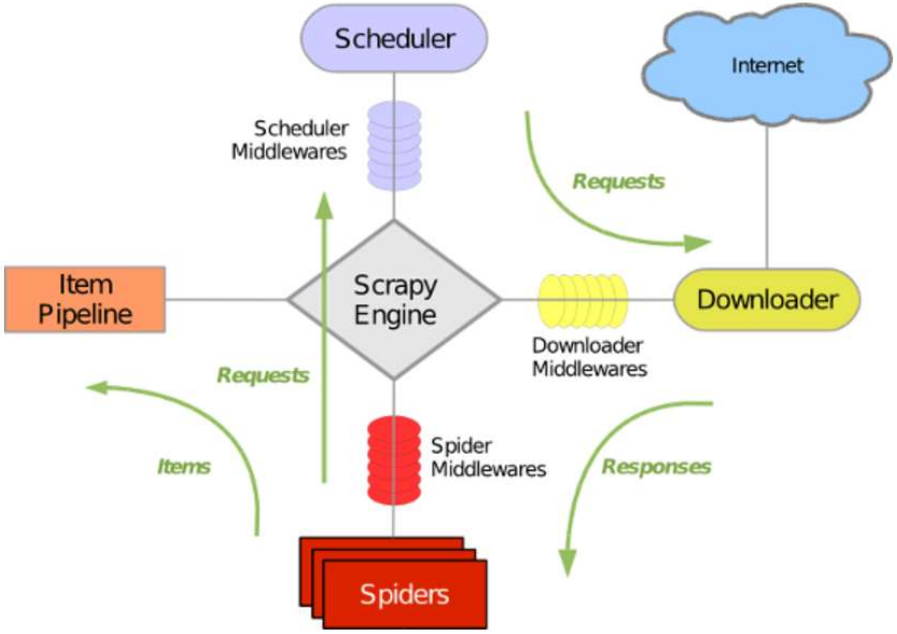
网络爬虫：去重服务

- 去重服务：避免信息的重复抓取，减小存储空间
  - **Bloom过滤器**：由一个很长的二进制向量和一系列随机映射函数组成，通过多个hash函数将一个元素映射成一个位阵列（Bit Array）中的多个点
    - 只有多个hash结果都一样时，才说明数据是重复的



网络爬虫：现有技术

- 基于Java的工具
  - HttpClient
  - Jsoup
- 基于Python的工具
  - **Scrapy**
  - Beautiful-soup



现有的爬虫框架很成熟，能够合理的控制爬取的过程，并有效的处理爬取过程中出现的各种异常，推荐使用Scrapy

网络爬虫：现有技术

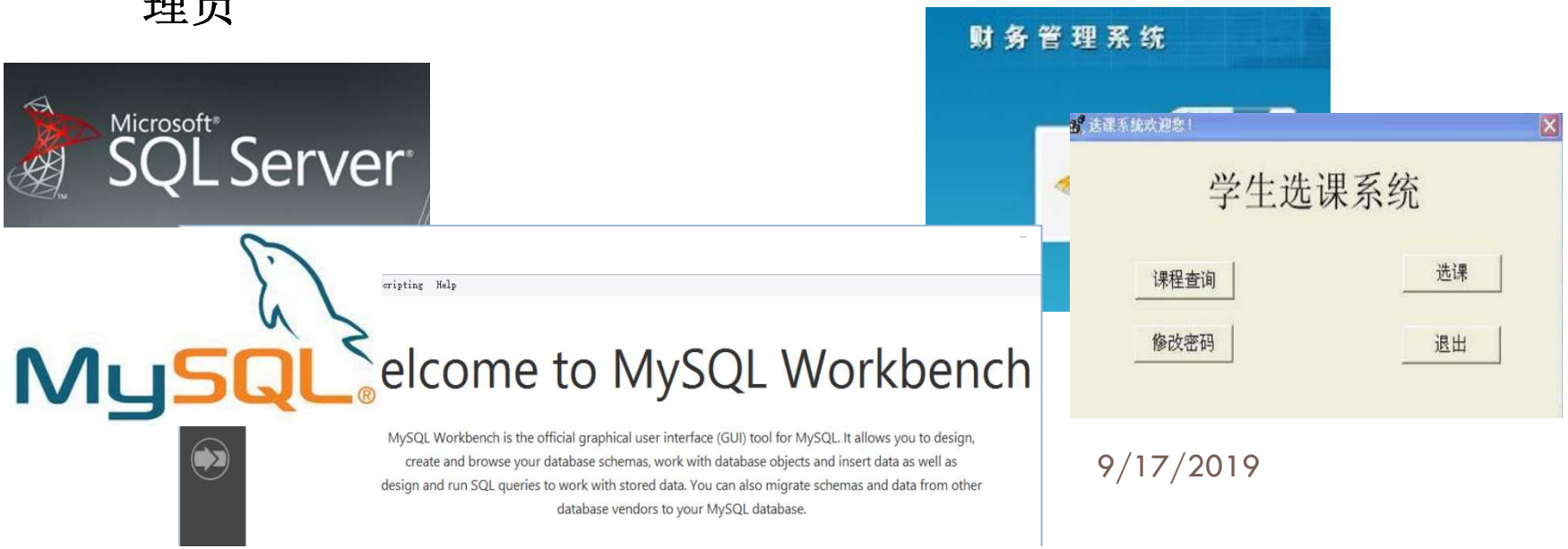
- ItSucks工具
  - 支持通过下载模板和正则表达式来定义下载规则
  - 提供swing GUI操作界面
- Spidernet工具
  - 以递归树为模型的多线程web爬虫程序
  - 存储于sqlite数据文件
- 完整解决方案
  - 基于用户浏览器的爬虫（插件）
  - 八爪鱼
  - 火车采集器

网络爬虫：存储

- 硬盘文件系统（Excel, word, txt…）

网络爬虫：存储

- 数据库(Database,简称DB)是长期储存在计算机内、有组织的、可共享的大量数据的集合。
- 数据库系统（Database System，简称DBS），在计算机系统中引入数据库后的系统构成
  - 数据库、数据库管理系统（及其开发工具）、应用系统、数据库管理员



