数据科学导论

Introduction to Data Science

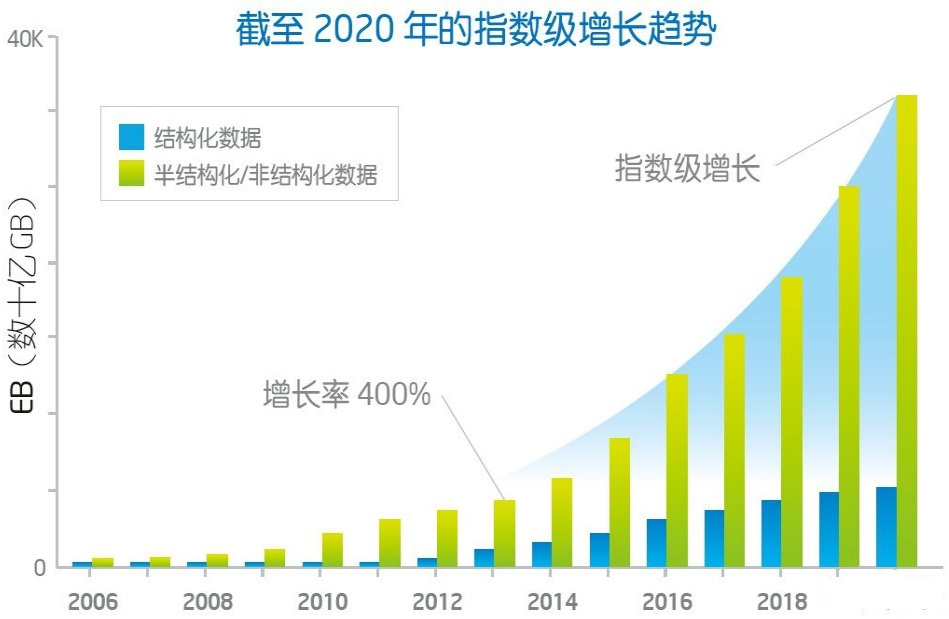
第二章 数据分析基础

黄振亚，陈恩红，刘淇
Email: huangzhy@ustc.edu.cn

课程主页：
http://staff.ustc.edu.cn/~huangzhy/Course/DS2021.html

回顾：数据分析基础

- 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



回顾：数据采集

- 数据检索
- 公开数据
- 批量数据获取
  - 网络爬虫
- 数据筛选

回顾：数据检索

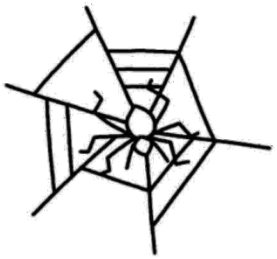
- 最简单、最灵活的数据获取方式就是依靠检索
- 学会使用搜索引擎
  - 百度：适合于搜索中文信息
  - Google：更适合搜索英文信息

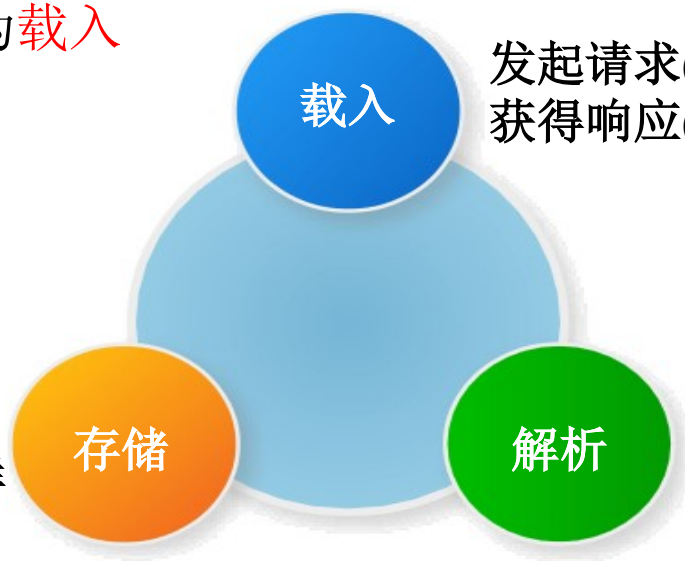
回顾：公开数据

- 代表性公开数据集
  - 用户评分MovieLens：https://grouplens.org/datasets/movielens/
  - 文本数据-头条：https://github.com/aceimnorstuvwxz/toutiao-text-classfication-dataset
  - 金融数据-股票：https://github.com/asxinyu/Stock
  - 网络数据-Large scale network：https://snap.stanford.edu/data/
  - 教育数据:
    - ASSISTmentsData-学业: https://sites.google.com/site/assistmentsdata/home/
    - BASEGroup: https://github.com/bigdata-ustc/EduData
  - 阿里天池数据-数据平台：https://tianchi.aliyun.com/dataset/
  - 公开大数据竞赛的数据：KDDCup，NeurIPS Challenge

回顾：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。
    - 请求网站并提取数据的自动化程序
  - 爬虫的行为可以划分为：载入、解析、存储，
    - 最复杂的部分为载入





