

数据管理

授课老师： 刘耿耿
联系方式： 13950363682
Email： 329717501@qq.com
liugenggeng@fzu.edu.cn



数据管理概述



关系数据库



分布式文件系统



新型数据管理与查询系统

PART 1 数据管理概述

本章重点介绍数据存储与管理技术的概念与发展过程，选择经典的关系数据库技术以及大数据时代的分布式文件系统技术、NoSQL与Sql on Hadoop技术新型大数据存储与查询技术进行介绍。

数据管理的内涵

□ 数据管理技术

数据管理技术是指对数据进行分类、编码、存储、索引和查询，是大数据处理流程中的关键技术，负责数据从落地存储（写）到查询检索（读）的核心系统。数据管理技术从最早人们使用文件管理数据，到数据库、数据仓库技术的出现与成熟，再到大数据时代NoSQL等新型数据管理系统的涌现，一直是数据领域研究和工程领域的热点。

□ 数据库

数据库(Database)是按照数据结构来组织、存储和管理数据的,建立在计算机存储设备上的仓库。简单来说本身可视为电子化的文件柜，用户可以对文件中的数据进行新增、截取、更新、删除等操作。严格来说，数据库是长期储存在计算机内、有组织的、可共享的数据集合。

数据管理历史

□ 关系数据库

上世纪70年代，IBM公司的E.F.Codd开创了关系数据库理论，80年代随着事务处理模型的完善，关系数据管理在学术届和工业界取得主导地位，并一直保持到今天。关系数据库的核心是将数据保存在由行和列组成的简单表中，而不是将数据保存在一个层次结构中。Codd开创了关系数据库和数据规范化理论研究，获得了1981年的图灵奖，关系数据库也很快成为数据库市场的主流。

□ 新型数据管理与查询系统

2010年前后，美国谷歌公司为了满足搜索业务的需求，推出了以分布式文件系统GFS（Google File System）、分布式计算框架MapReduce、列族数据库BigTable为代表的新型数据管理与分布式计算技术。Doug Cutting领衔的技术社区研发了对应的开源版本，在Apache开源社区推出，形成了Hadoop大数据技术生态，不断迭代发展出一系列大数据时代的新型数据管理技术，例如面向内存计算的Spark大数据处理软件栈，MongoDB、Cassandra等各类型NoSQL数据库，Impala、SparkSQL等分布式数据查询技术（Sql on Hadoop）。

PART 2 关系数据库

关系数据库建立在关系数据模型之上，是主要用来存储结构化数据并支持数据的插入、查询、更新、删除等操作的数据库。

01 关系模型

关系数据模型是以**集合论**中的关系概念为基础发展起来的。**关系数据模型**中无论是实体还是实体间的联系均由单一的数据结构——**关系**来表示。关系数据模型中对的数据操作通常由**关系代数**和**关系演算**两种抽象操作语言来完成，此外**关系数据模型**中还通过**实体完整性**、**参照完整性**和**自定义完整性**来确保数据的完整一致。

关系数据模型的基本数据结构就是关系（Relation），一个关系对应着一个二维表，二维表的名字就是关系名。

表3-1 学生表

关系名

学号	姓名	性别	年龄	图书证号	所在系
S3001	张明	男	22	B20050101	外语
S3002	李静	女	21	B20050102	外语
S4001	赵丽	女	21	B20050301	管理

01

关系模型：数据结构

- 如果在一个关系中存在唯一标识一个元组的属性集合（可以是单一属性构成的集合），则称该属性集合为这个关系的**键或码**。
- 用来唯一标识一个元组的最小属性集合，称为**主键（主码）**。

The diagram shows a table titled '学生表' (Student Table). The first column, '学号' (Student ID), is highlighted with a red border and has a speech bubble pointing to it labeled '主键' (Primary Key). The second column, '姓名' (Name), is highlighted with a yellow border and has a speech bubble pointing to it labeled '键' (Key). The table contains three rows of data.