网络爬虫：存储

- 关系型数据库系统
- 关系必须是规范化的，满足一定的规范条件（范式）

最基本的规范条件（第一范式）：关系的每一个分量必须是一个不可分的数据项，不允许表中还有表

图中工资和扣除是可分的数据项，不符合关系模型要求

职工号	姓名	职 称	应发工 资			扣 除		实发工资
			基 本	津 贴	职务	房 租	水 电	
86051	陈 平	讲 师	1305	1200	50	160	112	2283
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

图 一个工资表(表中有表)实例

网络爬虫：存储

- **SQL（Structured Query Language）结构化查询语言，是关系数据库的标准语言**

［例2］ 将学生张成民的信息插入到Student表中。

```
INSERT
        INTO  Student
        VALUES (‘200215126’,  ‘张成民’,  ‘男’，18，'CS');
```

```
1 • INSERT
2   INTO  Student
3   VALUES ("200215126", "张成民", "男", 18, "CS");
4 • SELECT * FROM student;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Sno	Sname	Ssex	Sage	Sdept
200215126	张成民	男	18	CS
200215128	陈冬	男	18	IS

网络爬虫：存储（冗余）

Student表

学号	所在系	系主任	课程名	成绩
S1	计算机系	张明	C1	95
S2	计算机系	张明	C1	90
S3	计算机系	张明	C1	88
S4	计算机系	张明	C1	70
S5	计算机系	张明	C1	78
….	…	…	…	…

关系模式**Student<U, F>**中存在的问题

1. 数据冗余太大
2. 更新异常（Update Anomalies）
3. 插入异常（Insertion Anomalies）
4. 删除异常（Deletion Anomalies）

网络爬虫：存储

- 数据库的完整性：实体完整性、参照完整性、用户自定义完整性

Student表

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA
200515125	张立	男	19	IS

Course表

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

Student-Course表

学 号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

网络爬虫：存储

- 数据库的安全性：自主存取控制与强制存取控制
  - 自主存取控制（Discretionary Access Control，简称DAC）
    - 用户可“自主”地决定将数据的存取权限授予何人、决定是否也将“授予”的权限授予别人
  - 强制存取控制（Mandatory Access Control，简称 MAC）
    - 系统“强制”地给用户和数据标记安全等级

(1)仅当主体的许可证级别大于或等于客体（数据）的密级时，该主体才能读取相应的客体

(2)仅当主体的许可证级别小于或等于客体（数据）的密级时，该主体才能写相应的客体

网络爬虫：存储

- 数据库的效率：索引
  - 建立索引（Index）的目的：加快查询速度
  - 谁可以建立索引
    - DBA 或 表的属主（即建立表的人）
    - DBMS一般会自动建立以下列上的索引
      PRIMARY KEY
      UNIQUE
  - 谁维护索引
      DBMS自动完成
- 使用索引
    DBMS自动选择是否使用索引以及使用哪些索引

网络爬虫：存储

- 数据库的效率：索引
- **RDBMS中索引一般采用B+树、HASH索引来实现**
  - **B+/B-树索引具有动态平衡的优点**
  - **HASH索引具有查找速度快的特点**

网络爬虫：存储

- B+/B-树索引
- 从二叉树讲起

比较的关键字次数

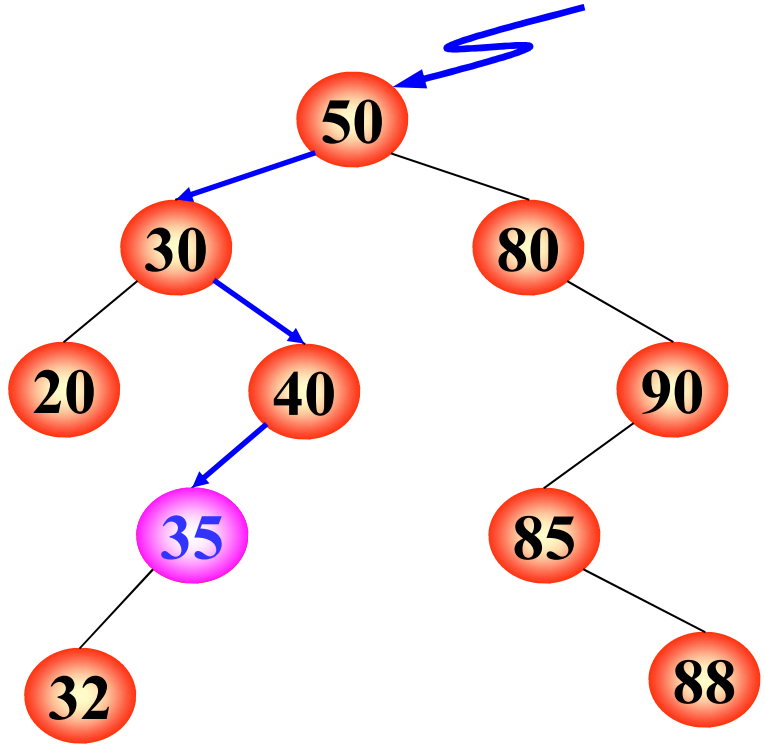
=

此结点所在层次数

最多的比较次数

=

树的深度

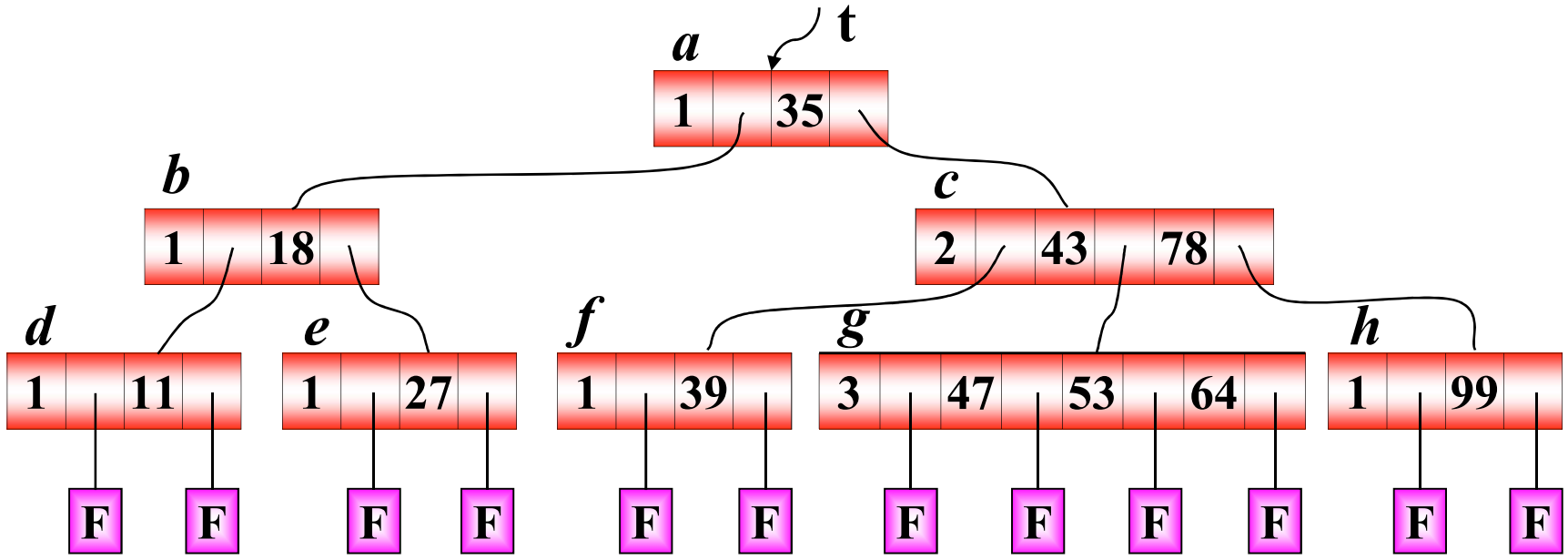


查找关键字：35

网络爬虫：存储

一棵 *m* 阶的 B- 树，或为空树或为满足下列特性的 *m* 叉树：

(5)、所有叶子结点在同一个层次上，且不含有任何信息。（可以看作是外部结点或查找失败的结点，实际上这些结点不存在，指向这些结点的指针为空）。



网络爬虫：存储

- **Hash索引**
- 预先知道所查关键字在表中的位置。即，记录在表中的位置和其关键字之间存在一种确定的关系。

例如：对于如下 9 个关键字：

{Zhao, Qian, Sun, Li, Wu, Chen, Han, Ye, Dai}

设哈希函数 f(key) = ⌊(Ord(关键字首字母) - Ord('A') + 1) / 2⌋

0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Chen	Dai		Han		Li		Qian	Sun		Wu	Ye	Zhao

问题：若添加关键字 Zhou，会出现什么情况？

Zhou

网络爬虫：存储

- NoSQL（Not Only SQL），泛指非关系型的DBMS

分类	Examples举例	典型应用场景	数据模型	优点	缺点
