



本课件仅用于教学使用。未经许可，任何单位、组织和个人不得将课件用于该课程教学之外的用途(包括但不限于盈利等)，也不得上传至可公开访问的网络环境

1

# 数据科学导论

## Introduction to Data Science

### 第二章 数据分析基础

黄振亚, 陈恩红

Email: [cheneh@ustc.edu.cn](mailto:cheneh@ustc.edu.cn), [huangzhy@ustc.edu.cn](mailto:huangzhy@ustc.edu.cn)

课程主页:

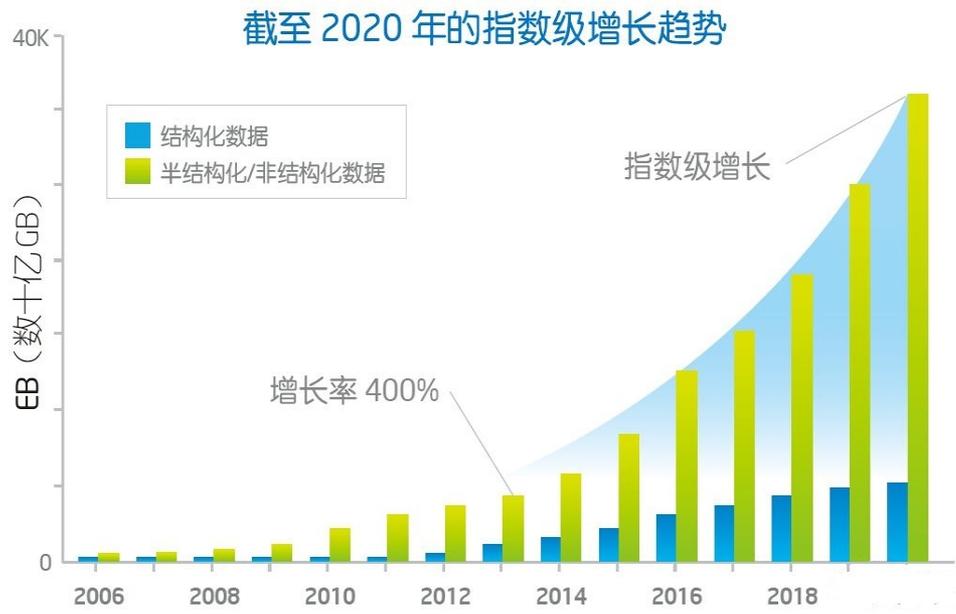
<http://staff.ustc.edu.cn/~huangzhy/Course/DS2024.html>



# 数据分析基础

2

- 数据采集 Data Collection
- 数据存储 Data Storage
- 数据预处理 Data Preprocessing
- 特征工程 Feature engineering





# 数据存储

## 我们面对的数据有哪些？

### MovieLens

\*ratings.csv - 记事本

文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)

userId,movieId,rating,timestamp

1,1,4.0,964982703

1,3,4.0,964981247

1,6,4.0,964982224

1,47,5.0,964983815

1,50,5.0,964982931

1,70,3.0,964982400

1,101,5.0,964980868

### SQuAD

#### Context:

Computational complexity theory is a branch of the theory of computation in theoretical computer science that focuses on classifying computational problems according to their inherent difficulty, and relating those classes to each other. A computational problem is understood to be a task that is in principle amenable to being solved by a computer, which is equivalent to stating that the problem may be solved by mechanical application of mathematical steps, such as an algorithm.

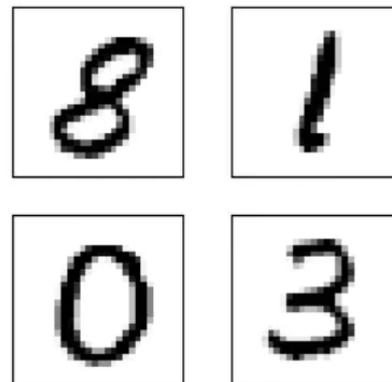
#### Question:

By what main attribute are computational problems classified using computational complexity theory?

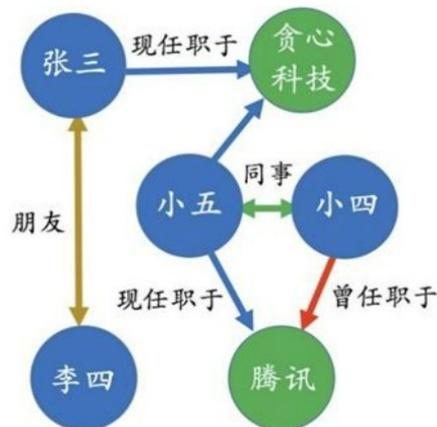
#### Answer:

inherent difficulty

### Image



### Graph





# 数据存储

## □ 结构化数据

- 可以使用关系型数据库表示和存储的数据，拥有固定结构

## □ 半结构化数据

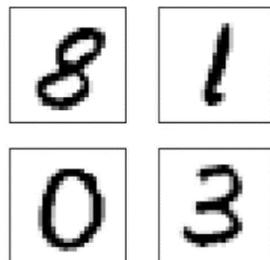
- 弱结构化，虽然不符合关系型数据模型的要求，但是含有相关的标记(自描述结构)，分割实体及期属性。如：XML，JSON等

## □ 非结构化数据

- 没有固定数据结构，或很难发现统一数据结构的数据
- 文档，文本，图片，视频，音频等

ID	时间	内容
陈赫	08-18	天霸
邓超	08-18	我们都很好
邓超	08-18	我也不知道

```
<province>  
  <name>黑龙江</name>  
  <cities>  
    <city>哈尔滨</city>  
    <city>大庆</city>  
  </cities>  
</province>
```





# 数据存储

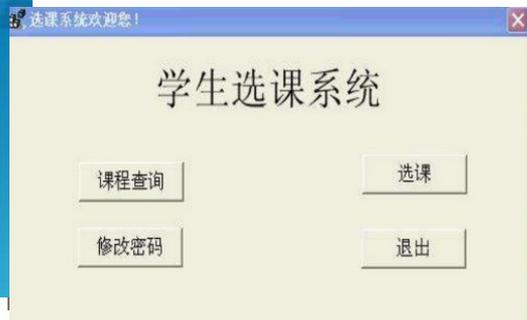
- 人脑，书籍，文件
- 硬盘文件系统（Excel, word, txt...）