学号	姓名	性别	年龄	图书证号	所在系
S3003	张磊	男	22	B20050101	外语
S3002	李静	女	21	B20050102	外语
S4006	张磊	女	21	B20050301	管理

01

关系模型：数据结构

- 从横向看，二维表中的一行被称为是关系中的一个元组（ Tuple ），关系本质上就是由同类元组构成的集合。
- 从纵向看，二维表由很多列构成，列被称为关系的属性（ Attribute ），同一个集合中的元组都由同样的一组属性值组成。
- 属性的取值范围被称为域（ Domain ），它也可以被理解为属性中值的数据类型。

属性域：男，女

表3-1 学生表

学号	姓名	性别	年龄	图书证号	所在系
S3001	张明	男	22	B20050101	外语
S3002	李静	女	21	B20050102	外语
S4001	赵丽	女	21	B20050301	管理

一个属性

一个元组

关系数据模型的数据操作分为查询和更新两类。

关系更新可细分：插入（Insert）、修改（Update）、删除（Delete）；

关系查询包括：选择（Select）、投影（Project）、并（Union）、差（Except）以及连接（Join）等。

- 插入：将一个新的元组（行）加入到现有的关系中。
- 修改：对关系中已有的数据进行修改，特指对各种属性值进行修改。
- 删除：如果关系中的一行或多行数据已经不再需要，可以用删除操作将它们从关系中彻底去掉。

1 传统的集合运算

交 (\cap)、并 (\cup)、差 ($-$)、笛卡尔积 (\times)

2 专门的关系运算

选择 (δ)、投影 (Π)、连接 (\Join) 和除 (商 \div)

3 关系代数表达式

01

关系模型：关系代数-传统的集合运算

传统的集合运算——运算将关系看成元组的集合，运算是从关系的行的角度进行。

设R和S是n元关系，而且相应的属性取自同一个域，则交（ \cap ）、并（ \cup ）、差（ $-$ ）操作定义如下：

属性域：男，女

表3-1 学生表

学号	姓名	性别	年龄	图书证号	所在系
S3001	张明	男	22	B20050101	外语
S3002	李静	女	21	B20050102	外语
S4001	赵丽	女	21	B20050301	管理

一个属性

一个元组

1. 并 (Union)

R 和 S

具有相同的目 n (即两个关系都有 n 个属性)

- 相应的属性取自同一个域

$R \cup S$

- 仍为 n 目关系，由属于 R 或属于 S 的元组组成

$$R \cup S = \{ t \mid t \in R \vee t \in S \}$$

并(续)

如R为07(1)(2)班学生登记表,

S为07(3)班学生登记表。

则： $R \cup S$ 为07级全体学生登记表。

相应的 关系模式表示为：

Student-R (姓名, 学号, 年级, 专业, 系)

Student-S (姓名, 学号, 年级, 专业, 系)

Student- $R \cup S$ (姓名, 学号, 年级, 专业, 系)

01

关系模型：关系代数-传统的集合运算

并(续)

R

A	B	C
$a1$	$b1$	$c1$
$a1$	$b2$	$c2$
$a2$	$b2$	$c1$

S

A	B	C
$a1$	$b2$	$c2$
$a1$	$b3$	$c2$
$a2$	$b2$	$c1$

 $R \cup S$

A	B	C
$a1$	$b1$	$c1$
$a1$	$b2$	$c2$
$a1$	$b3$	$c2$
$a2$	$b2$	$c1$

2. 差 (Difference)

R 和 S

- 具有相同的目 n
- 相应的属性取自同一个域

$R - S$

- 仍为 n 目关系，由属于 R 而不属于 S 的所有元组组成
 $R - S = \{ t \mid t \in R \wedge t \notin S \}$

如上例中 R 为07级各班学生登记表；

S 为07软件专业学生登记表

$R-S$ 为07级各班中非软件专业的学生登记表。

01

关系模型：关系代数-传统的集合运算

差(续)

R

A	B	C
$a1$	$b1$	$c1$
$a1$	$b2$	$c2$
$a2$	$b2$	$c1$

 $R-S$

A	B	C
$a1$	$b1$	$c1$

S

A	B	C
$a1$	$b2$	$c2$
$a1$	$b3$	$c2$
$a2$	$b2$	$c1$

3. 交 (Intersection)