键值（key-value）	Tokyo Cabinet/Tyrant, **Redis,** Voldemort, Oracle BDB	内容缓存，主要用于处理大量数据的高访问负载，也用于一些日志系统等等。	Key 指向 Value 的键值对，通常用hash table来实现	查找速度快，可以通过key快速查询到其value。一般来说，存储不管value的格式，照单全收。	数据无结构化，通常只被当作字符串或者二进制数据
列存储数据库	Cassandra, **HBase,** Riak	分布式的文件系统	以列簇式存储，将同一列数据存在一起	查找速度快，可扩展性强，方便做数据压缩，对针对某一列或者某几列的查询有IO优势。	功能相对局限
文档型数据库	CouchDB, **MongoDb**	Web应用（与Key-Value类似，Value是结构化的，不同的是数据库能够了解Value的内容）	Key-Value对应的键值对，Value为结构化数据	数据结构要求不严格，表结构可变，不需要像关系型数据库一样需要预先定义表结构	查询性能不高，而且缺乏统一的查询语法。
图形(Graph)数据库	**Neo4J,** InfoGrid, Infinite Graph	社交网络，推荐系统等。专注于构建关系图谱	图结构	利用图结构相关算法。比如最短路径寻址，N度关系查找等	经常需对整个图做计算才能得出需要的信息，且这种结构不太好做分

网络爬虫：存储

- NoSQL适用于
  - 数据模型比较简单；
  - 需要灵活性更强的IT系统；
  - 对DBMS性能要求较高；
  - 不需要高度的数据一致性；
  - 对于给定key，比较容易映射复杂值的环境
- 比较
  - NoSQL DBMS的产生是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题
  - NoSQL结构比较简单，逻辑控制相对较少，同等存量下数据量超过关系型DBMS，但是处理能力不一定高
  - 对于数据间有固定模式且紧密联系的，还是建议选择关系型DBMS。

网络爬虫：图像(Image)存储

图像数据一般较大，所以可以考虑以下两种存储方式：

存储图片路径

将图片存在本地，比如windows系统中，在数据库中写入图片的路径，用来索引图片

id	path
1	/image/apple.jpg
2	/image/car.jpg
3	/image/cat.jpg

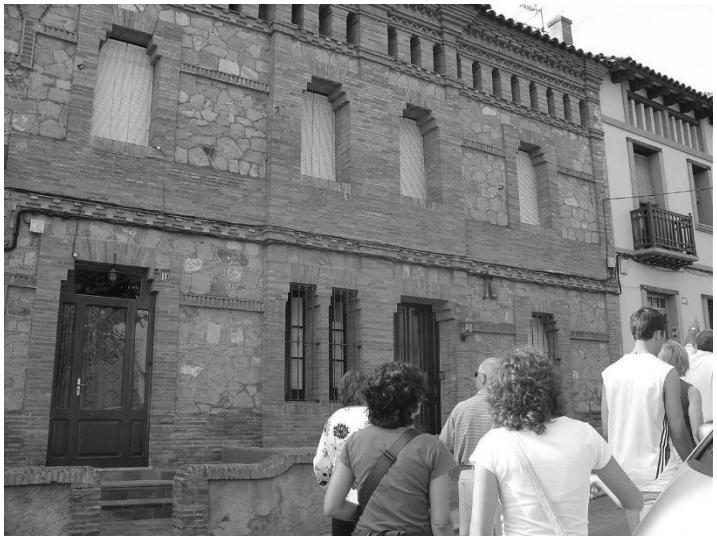
存储图像数据

直接将图像数据存入数据库系统中。可以直接提取图像的像素值，存为numpy, json等格式数据，写入数据库系统中。也可以将其以二进制文件的格式写入。

id	data
1	[[137 124 143 111 62 248 253] [133 116 160 125 133 153 85] [102 122 123 137 142 128 130] [116 52 121 121 56 124 98] [99 116 118 36 127 134 169] [67 119 89 253 158 222 204] [100 54 110 62 202 213 184]]

网络爬虫：图像(Image)存储

height: 960
width: 1280



彩色图

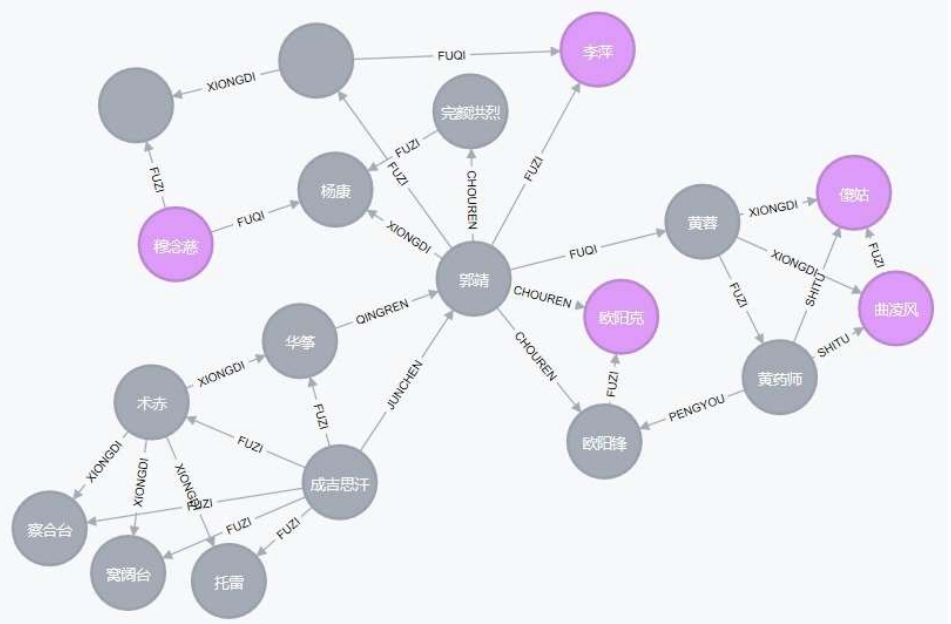
灰度图

当计算机看到一张图像时，它看到的是一堆像素值。
对于左边彩色图，它将看到一个960*1280 *3的数组，3指代的是RGB三个通道；
对于右边灰度图，它将看到一个960*1280的数组。
其中，每个数字的值从0到255不等，其描述了对应那一点的像素灰度。
所以，计算机对图像做处理时，实际上就是对这些数组中的像素值做处理。

网络爬虫：图形(Graph)数据存储

- 如何表示关系型数据？
- SQL
  - 冗余数据
  - 大量空白

名字	子女	兄弟	父亲	年龄
郭靖	破虏	杨康		33
杨康	\	郭靖	完颜洪烈	
破虏	\	\	郭靖	10



Create(n:person{name:”郭靖”,age:”33” })
Create(m:person{name:”破虏”,age:”10” })
create (n)-[:R{type:"父子"}]->(m)

- Neo4J
  - 符合图数据特性
  - 方便删改

cypher是neo4j官网提供的声明式查询语言

网络爬虫：图形(Graph)数据存储

- 删去郭靖

名字	子女	兄弟	父亲	年龄
郭靖	破虏	杨康		33
杨康	\	郭靖	完颜洪烈	
破虏	\	\	郭靖	10

match(n:person{name:”郭靖”}) delete n

SQL **Neo4j**

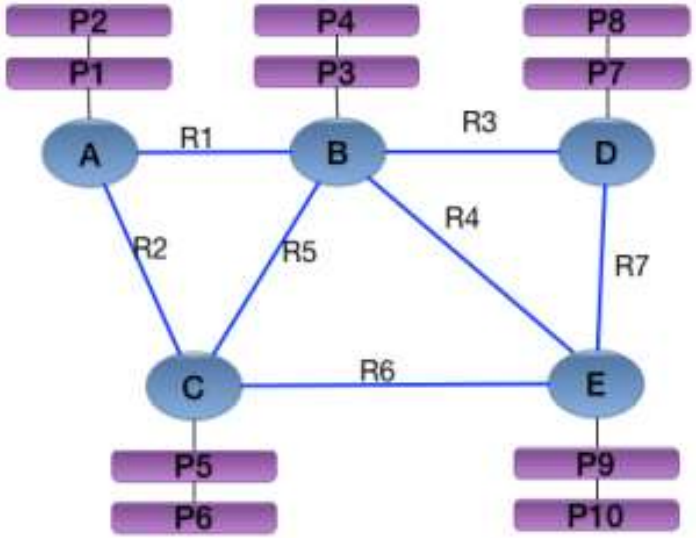
- 杨康和破虏是什么关系？

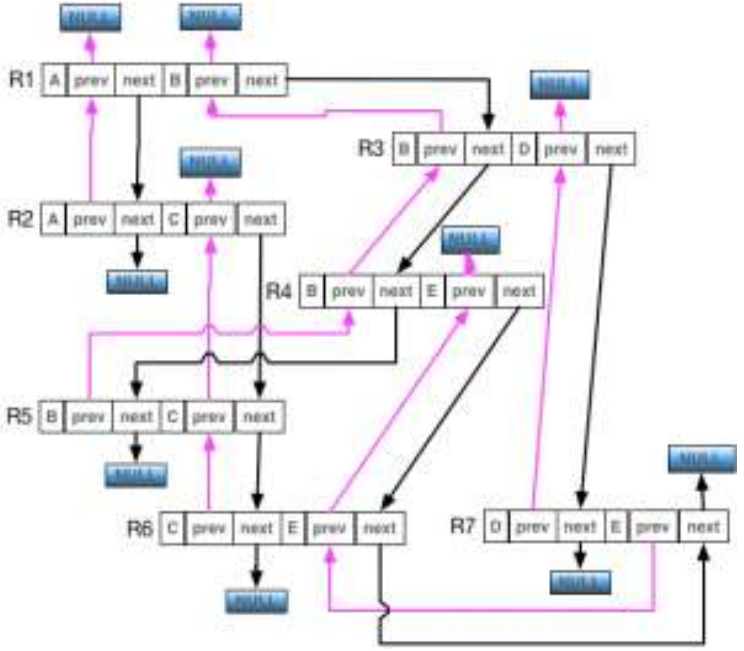
名字	子女	兄弟	父亲	年龄
郭靖	破虏	杨康		33
杨康	\	郭靖	完颜洪烈	
破虏	\	\	郭靖	10

match (n:Person{name:“杨康"}),
(m:Person{name:”破虏”}) ,
r=(n)-[]-(m) ,
return n,m,r

网络爬虫：图形(Graph)数据存储

- 为什么Neo4j适合存储图数据？
  - 使用指针遍历所有节点
  - 符合节点+关系的图结构





通过指针遍历所有节点以及关系

网络爬虫：文档数据存储-MongoDB

- MongoDB简介
  - 基于分布式文件存储，由 C++ 语言编写，旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。
  - 介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。
  - 将数据存储为一个文档，数据结构由键值(key=>value)对组成，类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组