# 数据存储：数据库

- 数据(Data)
- 数据库(Database, DB)
  - 长期储存在计算机内、有组织的、可共享的大量数据的集合。
- 数据库管理系统(Database Management System, DBMS)
  - 位于用户与操作系统之间的一层数据管理软件
- 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
  - DB、DBMS（及其开发工具）、应用系统、数据库管理员

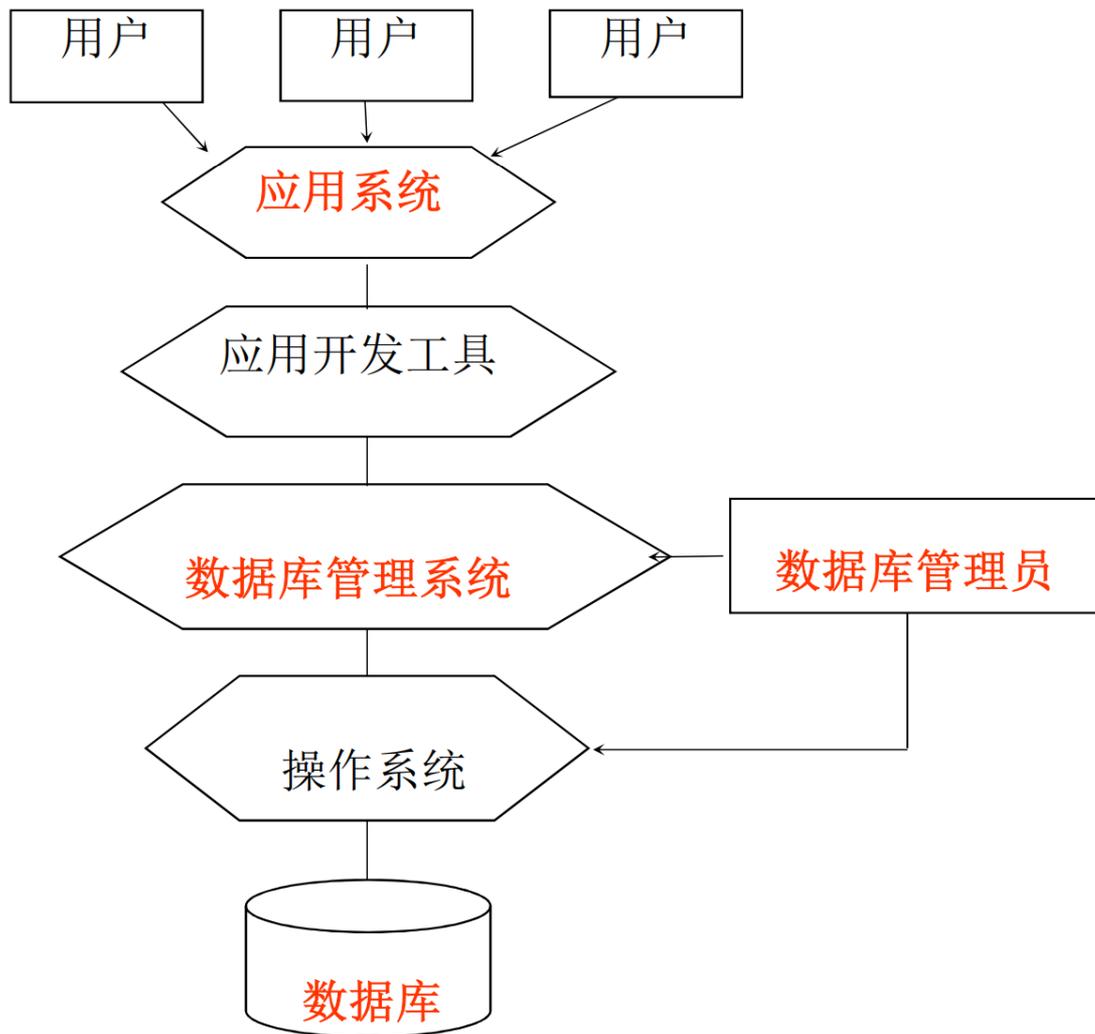




# 数据存储：数据库

7

数据库系统





# 数据存储：数据库

## □ 数据库有哪些





# 我国的数据库

- 数据库有哪些
  - 人大金仓KingBase
  - 达梦数据库
  - 神通数据库
  - 阿里OceanBase, PolarDB
  - 腾讯的TDSQL
  - 华为的GaussDB



PRODUCT DISPLAY  
神通大型通用数据库



神通数据库管理系统



神通数据库(MPP集群)



神通数据库 (openGauss 版)



神通高可用集群软件

# OCEANBASE

用技术让海量数据的管理和使用更简单





# 数据存储：数据库

□ DB-Engines: <https://db-engines.com/en/ranking>

423 systems in ranking, September 2024

Rank	Rank			DBMS	Database Model	Score			
	Sep	Aug	Sep			Sep	Aug	Sep	
119.	↑	128.	↑	146.	<b>OceanBase</b>	Relational, Multi-model ⓘ	1.97	+0.08	-0.23
120.	↑	125.	↑	122.	<b>LMDB</b>	Key-value	1.94	+0.00	-0.86
		3.		3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	807.76	-7.41	-94.45
		4.		4.	PostgreSQL +	Relational, Multi-model ⓘ	644.36	+6.97	+23.61
		5.		5.	MongoDB +	Document, Multi-model ⓘ	410.24	-10.74	-29.18
		6.		6.	Redis +	Key-value, Multi-model ⓘ	149.43	-3.28	-14.26
192.	↑	196.	↑	210.	<b>TDSQL for MySQL +</b>	Relational, Multi-model ⓘ	0.96	+0.06	-0.19
193.	↑	198.	↑	220.	<b>StarRocks</b>	Relational	0.95	+0.06	-0.07
194.	↑	197.	↓	186.	<b>Dolt</b>	Relational, Multi-model ⓘ	0.93	+0.02	-0.49
195.	↓	191.	↑	253.	<b>Alibaba Cloud AnalyticDB for MySQL +</b>	Relational, Multi-model ⓘ	0.92	-0.04	+0.13
		11.		11.	Apache Cassandra +	Wide-column, Multi-model ⓘ	98.94	+1.94	+1.14
		12.		12.	Microsoft Access	Relational	93.76	-2.61	-34.81
		13.		13.	Splunk	Search engine	93.02	-3.08	+1.63
		14.	↑	15.	Databricks +	Multi-model ⓘ	84.24	-0.22	+9.06
		15.	↓	14.	MariaDB +	Relational, Multi-model ⓘ	83.44	-3.09	-17.01
		16.		16.	Microsoft Azure SQL Database	Relational, Multi-model ⓘ	72.95	-2.08	-9.78
257.	↑	263.	↑	311.	<b>Alibaba Cloud Log Service +</b>	Search engine	0.44	+0.02	+0.07
258.	↑	262.	↑	272.	<b>Kingbase</b>	Relational, Multi-model ⓘ	0.44	+0.01	-0.19
		19.	↓	10.	Google BigQuery +	Relational	52.07	-2.00	-5.00
		20.		20.	FileMaker	Relational	45.20	-1.47	-8.40
		21.		21.	Neo4j +	Graph	42.68	-1.22	-7.71