回顾：抓取微博评论

获得评论的json格式

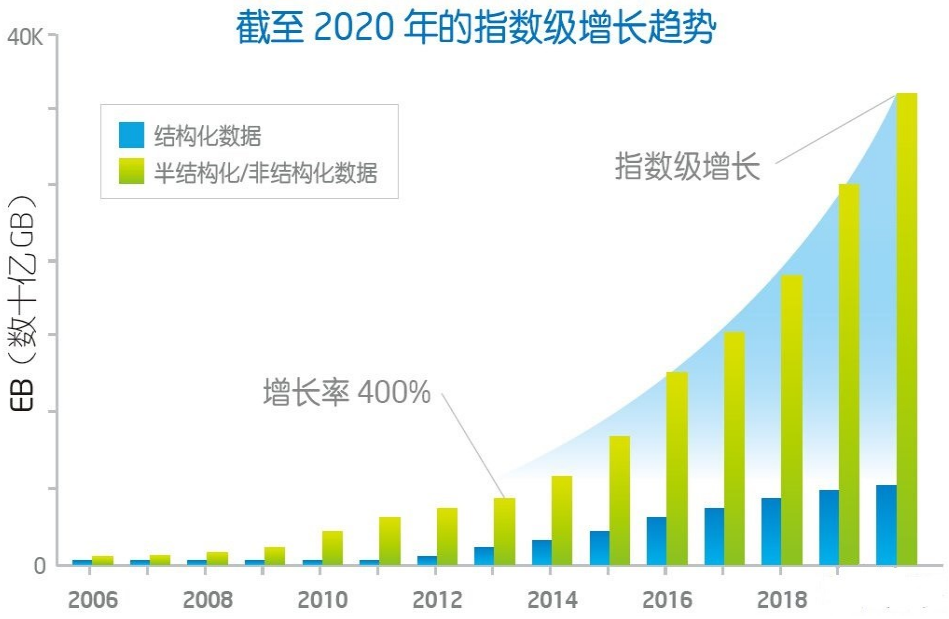

```
{
    "mod_type":"mod/pagelist",
    "previous_cursor":"",
    "next_cursor":"",
    "card_group":[
        {
            "id":4142016554789113,
            "created_at":"08-18 08:46",
            "source":"柔光自拍vivo X7",
            "user":Object{...},
            "text":"回复<a
href="/n/%E9%82%93%E8%B6%85">@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
……	……	……

数据分析基础

- 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(F) 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.

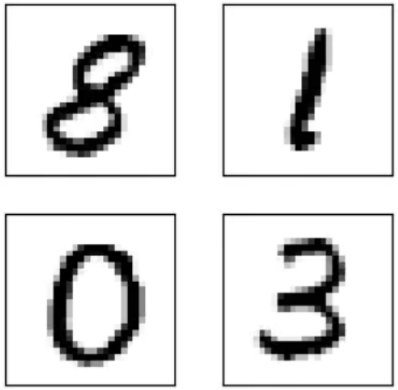
Question:

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

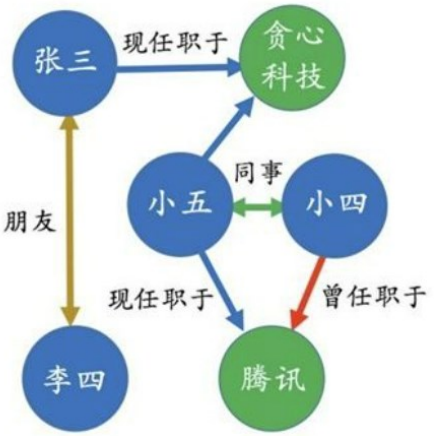
Answer:

inherent difficulty

Image



Graph

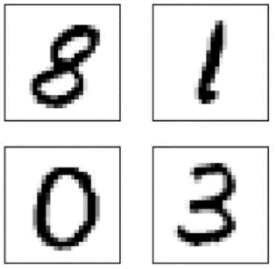


数据存储

- 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>
```