```
{
  name: "sue",          <-- field: value
  age: 26,              <-- field: value
  status: "A",          <-- field: value
  groups: [ "news", "sports" ]   <-- field: value
}
```

与传统的sql不同，各个字段不需要预先定义好数据类型，所以mongodb很灵活。

MongoDB VS SQL

SQL术语/概念	MongoDB术语/概念	解释/说明
database	database	数据库
table	collection	数据库表/集合
row	document	数据记录行/文档
column	field	数据字段/域
index	index	索引
table joins		表连接，MongoDB不支持
primary key	primary key	主键，MongoDB自动将_id字段设置为主键

MongoDB基本语句

- 创建数据库
  - use DATABASE_NAME
- 查看数据库
  - show dbs
- 插入文档
  - db.COLLECTION_NAME.insert(document)
- 查看文档
  - db. COLLECTION_NAME.find()
- 删除文档
  - db. COLLECTION_NAME.remove(<query>, <justOne>)

只有数据库非空，
才会显示该数据库

document可以是已经
定义好的Bson文档

可以设置一定
的查询条件

只想删除第一条找到的记录
可以设置 justOne 为 1

什么时候使用MongoDB?

- 一般情况
  - Sql能存储的一般都可以用mongodb存储（事务除外）
- 非常适合日志、博客等比较杂乱的系统的存储
  - 种类较多、范围较大、内容也比较杂乱
  - 原因：在collection中，document对字段没有强约束
- 大文件存储
  - 二进制存储，可以用pickle等工具包对其进行压缩
  - GridFS：是MongoDB规范，用于存储和检索图片、音频、视频等大文件，可以存储超过16M的文件

MongoDB存储举例——存储字段