# 数据库

## “中国知网”：2019年，2021年

The screenshot shows the CNKI website interface. At the top, there is a navigation bar with links for 手机版 (Mobile), English, 旧版入口 (Old Version Entry), 网站地图 (Site Map), 帮助中心 (Help Center), 购买知网卡 (Purchase CNKI Card), 充值中心 (Recharge Center), 个人/机构馆 (Personal/Institutional Library), and 我的CNKI (My CNKI). Below the navigation bar is a search area with a dropdown menu for 主题 (Topic) and a search box containing 中文文献、外文文献 (Chinese Literature, Foreign Literature). To the left of the search area are three buttons: 文献检索 (Literature Search), 知识元检索 (Knowledge Element Search), and 引文检索 (Citation Search). Below the search area are two rows of filters. The first row includes 跨库 (Cross-database) with checkboxes for 学术期刊 (Academic Journal), 博硕 (Master/Doctoral), 会议 (Conference), 报纸 (Newspaper), 年鉴 (Yearbook), and 专利 (Patent). The second row includes 单库 (Single Database) with checkboxes for 图书 (Books), 古籍 (Ancient Books), 法律法规 (Laws and Regulations), 政府文件 (Government Documents), 企业标准 (Enterprise Standards), and 科技报告 (Technical Reports).

### 行业知识服务与知识管理平台

农林牧渔、卫生、科学研究

农业 食品 医疗 药业 公共卫生 国土  
检验检疫 环保 水利 气象 海洋 地震

### 研究学习平台

研究型学习平台

研究生 本科生 高职学生  
中职学生 中学生

大数据研究平台

专利分析 学术图片 统计数据 学术热点  
学者库 统计分析



# 关系数据库

12

- 1962年，CODASYL发表“信息代数”
- 1968年，IBM7090机实现集合论数据结构
- 提出关系模型的是美国**IBM**公司的**E.F.Codd**

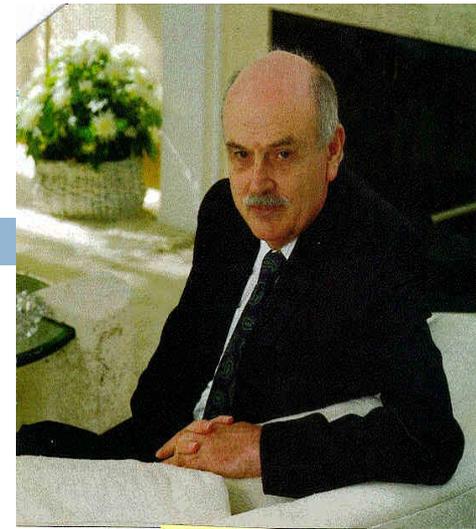
- 1970年，提出关系数据模型

E.F.Codd, “A Relational Model of Data for Large

Shared Data Banks”, 《Communication of the ACM》, 1970

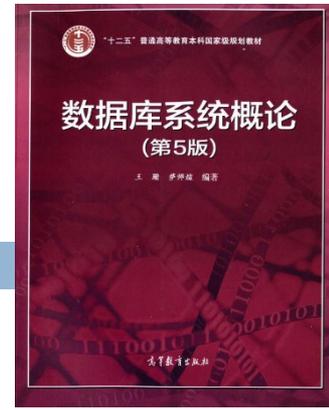
✓ 1958年以来的四分之一世纪具有里程碑意义的25篇研究论文之一

- 之后，提出了关系代数和关系演算的概念
- 1972年，提出了关系的第一、第二、第三范式
- 1974年，提出了关系的**BC**范式





# 数据存储：关系型数据库

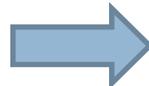


## 关系型数据库系统 (SQL)

- 关系模型中数据的逻辑结构是一张二维表，行与列
- 特点：结构化
- 概念：关系，属性，元组
- 优点1**：管理效率高（十万，百万，千万...）

记录三个学生信息：

- 学号：001，姓名：张三，年龄：20，性别：男，专业：计算机
- 学号：002，姓名：李四，年龄：19，性别：女，专业：管理
- 学号：003，姓名：王五，年龄：21，性别：男，专业：数学
- ...



关系  
学生信息表

学号	姓名	年龄	性别	专业
001	张三	20	男	计算机
002	李四	19	女	管理
003	王五	21	男	数学
...	...	...	...	...

元组



# 数据存储：关系型数据库

- 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



# 数据存储：关系型数据库

## □ 关系型数据库系统（SQL）

- 数据库操作：增、删、改、查，通过SQL语言完成
- **优点2**：存取路径对用户隐蔽，用户只要指出“干什么”，不必详细说明“怎么干”

学生信息表

学号	姓名	年龄	性别	专业
001	张三	20	男	计算机
002	李四	19	女	管理
003	王五	21	男	数学
...	...	...	...	...

### 增加学生信息：

- 学号：004，姓名：赵六，年龄：22，性别：男，专业：计算机

### 删除002号学生的信息

### 修改003号学生的专业为计算机

### 查询年龄大于20的学生信息



# 数据存储：关系型数据库

- 实现：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



# 数据存储：关系型数据库

□ 有没有不足？—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：系主任
2. 更新异常：更新系主任
3. 插入异常：数学系没有学生，插入系主任
4. 删除异常：学生毕业了



# 数据存储：关系型数据库

数据库的完整性：实体完整性，参照完整性，用户定义完整性

主码不为空

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

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

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

主码

外码

成绩范围在[0,100]



# 数据存储：关系型数据库

19

- 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(如用户)的许可证级别**大于或等于**客体（数据，表，索引等）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