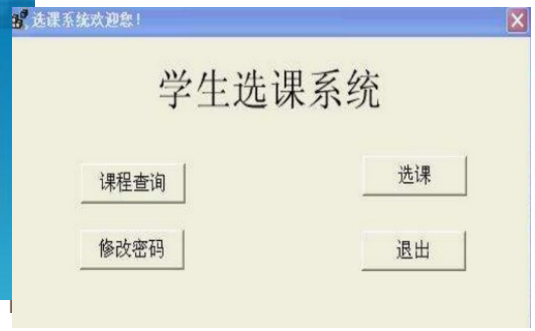


数据存储

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

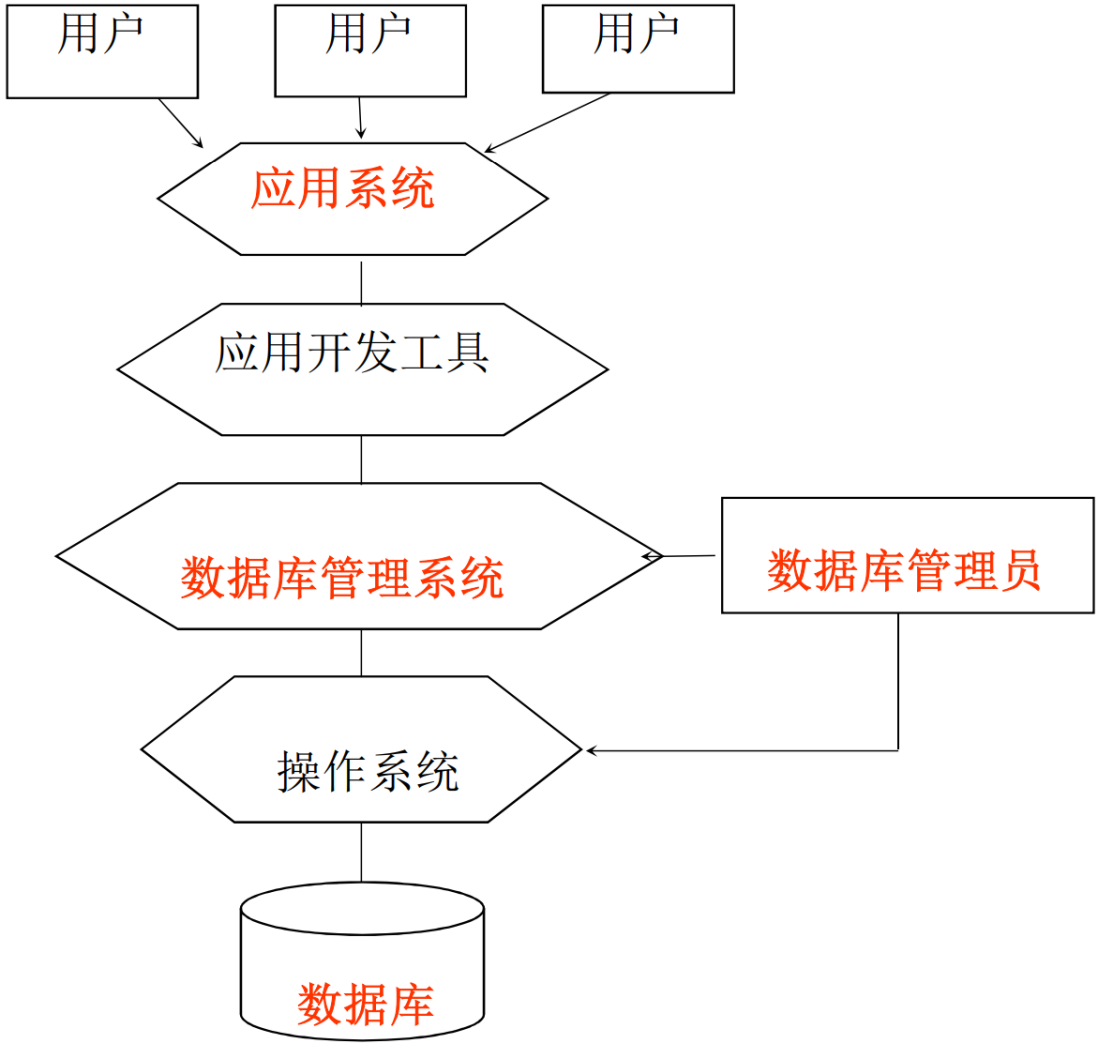
数据存储：数据库

- 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

数据库系统



数据存储：数据库

- 数据库有哪些

数据存储：数据库

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

378 systems in ranking, September 2021

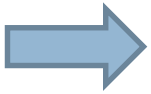
Rank Sep 2021	Rank Aug 2021	Rank Sep 2020	DBMS	Database Model	Score Sep 2021	Score Aug 2021	Score Sep 2020
1.	1.	1.	Oracle	Relational, Multi-model	1271.55	+2.29	-97.82
2.	2.	2.	MySQL	Relational, Multi-model	1212.52	-25.69	-51.72
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	970.85	-2.50	-91.91
4.	4.	4.	PostgreSQL	Relational, Multi-model	577.50	+0.45	+35.22
5.	5.	5.	MongoDB	Document, Multi-model	496.50	-0.04	+50.02
6.	6.	↑ 7.	Redis	Key-value, Multi-model	171.94	+2.05	+20.08
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	166.56	+1.09	+5.32
8.	8.	8.	Elasticsearch	Search engine, Multi-model	160.24	+3.16	+9.74
9.	9.	9.	SQLite	Relational	128.65	-1.16	+1.98
10.	↑ 11.	10.	Cassandra	Wide column	118.99	+5.33	-0.18
14.	14.	↑ 15.	Hive	Relational	85.58	+1.64	+14.41
15.	15.	↑ 17.	Microsoft Azure SQL Database	Relational, Multi-model	78.26	+3.11	+17.81
16.	16.	16.	Amazon DynamoDB	Multi-model	76.93	+2.03	+10.75
17.	17.	↓ 14.	Teradata	Relational, Multi-model	69.68	+0.86	-6.72
18.	18.	↑ 21.	Neo4j	Graph	57.63	+0.68	+7.00
22.	↓ 20.	↓ 20.	Solr	Search engine, Multi-model	49.81	-1.26	-1.81
23.	↓ 22.	↓ 18.	SAP Adaptive Server	Relational, Multi-model	47.01	-0.60	-7.00
24.	24.	↓ 22.	HBase	Wide column	45.05	+0.41	-3.30

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

属性

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

主码　　外码

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

成绩范围在[0,100]