```
> db.student.insert({"_id":"BA18011000", "name":"zhang san",
"sex":"male","age":18,"introduction":"Zhang San is a handsome guy!"})
```

```
db.getCollection('student').find({})
```

显示该数据集合

student 0.001 sec.

	_id	name	sex	age	introduction
1	BA18011000	zhang san	male	18.0	Zhang San is a handsome guy!

```
> db.student.insert({ “_id” : “BA18011001” , “name” : “Li Si", "sex":"female",
"age":17,"introduction":"Li Si is a beautiful girl!", "class":["Math","Music","Physics"]})
```

```
db.getCollection('student').find({})
```

document对字段没有强约束，Value可以是各种类型的文档

	_id	name	sex	age	introduction	class
1	BA18011000	zhang san	male	18.0	Zhang San is a handsome guy!	
2	BA18011001	Li Si	female	17.0	Li Si is a beautiful girl!	[3 elements]

MongoDB存储举例——存储图片


```
file_path = "E:\\USTC\\store_data\\photo"
files = os.listdir(file_path)
# print(files)

pics =[]
#遍历图片目录集合
for index, file in enumerate(files):
    filename = file_path + '\\' + file
    print(filename)
    with open(filename, "rb") as b_image:
        content = b_image.read()
        pics.append(content)
```

二进制读取图片

```
dic = {
    "_id" : "BA18011002",
    "name" : "Wang Wu",
    "sex" : "female",
    "age" : 19.0,
    "introduction" : "Wang Wu studies
very hard!",
    "class" : [
        "Math",
        "Physics",
        "chemistry"
    ],
    "picture": pics
}
student.insert_one(dic)
```

构造图片文档

```
db.getCollection('student').find({})
```

student 0.003 sec.

	_id	name	sex	age	introduction	class	picture
1	BA18011000	zhang san	male	18.0	Zhang San i...		
2	BA18011001	Li Si	female	17.0	Li Si is a be...		
3	BA18011002	Wang Wu	female	19.0	Wang Wu s...	[3 element...	[5 element...

```
"picture" : [
    { "$binary" : "/9j/4AAQSkZJRgABAQEASA
    { "$binary" : "/9j/4AAQSkZJRgABAQEASA
    { "$binary" : "/9j/4AAQSkZJRgABAQEASA
    { "$binary" : "/9j/4AAQSkZJRgABAQEASA
    { "$binary" : "/9j/4AAQSkZJRgABAQEASA
```

MongoDB存储举例


```
db.getCollection('student').find({})
```

student 0.005 sec.