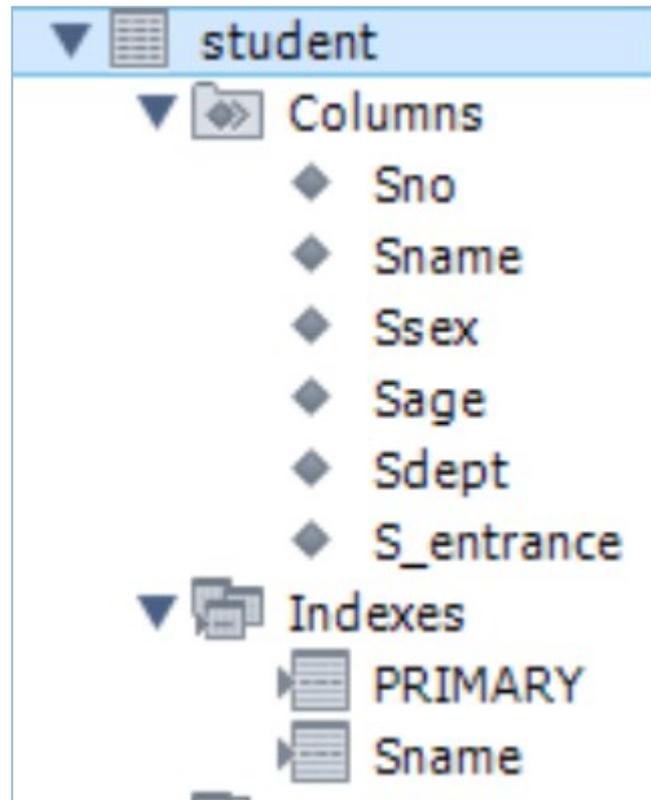
# 数据存储：关系型数据库

## 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





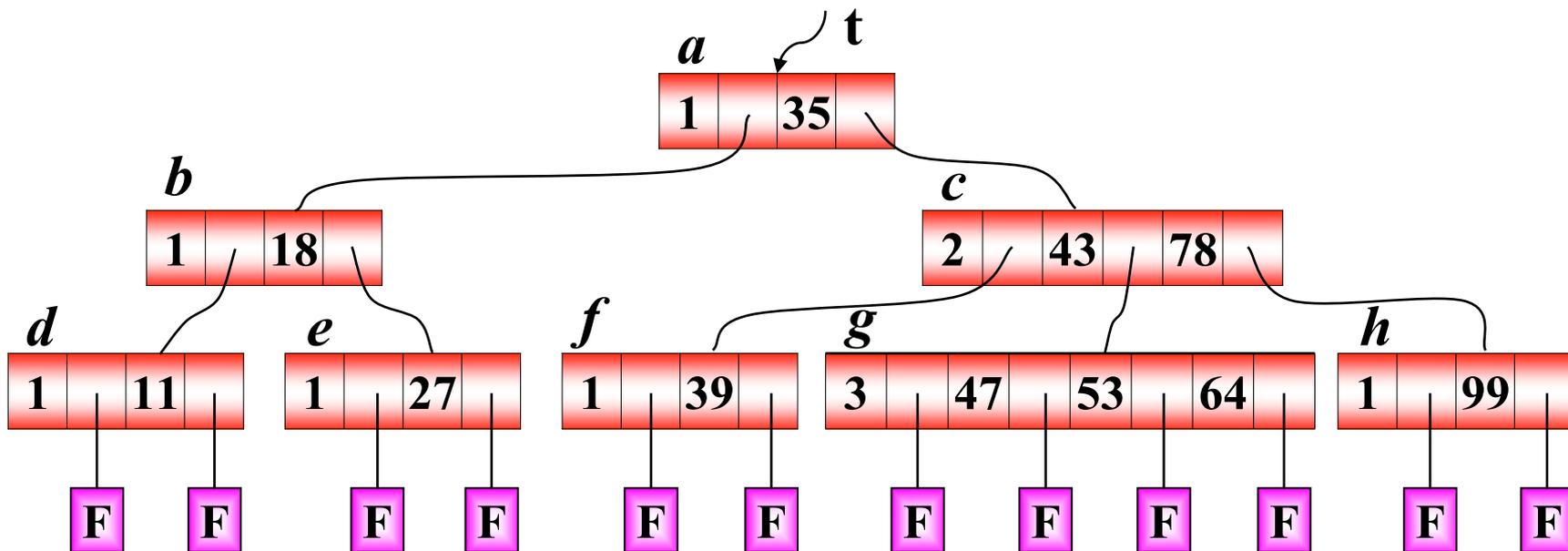
# 数据存储：关系型数据库

21

- 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



# B+/-树—B-树索引（补充）





# 数据存储：关系型数据库

- 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  $f(\text{key}) = \lfloor (\text{Ord}(\text{关键字首字母}) - \text{Ord}('A') + 1) / 2 \rfloor$

ASCII码表：A-Z: 65-90

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

若查找关键字 Zhou，会出现什么情况？ 查找Zhong？



# 数据存储：NoSQL

25

- NoSQL (Not Only SQL)
  - 非关系型的DBMS，不同于传统关系型DB的DBMS的统称
  - 里程碑：2009年，强调KV和文档数据库优点，不反对RDMBS
  - 超大规模数据存储，不需要固定模式，无需多余操作横向扩展
- 为什么NoSQL？（大数据）
  - 数据量的飞速增长：GB-TB-PB
  - RDBMS的范式约束、事务特性、磁盘IO等特点的限制
  - 开发需求的前期不明确。例如，直播带货



# 数据存储：NoSQL

## □ NoSQL vs. RDBMS

NoSQL	RDBMS
代表着不仅仅是SQL	高度组织化结构化数据
没有声明性查询语言	结构化查询语言 (SQL) (SQL)
没有预定义的模式	数据和关系都存储在单独的表中。
键 - 值对存储, 列存储, 文档存储, 图形数据库	数据操纵语言, 数据定义语言
最终一致性, 而非ACID属性	严格的一致性
非结构化和不可预知的数据	基础事务
CAP定理	
高性能, 高可用性和可伸缩性	



# 非关系型数据库

27

- **NOSQL**: 打破关系模式的限制，提供更为灵活的非关系型的数据存储和管理
  - 文档数据库: **MongoDB, CouchDB**
  - 图数据库: **Neo4j**
  - 列存储数据库: **Hbase, Cassandra**
  - 键值存储数据库: **Redis**