R 和 S

- 具有相同的目 n
- 相应的属性取自同一个域

$R \cap S$

- 仍为 n 目关系，由既属于 R 又属于 S 的元组组成

$$R \cap S = \{ t \mid t \in R \wedge t \in S \}$$

$$R \cap S = R - (R - S)$$

如 R 为07级学生登记表； S 为软件专业学生登记表

$R \cap S$ 为07级软件专业的学生登记表。

$R - S$ ：为07级非软件专业的学生登记表。

$R \cap S = R - (R - S)$ 为07级软件专业的学生登记表

01

关系模型：关系代数-传统的集合运算

交（续）

R

A	B	C
$a1$	$b1$	$c1$
$a1$	$b2$	$c2$
$a2$	$b2$	$c1$

 $R \cap S$

A	B	C
$a1$	$b2$	$c2$
$a2$	$b2$	$c1$

S

A	B	C
$a1$	$b2$	$c2$
$a1$	$b3$	$c2$
$a2$	$b2$	$c1$

4. 广义笛卡尔积

严格地讲应该是广义的笛卡尔积（Extended Cartesian Product）

R —— n 目关系， k_1 个元组

S —— m 目关系， k_2 个元组

$R \times S$

- 列：（ $n+m$ ）列元组的集合
 - 元组的前 n 列是关系 R 的一个元组
 - 后 m 列是关系 S 的一个元组
- 行： $k_1 \times k_2$ 个元组
 - $R \times S = \{ \overbrace{t_r t_s} \mid t_r \in R \wedge t_s \in S \}$

广义笛卡尔积（续）

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

 $R \times S$

S

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

R.A	R.B	R.C	S.A	S.B	S.C
a1	b1	c1	a1	b2	c2
a1	b1	c1	a1	b3	c2
a1	b1	c1	a2	b2	c1
a1	b2	c2	a1	b2	c2
a1	b2	c2	a1	b3	c2
a1	b2	c2	a2	b2	c1
a2	b2	c1	a1	b2	c2
a2	b2	c1	a1	b3	c2
a2	b2	c1	a2	b2	c1

01

关系模型：关系代数-专门的关系运算

1 传统的集合运算

交 (\cap)、并 (\cup)、差 ($-$)、笛卡尔积 (\times)

2 专门的关系运算

选择 (δ)、投影 (Π)、连接 (\Join) 和除 (或商 \div)

3 关系代数表达式

1. 选择 (Selection)

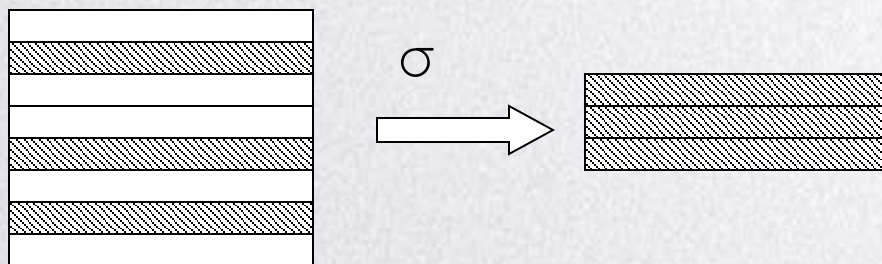
专门的关系运算——运算不仅涉及关系的行也涉及列，有时需要比较与逻辑运算来辅助进行操作。

1、选择操作 (Selection)：选择操作在关系R中选择满足给定条件的元组。

设R为一个n元关系，F是一个形如 $X_1 \theta Y_1$ 的逻辑表达式，F由逻辑运算符连接各算术表达式组成，R的选择操作定义为 $\delta_F(R) = \{ t \mid t \in R \wedge F(t) \}$ 。

选择（续）

选择运算是从关系 R 中选取使逻辑表达式 F 为真的元组，是从行的角度进行的运算



- 举例：R为07级学生登记表

$\sigma_{\text{专业}='软件'}(R)$ 表示：

从07级学生登记表中选出07级软件专业学生。

2. 投影 (Projection)