```
/* 1 */
{
    "_id" : "BA18011000",
    "name" : "zhang san",
    "sex" : "male",
    "age" : 18.0,
    "intorduction" : "Zhang San is a handsome guy!"
}

/* 2 */
{
    "_id" : "BA18011001",
    "name" : "Li Si",
    "sex" : "female",
    "age" : 17.0,
    "intorduction" : "Li Si is a beautiful girl!",
    "class" : [
        "Math",
        "Music",
        "Physics"
    ]
}

/* 3 */
{
    "_id" : "BA18011002",
    "name" : "Wang Wu",
    "sex" : "female",
    "age" : 19.0,
    "intorduction" : "Wang Wu studies very hard!",
    "class" : [
        "Math",
        "Physics",
        "chemistry"
    ],
    "picture" : [
        { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4TeGRXhpZgAATU0AKgAAAAgABgALAAIAAAAmAAAIYgESAAMAAAABAAEAAA
        { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4Tg6RXhpZgAATU0AKgAAAAgABgALAAIAAAAmAAAIYgESAAMAAAABAAEAAA
        { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4V70RXhpZgAATU0AKgAAAAgABgALAAIAAAAmAAAIYgESAAMAAAABAAEAAA
        { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4TjeRXhpZgAATU0AKgAAAAgABgALAAIAAAAmAAAIYgESAAMAAAABAAEAAA
        { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4SucRXhpZgAATU0AKgAAAAgABgALAAIAAAAmAAAIYgESAAMAAAABAAEAAA
    ]
}
```

正如前面所说的非关系数据库MongoDB是文档型存储，数据集合中存储的正是这样的三条文档记录，与json非常的类似。

MongoDB VS MySQL

对比角度	MongoDB	MySQL
数据库模型	非关系型	关系型
存储方式	虚拟内存+持久化	根据引擎有不同
查询语句	独特的mongodb查询语句	传统的sql语句
架构特点	通过副本集和分片可实现高可用	单点，M-S，MHA，MMM，Cluster等
数据处理方式	将热数据存储于内存，高速读写	根据引擎有不同
成熟度	新兴，成熟度较低	较为成熟的体系
广泛度	在Nosql中较为完善，使用人群也在不断增长	开源数据库的份额在增加，mysql的份额也在增长
优势	在适量级内存的Mongodb的性能是非常迅速的；高扩展性，存储的数据格式是json格式；非常适合日志、博客等比较杂乱的系统的存储	拥有较为成熟的体系，成熟度很高，支持事务
劣势	不支持事务，而且开发文档不是很完全，完善	在海量数据处理的时候效率会显著变慢

网络爬虫：存储

- 键值数据库：Redis
  - 数据必须存储在某个key下
  - 数据可以是：
    - 整数
    - 字符串
    - 列表
    - 哈希表
    - …
  - 只提供数据的基本操作

Key Value

```
SET counter 10
INCR counter => 11
GET counter => 11

RPUSH names “taylor”
RPUSH names “swift”
LLEN names => 2
LPOP names => “taylor”