# 非关系型数据库

□ NoSQL (Not Only SQL) ，泛指非关系型的数据库

分类	举例	典型应用场景	数据模型	优点	缺点
文档型数据库	<b>MongoDb</b> CouchDB	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）	文档形式存储，类似 <b>JSON的KV存储</b> Value: 简单类型(字符串)，复杂类型（表格，对象）	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	<b>Neo4J</b> , 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布式集群方案
键值 (key-value)	<b>Redis</b> , Cabinet/Tyrant, Voldemort, Oracle BDB	内容缓存，用户信息等，如会话，日志等。 主要用于处理大量数据的高访问负载	<b>Key 指向 Value 的键值对</b> ，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	<b>HBase</b> , Cassandra	日志，分布式的文件系统，时空数据等	以 <b>列簇式存储</b> ，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，针对某一系列或者某几列的查询有IO优势。	功能相对局限缺乏统一查询语言



# MongoDB简介



# mongoDB

29

- **MongoDB**是一个开源的高性能无模式文档型数据库
  - 面向文档(document-oriented): 数据库中的每一条记录是一个文档对象, 采用类JSON的文档格式, 非常接近真实对象模型
  - 无模式(Schema-less): 每个文档格式都可以不同, 没有严格的模式定义, 灵活方便
  - 高性能: 拥有卓越的读写性能, 并具有高可用副本集和可扩展分片集群技术, 从先天上支持数据库的高扩展性和高伸缩性



# MongoDB简介 – 适用场景

30

## □ 适用场景

- 网站数据：MongoDB非常适合实时的插入，更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性
- 缓存：由于性能很高MongoDB适合作为信息基础设施的持久化缓存层
- 高伸缩性的集群场景：适合由数台服务器组成的大规模数据存储
- 用于对象及JSON数据的存储：MongoDB的BSON数据格式非常适合文档格式化的存储及查询

## □ 不适用场景

- 要求高度事务性的系统：例如银行结算系统，采用传统关系型数据库更适合
- 传统的商业智能应用：BI应用一般要求高度优化的特定查询方式，如上下钻取、切片等，采用数据仓库更适合



# MongoDB的使用

31

## □ 文档型数据

```
{  
  "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>
```



# 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

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

student 0.001 sec.

	_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

33



```
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)
```

二进制读取图片

构造图片文档

```
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)
```

```
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
```



# MongoDB存储举例

34

```
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" : [
    { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4TteGRXhpZgAATU0AKgAAAAGABgALAAIAAAAmAAAIYgESAAMAAAABAEEAA" },
    { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4Tg6RXhpZgAATU0AKgAAAAGABgALAAIAAAAmAAAIYgESAAMAAAABAEEAA" },
    { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4V70RXhpZgAATU0AKgAAAAGABgALAAIAAAAmAAAIYgESAAMAAAABAEEAA" },
    { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4TjerXhpZgAATU0AKgAAAAGABgALAAIAAAAmAAAIYgESAAMAAAABAEEAA" },
    { "$binary" : "/9j/4AAQSkZJRgABAQEASABIAAD/4SucRXhpZgAATU0AKgAAAAGABgALAAIAAAAmAAAIYgESAAMAAAABAEEAA" }
  ]
}
```

正如前面所说的非关系数据库MongoDB是文档型存储，数据集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

35

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	将热数据存储在内存, 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



# 总结

36

- MongoDB是一个开源的高性能无模式文档型数据库
- 在MongoDB上可以通过非常简单的方式进行数据库、集合以及文档上的各种操作
- MongoDB官方手册: <https://docs.mongodb.com/manual/>
- Pymongo: <https://pymongo.readthedocs.io/en/stable/>



# 数据存储：图像(Image)存储

37

- 如何存储图形图像数据？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** **MINIST**



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]]