数据存储：关系型数据库

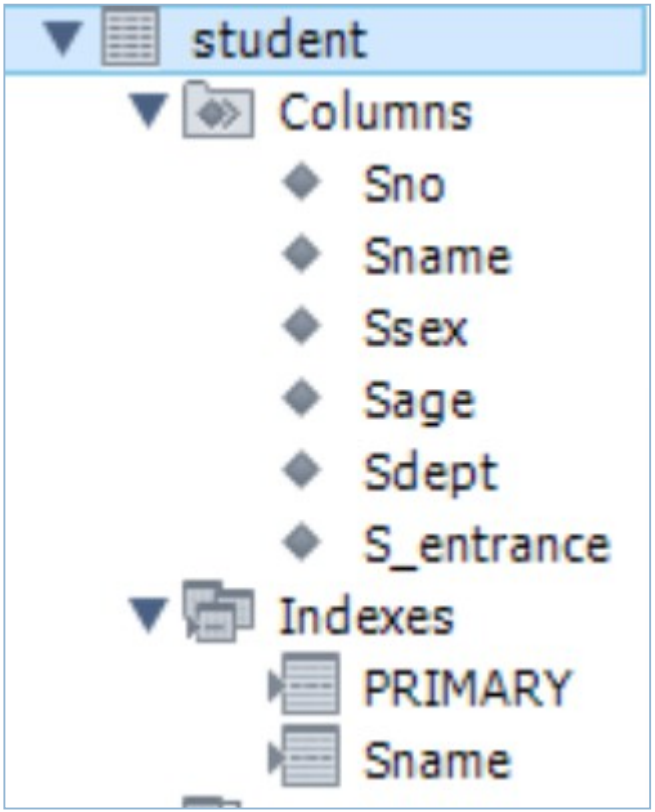
- 数据库的安全性：自主存取控制与强制存取控制
  - 自主存取控制（Discretionary Access Control，简称DAC）
    - 用户可“自主”地决定将数据的存取权限授予何人、决定是否也将“授予”的权限授予别人
  - 强制存取控制（Mandatory Access Control，简称 MAC）
    - 系统“强制”地给用户和数据标记安全等级