HSET user:1000 name "John"
HSET user:1000 password "s3cret”
HGET user:1000 name => "John"
```

- 因为简单，所以快！

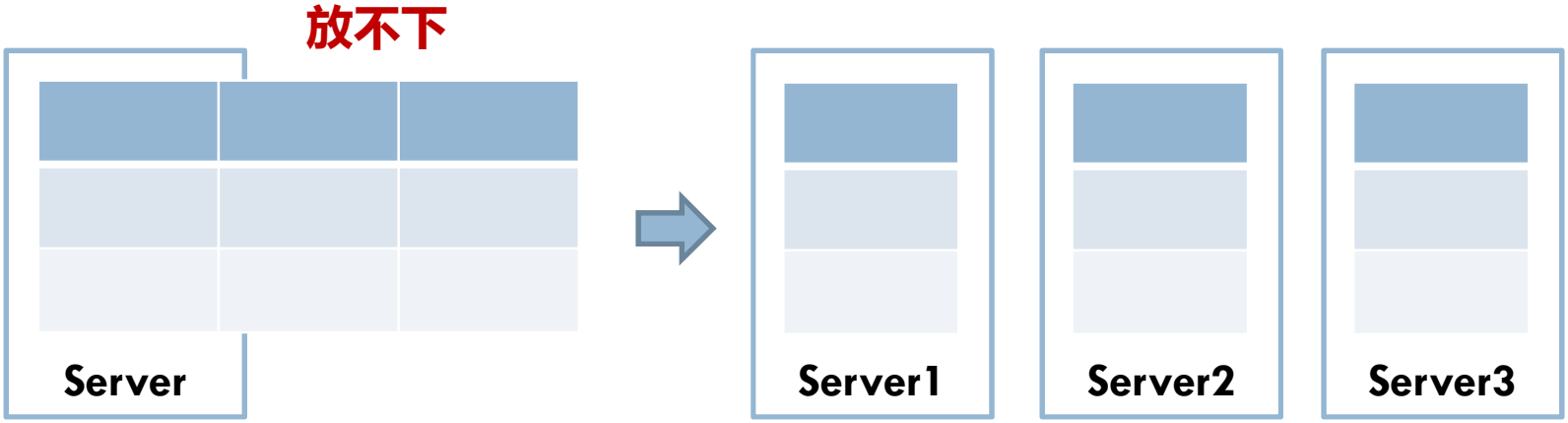
网络爬虫：存储

- Redis (REmote DIctionary Server)
  - key-value存储系统
- 应用场景
  - 电商“秒杀”：
    - 短时间内极大量访问
    - 避免“超抢”、“超卖”

网络爬虫：存储

- 列存储数据库：HBase
  - 分布式的，同一列数据存在一起
  - 应对超大量数据（十亿及百亿行）
  - 将数据表分布式存储
  - 原生仅支持对数据表的简单操作（NoSQL）
  - 可通过扩展模块支持：SQL、图查询、Hadoop、搜索引擎等

只要数据量传统数据库能够解决，就不用HBase，分布式增加性能损耗。

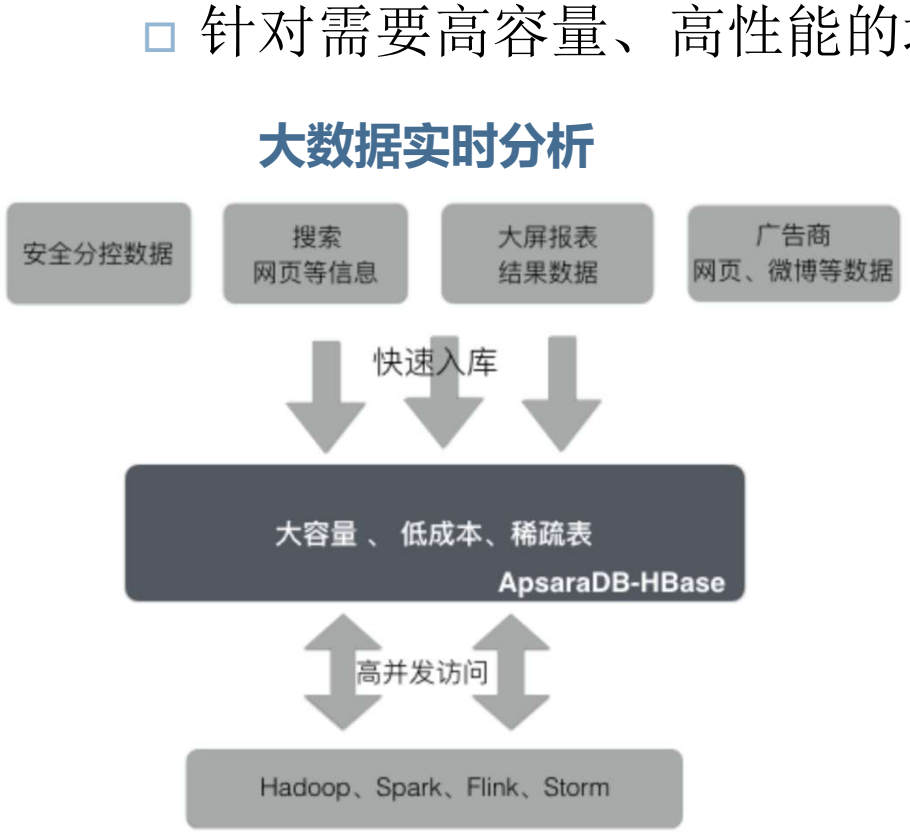


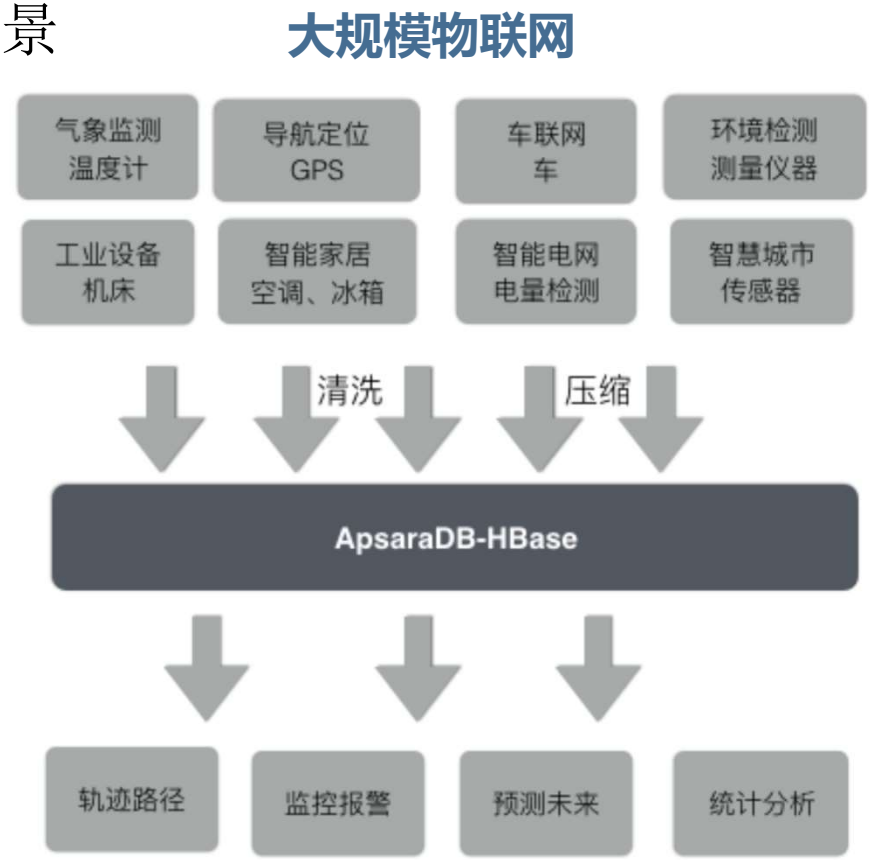
网络爬虫：存储

- HBase应用场景

高容量：支持大规模实时数据快速入库
高性能：支持大数据分析应用（如**hadoop**、**spark**）高速读取、分析

- 针对需要高容量、高性能的场景





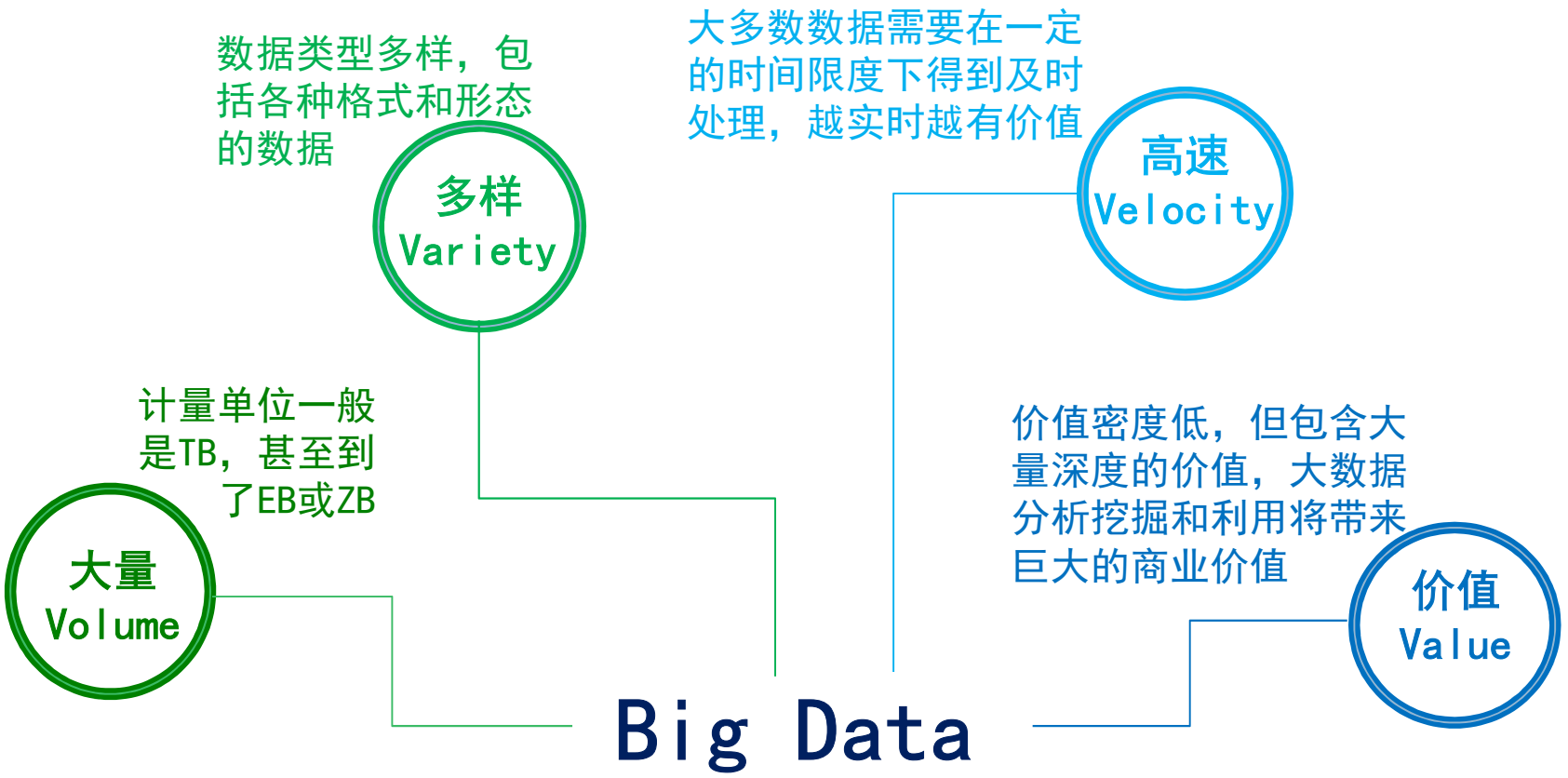
网络爬虫：资料

Beautiful Soup	http://beautifulsoup.readthedocs.io/zh_CN/latest/
MongoDB 教程	http://www.runoob.com/mongodb/mongodb-tutorial.html
CSS选择器教程	http://www.w3school.com.cn/cssref/css_selectors.asp
jsoup教程	http://blog.csdn.net/column/details/jsoup.html
scrapy教程	http://scrapy-chs.readthedocs.io/zh_CN/0.24/intro/tutorial.html

数据预处理

- 大数据环境下的数据特征
- 为什么需要进行预处理
- 预处理的基本方法
  - 数据清洗
  - 数据集成
  - 数据变换
  - 数据规约

大数据环境下的数据特征



大数据环境下的数据特征

- 现实世界中大型数据库，互联网等数据的共同特点
  - 不准确
    - 不正确的属性值