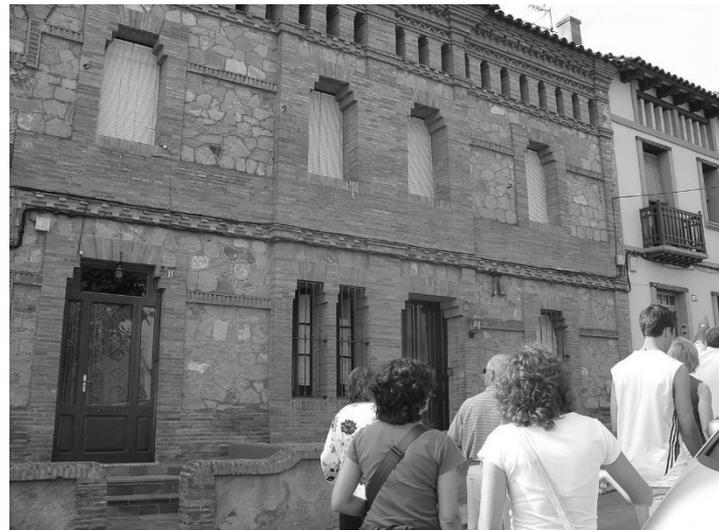
# 数据存储：图像(Image)存储

38



彩色图

height: 960  
width: 1280



灰度图

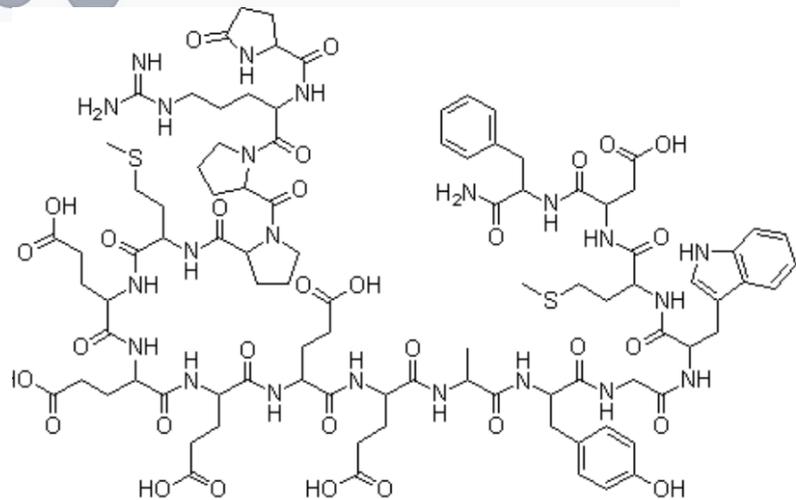
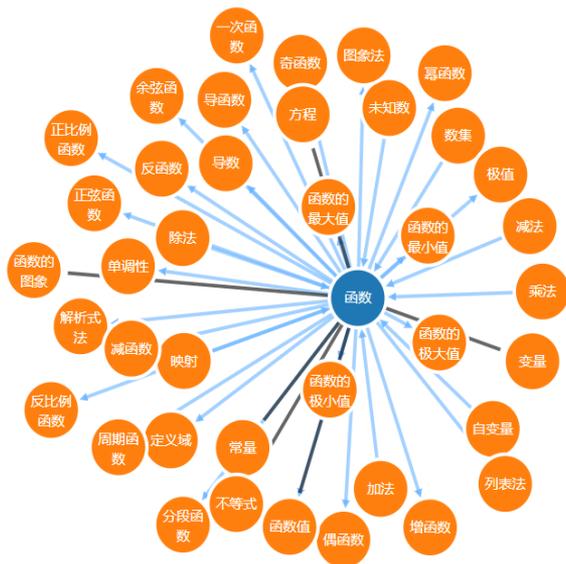
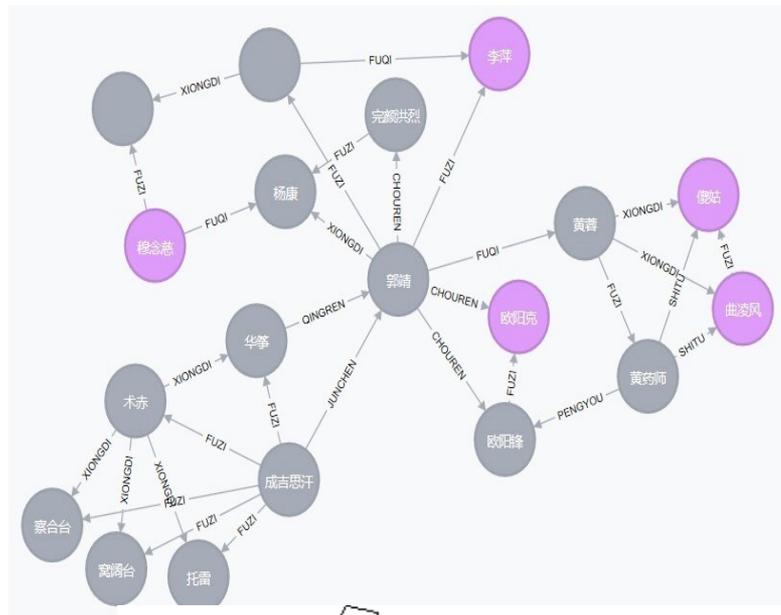
当计算机看到一张图像时，它看到的是一堆像素值。  
对于左边**彩色图**，它将看到一个 $960 \times 1280 \times 3$ 的数组，3指代的是RGB三个通道；  
对于右边**灰度图**，它将看到一个 $960 \times 1280$ 的数组。  
其中，每个数字的值从0到255不等，其描述了对应那一点的像素灰度。  
所以，计算机对图像做处理时，实际上就是对这些数组中的像素值做处理。



# 数据存储：图(Graph)数据存储

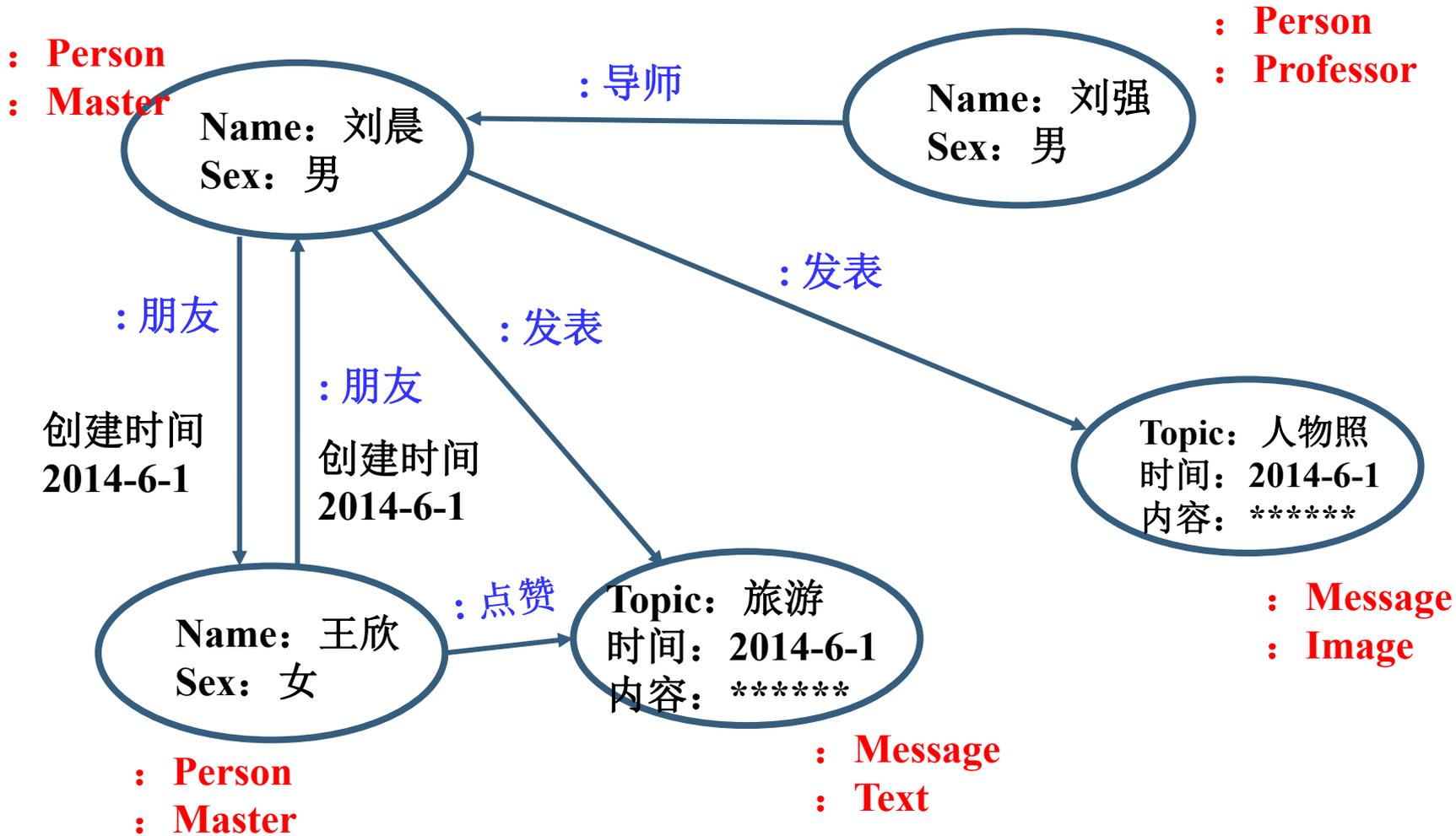
## 图数据（关系型数据）

- 社交网络
- 知识图谱
- 推荐系统
- 欺诈检测
- ...