(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

- 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**

数据存储：关系型数据库

- B+/B-树索引
- 从二叉树讲起

比较的关键字次数

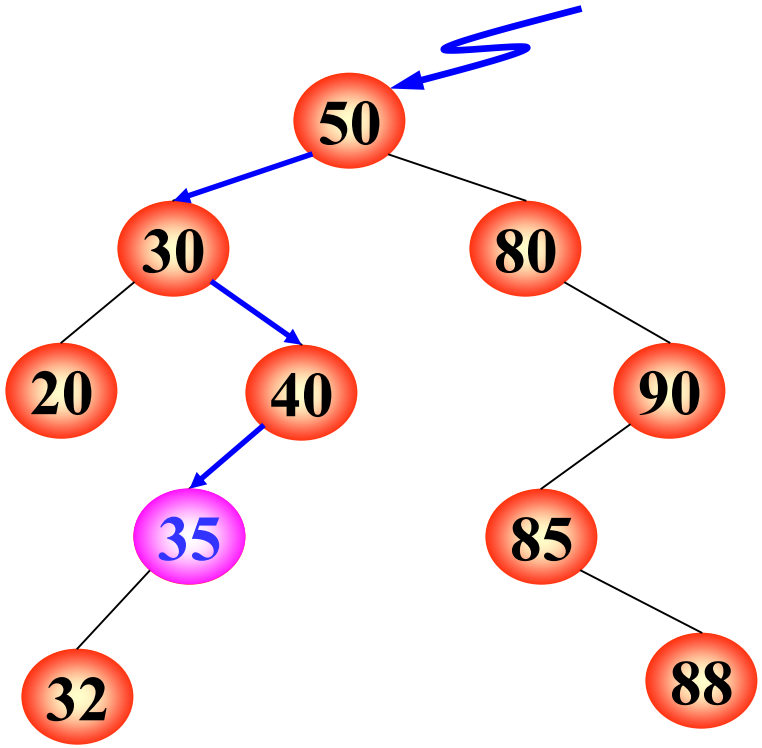
=

此结点所在层次数

最多的比较次数

=

树的深度

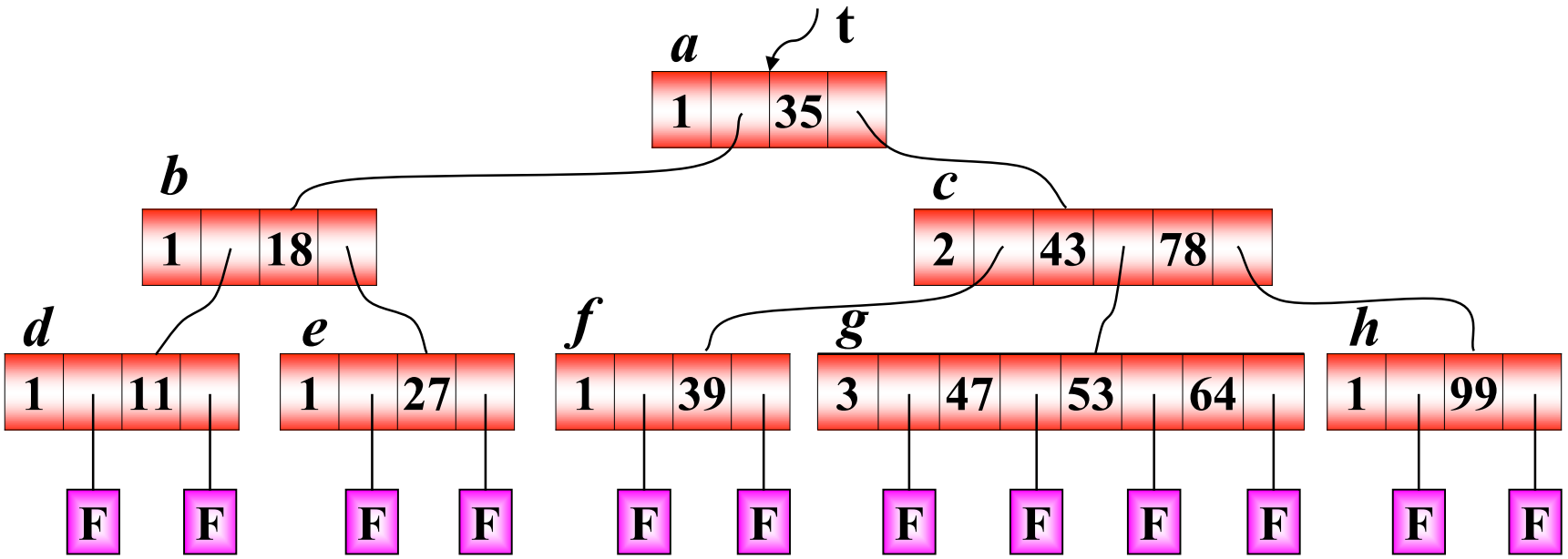


查找关键字：35

数据存储：关系型数据库

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）。



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 (key) = ⌊(Ord(关键字首字母) - Ord('A') + 1) / 2⌋

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

- 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	

数据存储：NoSQL

- NoSQL（Not Only SQL），泛指非关系型的DBMS

分类	举例	典型应用场景	数据模型	优点	缺点
文档型数据库	**MongoDb** 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JSON的KV存储 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	**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布式集群方案
键值（key-value）	**Redis,** 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	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	**HBase,** Cassandra	日志，分布式的文件系统，时空数据等	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。	功能相对局限缺乏统一查询语言

数据存储：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。Value可以包含其他文档，数组及文档数组
  - 几乎可以支持RDBMS的绝大多数功能，还支持索引

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

与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

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：在数据集合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"]})
```

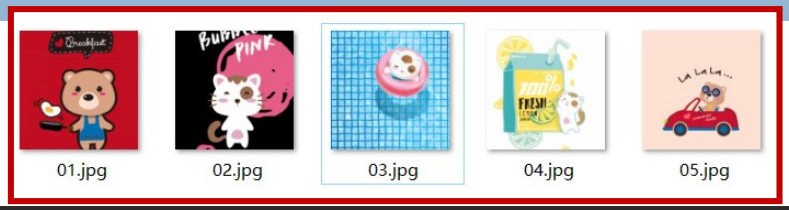
```
db.getCollection('student').find({})
```

document对字段没有强约束，Value可以是各种类型的文档

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




```
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

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	embedded document	表连接，MongoDB不支持，但支持文档嵌套
primary key	primary key	主键，MongoDB自动将_id字段设置为主键