    - 包含错误或偏离期望的离群
    - 大量的模糊信息
  - 不完整
    - 有些数据属性的值丢失或不确定
    - 缺失必须的数据
  - 不一致
    - 原始系统从各个实际应用系统中获得，数据结构有较大差异
    - 不同系统中数据由于合并普遍存在数据重复和信息冗余现象

数据预处理

- 大数据环境下的数据特征
- 为什么需要进行预处理
- 预处理的基本方法
  - 数据清洗
  - 数据集成
  - 数据变换
  - 数据规约

为什么进行数据预处理

- 现实世界的数据是“脏的”——数据多了，什么问题都会出现
  - 不完整
    - 缺少数据值；缺乏某些重要属性
  - 有噪声
    - 包含错误或者孤立点
    - e.g. Salary = -10
  - 数据不一致
    - e.g.， 在编码或者命名上存在差异
    - e.g.， 过去的等级:“1,2,3”， 现在的等级:“A, B, C”
    - e.g.， 重复记录间的不一致性

为什么进行数据预处理

- 现实世界的数据是“脏的”——数据多了，什么问题都会出现
  - 滥用缩写词　　---中科大， 科大，中国科大，USTC
  - 数据输入错误　　---裤子大、国科大…
  - 数据中的内嵌控制信息　---E3=F3*C3
  - 不同的惯用语　　----南七技校
  - 重复记录
  - 丢失值
  - 拼写变化
  - 不同的计量单位
  - 过时的编码
  - 含有各种噪声，如中学生年龄

为什么进行数据预处理

- 数据错误的不可避免性
  - 数据输入和获得过程数据错误的不可避免性
  - 数据集成所表现出来的错误
  - 数据传输过程所引入的错误
  - 据统计有错误的数据占总数据的5%左右

- 其他类型的数据质量问题:
  - Data needs to be integrated from different sources
  - Missing values
  - Noisy and inconsistent values
  - Data is not at the right level of aggregation

为什么进行数据预处理

- 没有高质量的数据，就没有高质量的结果
  - 高质量的决策必须依赖高质量的数据
    - e.g. 重复值或者空缺值将会产生不正确的或者误导人的统计

- 数据质量的含义
  - 正确性（ Correctness）
  - 一致性（ Consistency）
  - 完整性（ Completeness）
  - 可靠性（ Reliability）

- 数据预处理是进行大数据的分析和挖掘的工作中占工作量最大的一个步骤