# 基本概念-属性图 (续)





# 基本概念-属性图

41

- 属性图由顶点、边、标签、和属性组成
- 顶点可以包含一组属性，每个属性由键值对组成，即属性名和属性值
- 顶点可以被赋予一个或多个标签，每个标签代表该顶点的类别
- 边是有方向的，边 $v_i \rightarrow v_j$ 表示源点 $v_i$ 到终点 $v_j$ 的联系类似于顶点，边也可以包含属性和标签



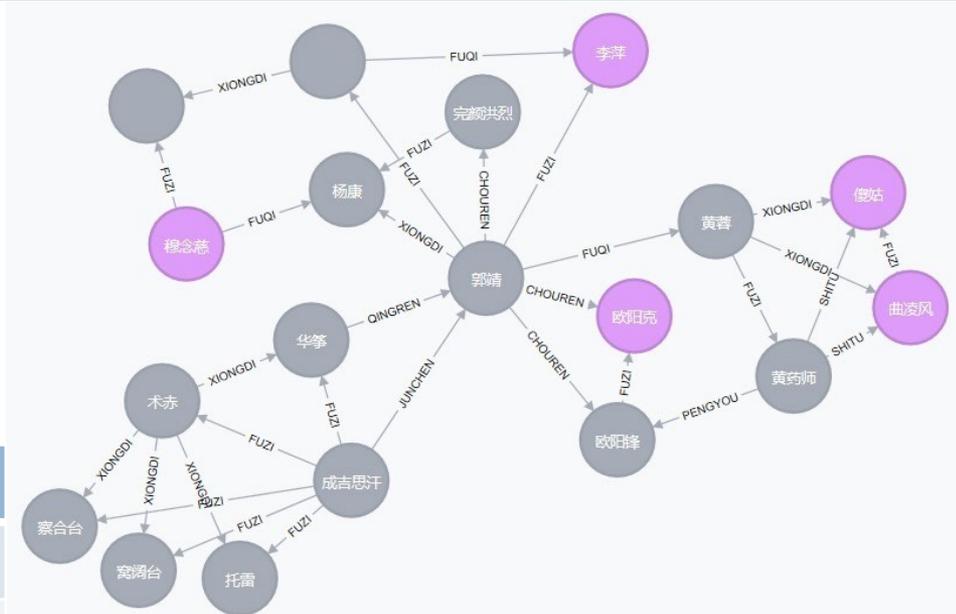
# 数据存储：图(Graph)数据存储

- 如何表示图数据？
- 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

## Neo4J

- 以“图形”的形式存储数据，符合图数据特性
- 方便删改



```

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

```

cypher是neo4j官网提供的声明式查询语言



# 数据存储：图(Graph)数据存储

## 删除郭靖

SQL

名字	子女	兄弟	父亲	年龄
<del>郭靖</del>	<del>破虏</del>	<del>杨康</del>		<del>33</del>
杨康	\	<del>郭靖</del>	完颜洪烈	
破虏	\	\	<del>郭靖</del>	10