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

数据存储：图像(Image)存储

- 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

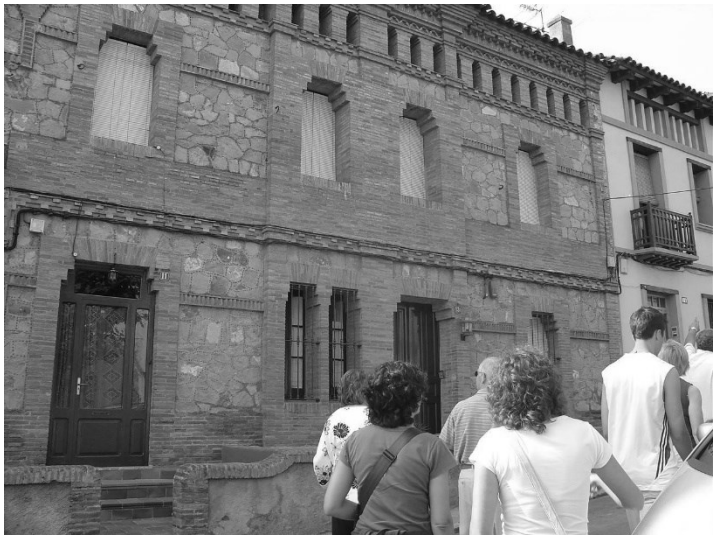
存储图像数据　MINIST

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

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。

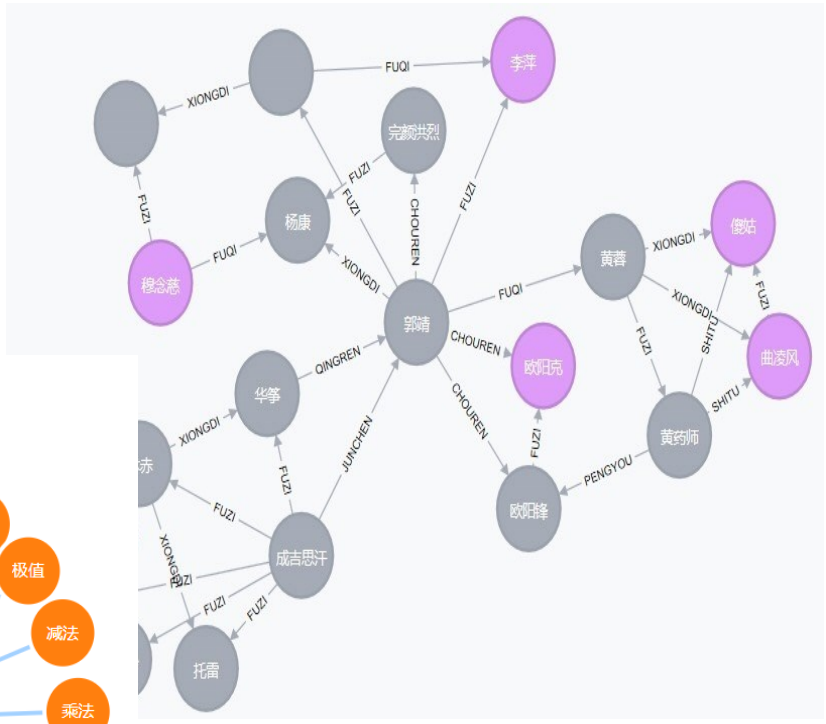
数据存储：NoSQL

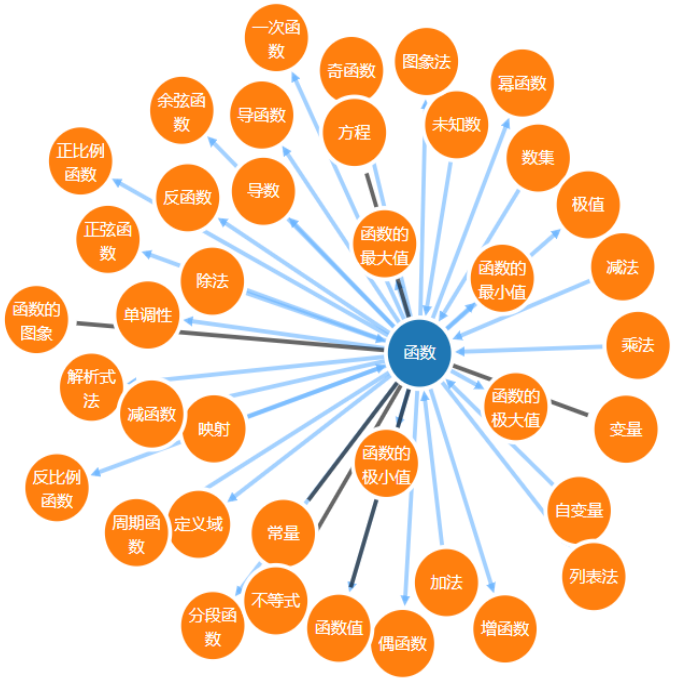
- NoSQL（Not Only SQL），泛指非关系型的DBMS

分类	举例	典型应用场景	数据模型	优点	缺点
文档型数据库	**MongoDb** 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JSON的KV存储 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	**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布式集群方案
键值（key-value）	**Redis,** 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	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	**HBase,** Cassandra	日志，分布式的文件系统，时空数据等	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。	功能相对局限缺乏统一查询语言

数据存储：图形(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
  - …

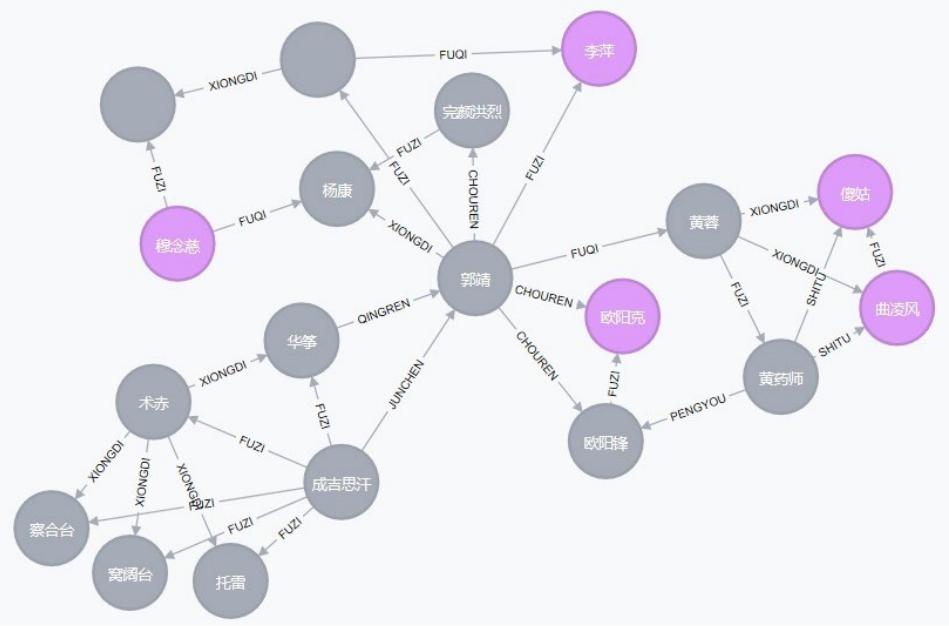




数据存储：图形(Graph)数据存储

- 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




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

- Neo4J
  - 以“图形”的形式存储数据，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

数据存储：图形(Graph)数据存储

- 删去郭靖

SQL

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

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

- 杨康和破虏是什么关系？


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

数据存储：图形(Graph)数据存储

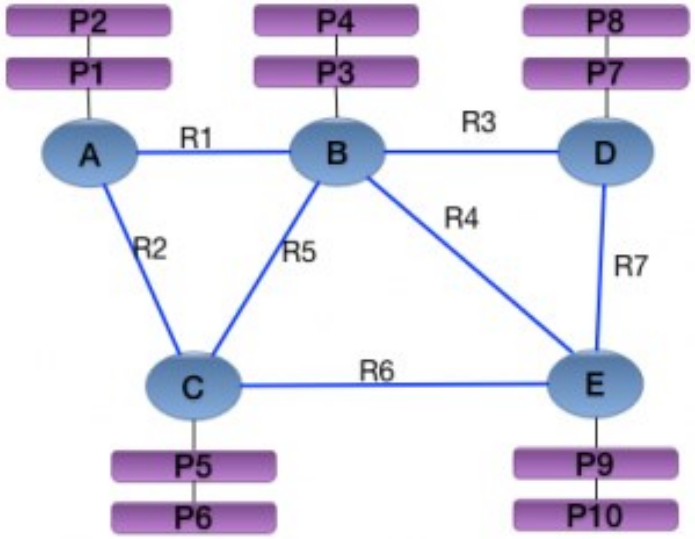
- Neo4J vs MySQL性能对比
  - 具有关系的数据
  - 例：在一个社交网络中找到深度为5的朋友，数据集约为100万人，平均每人50个朋友

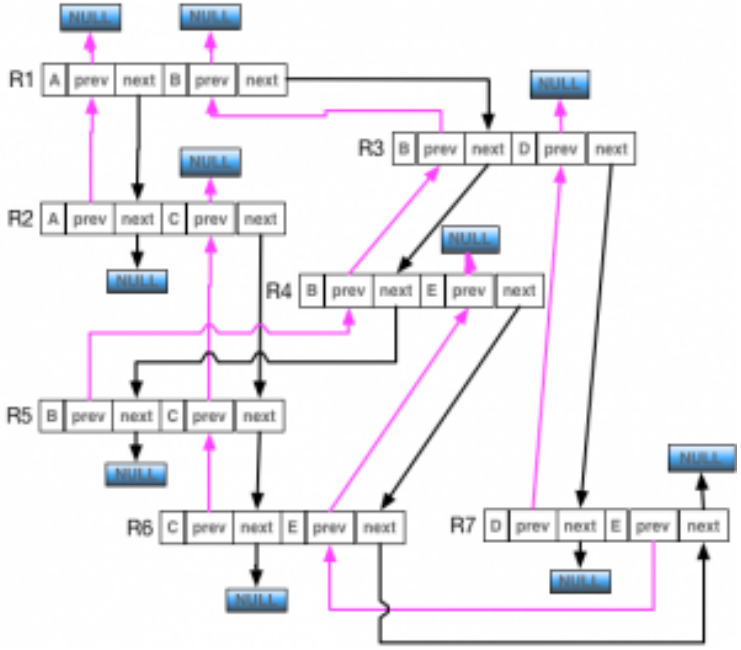
实验结果如下:

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：图形(Graph)数据存储

- 为什么Neo4j适合存储图数据？
  - **存储结构和数据查询方式，均基于图论**
  - 使用指针遍历所有节点及关系
  - 符合节点+关系的图结构





直接在点和边中保存相应点/边/属性的物理地址，直接进行寻址的遍历方法，免去了基于索引进行扫描查找的开销，实现从O(logn)到O(1)的性能提升。这种图处理方式就叫做**免索引邻接**。它不会随着图数据量的增加影响。仅跟遍历所涉及的数据集有关。

数据存储：NoSQL

- NoSQL（Not Only SQL），泛指非关系型的DBMS

分类	举例	典型应用场景	数据模型	优点	缺点
文档型数据库	**MongoDb** 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JSON的KV存储 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	**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布式集群方案
键值（key-value）	**Redis,** 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	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	**HBase,** Cassandra	日志，分布式的文件系统，时空数据等	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。	功能相对局限缺乏统一查询语言

数据存储：键值数据库

- 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

- 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”

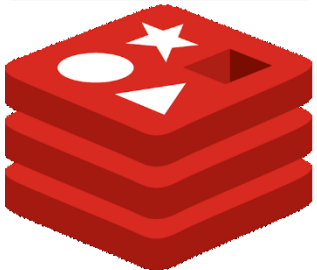

```
HMGET id Total Booked
HINCRBY id Booked 1
```