Neo4j

```
match(n:person{name:"郭靖"}) delete n
```

## 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
```



# 数据存储：图(Graph)数据存储

## □ Neo4J vs MySQL性能对比

- 具有关系的数据
- 例：在一个社交网络中找到深度为5的朋友，数据集约为100万人，平均每人50个朋友

实验结果如下：

深度	MySQL执行时间(s)	Neo4J执行时间(s)	返回记录数
2	0.016	0.01	~2500
3	30.267	0.168	~110 000
4	1543.505	1.359	~600 000
5	未完成	2.132	~800 000



# 数据存储：键值数据库

45

- 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
    - 二进制
  - 只提供数据的基本操作
- 因为简单，所以快！

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"

第一部分是存储数字（设置，自增，获取）  
第二部分是存储队列（入队，获取长度，出队）  
第三部分是存储字典（设置值、获取值）



# 数据存储：键值数据库

46

## Redis (REmote DIctionary Server)

- key-value 存储系统

## 应用场景

- 发布订阅，地图信息，计数器等

- 电商“秒杀”：

  - 短时间内极大量访问

  - 避免“超抢”、“超卖”



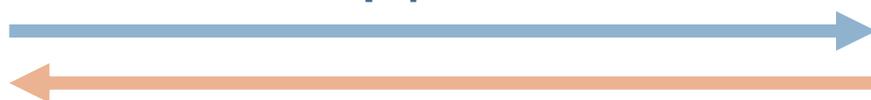
```
HMGET id Total Booked  
HINCRBY id Booked 1
```

库存总量  
订单总量

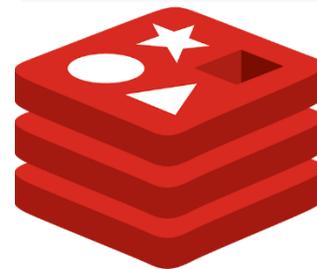
```
"goodsId" : {  
  "Total": 100  
  "Booked": 100  
}
```



下单



秒杀成功/失败





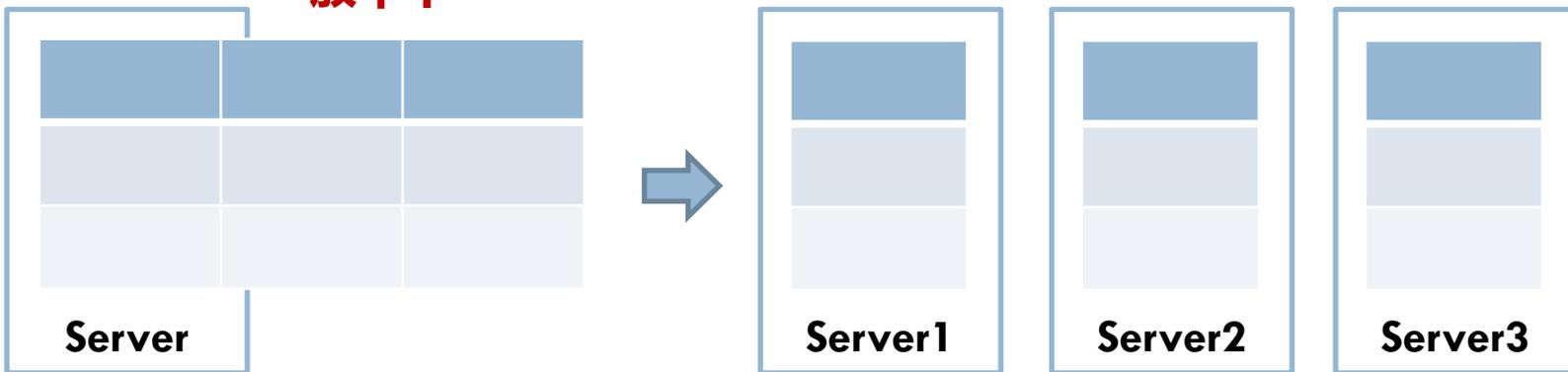
# 数据存储：列存储数据库

## 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。

放不下





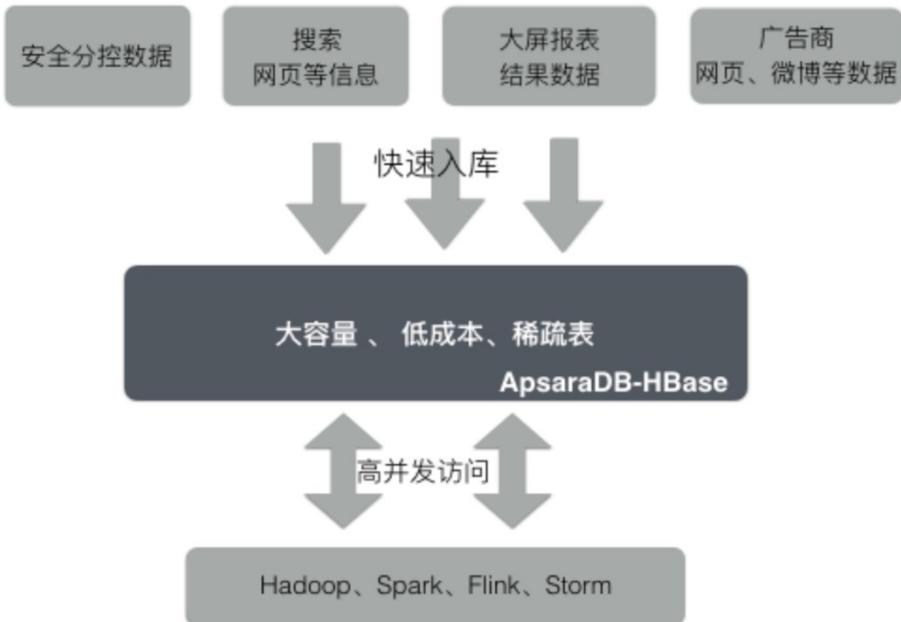
# 数据存储：列存储数据库

## □ HBase应用场景

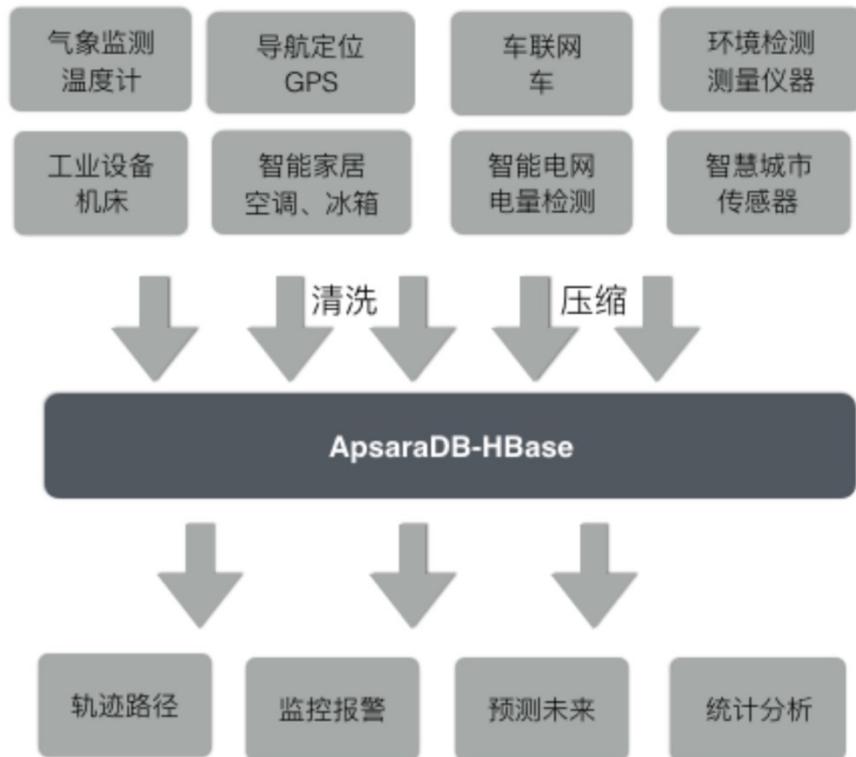
高容量：支持大规模实时数据快速入库  
高性能：支持大数据分析应用（如hadoop、spark）高速读取、分析

### □ 针对需要高容量、高性能的场景

#### 大数据实时分析



#### 大规模物联网

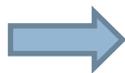




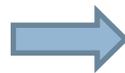
# 数据存储：大模型与数据库

## 大模型与数据库

数据库  
(1970s)



大数据分析  
(2010s)



智能大模型  
(2020s)

**数据库系统**

```

1 Create Table DemoSQLTable (
2   id int,
3   myGETDATE smalldatetime default GETDATE(),
4   myCurrentTimeStamp datetime default CURRENT_TIMESTAMP,
5   mySYSDATETIME datetime2 default SYSDATETIME()
6 );
7 GO
8
9 insert into DemoSQLTable (ID) values (1);
10 GO
11
12 select * from DemoSQLTable;
13

```

id	myGETDATE	myCurrentTimeStamp	mySYSDATETIME
1	2021-12-25 03:25:00	2021-12-25 03:24:48.920	2021-12-25 03:24:48.9233333

**SQL**

**互联网**

**搜索引擎**

**大模型**

**生成试问答**



# 数据存储

**Beautiful Soup**

[http://beautifulsoup.readthedocs.io/zh\\_CN/latest/](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](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](http://scrapy-chs.readthedocs.io/zh_CN/0.24/intro/tutorial.html)



# 数据分析基础

51

- 数据采集 Data Collection
- 数据存储 Data Storage
- 数据预处理 Data Preprocessing
- 特征工程 Feature engineering