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：NoSQL

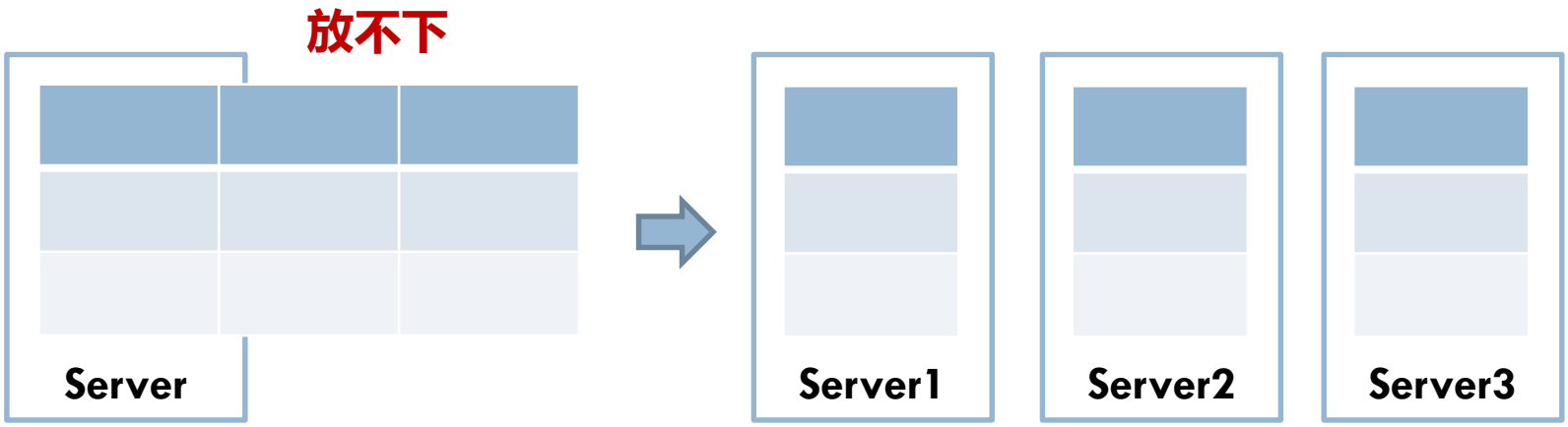
- NoSQL（Not Only SQL），泛指非关系型的DBMS

分类	举例	典型应用场景	数据模型	优点	缺点
文档型数据库	**MongoDb** 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JSON的KV存储 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	**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布式集群方案
键值（key-value）	**Redis,** 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	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	**HBase,** Cassandra	日志，分布式的文件系统，时空数据等	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。	功能相对局限缺乏统一查询语言

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。

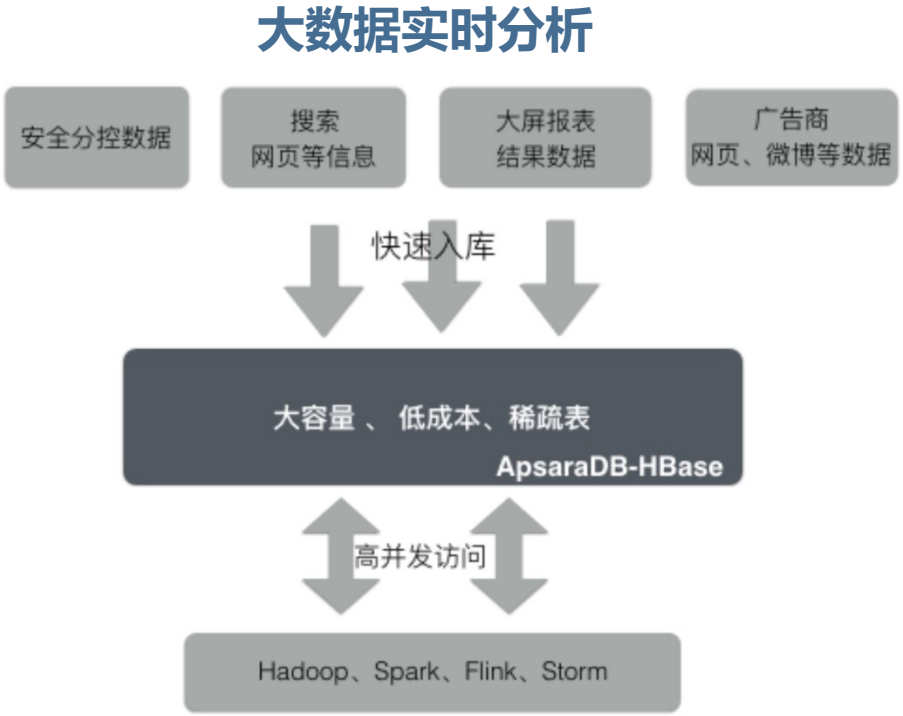


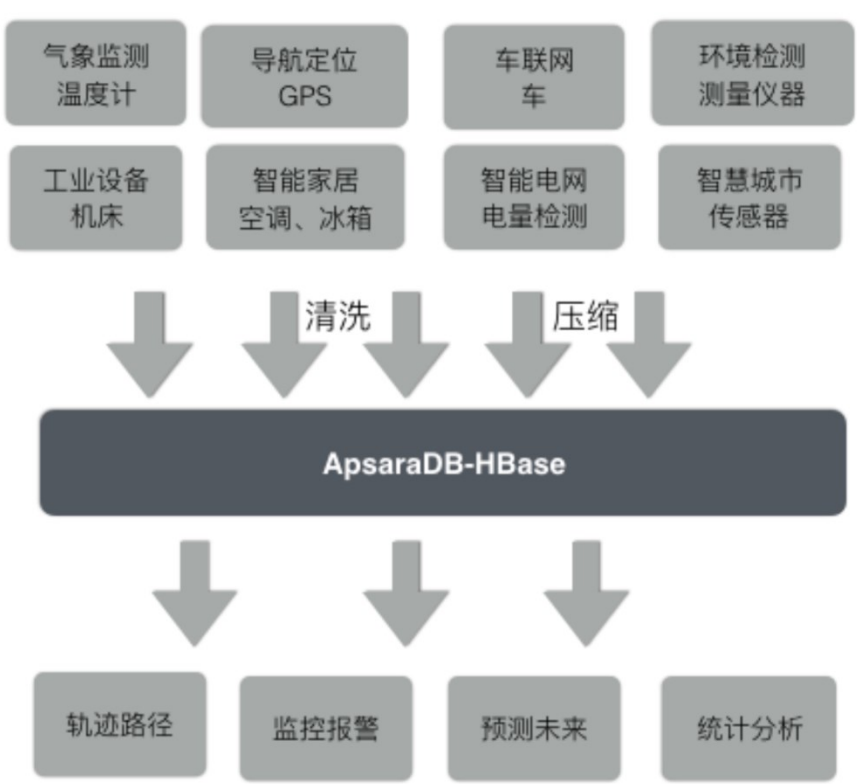
数据存储：列存储数据库

- 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





数据存储：NoSQL

- 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。

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	

数据存储

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

数据分析基础

- 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