2、投影操作 (Projection)：关系R上的投影操作是从关系R中选择出若干属性列，组成新的关系。

设R为一个n元关系，R的投影操作定义为：

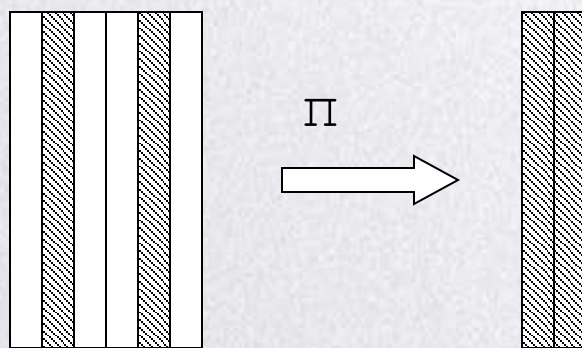
$$\Pi_A(R) = \{t[A] \mid t \in R\},$$

其中：A为R属性集的子集，

$t[A]$ 表示元组t中对应属性子集A的分量。

2. 投影 (Projection)

投影操作主要是从列的角度进行运算



- 但投影之后不仅取消了原关系中的某些列，而且还可能取消某些元组（避免重复行）

投影（续）

例、查询学生关系S中学生都来自哪些系

$\pi_{\text{系名}}(S)$ ：

结果：

系名
CS
IS
MA

01

关系模型：关系代数-专门的关系运算

选择和投影运算举例：

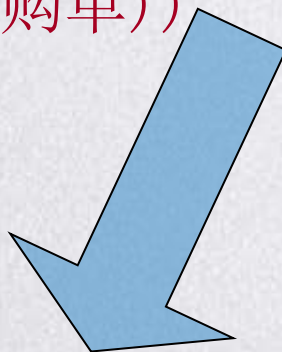
订购单关系

从订购单关系中，
选取出E3职工所经手
的订购单号和与之相
关的供应商号。

职工号	供应商号	订购单号	订购日期
E3	S7	OR67	2002/06/23
E1	S4	OR73	2002/07/28
E7	S4	OR76	2002/05/25
E6	S6	OR77	2002/06/19
E3	S4	OR79	2002/07/29
E1	S6	OR80	2002/06/22
E3	S6	OR90	2002/07/13
E3	S3	OR91	2002/10/27

π 供应商号, 订购单号 (σ 职工号='E3', (订购单))

供应商号	订购单号
S7	OR67
S4	OR79
S6	OR90
S3	OR91



3. 连接 (Join)

3、连接 (Join)：两个关系的连接操作是从两个关系的笛卡尔乘积中选取属性间满足一定条件的元组。

设： R 是 n 元关系， S 是 m 元关系，

A ， B 分别为 R 和 S 上可比的属性集

θ 是算术运算 $\{=, \neq, <, >, \leq, \geq\}$

$$R \bowtie_{A \theta B} S = \{rs \mid r \in R \wedge s \in S \wedge (r[A] \theta s[B])\}$$

$$= \delta_{A \theta B}(R \times S)$$

01 关系模型：关系代数-专门的关系运算

如 R

A	B
1	2
4	5
7	8

$R \times S$:

A	B	C	D
1	2	3	1
1	2	6	2
4	5	3	1
4	5	6	2
7	8	3	1
7	8	6	2

$R \bowtie_{B < C} S$:

A	B	C	D
1	2	3	1
1	2	6	2
4	5	6	2

S :

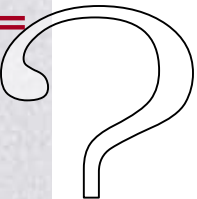
C	D
3	1
6	2

则属性B，C称为连接属性， $B < C$ 为连接条件

象集的例子

职工号	供应商号	订购单号	订购日期
E3	S7	OR67	2002/06/23
E1	S4	OR73	2002/07/28
E7	S4	OR76	2002/05/25
E6	S6	OR77	2002/06/19
E3	S4	OR79	2002/07/29
E1	S6	OR80	2002/06/22
E3	S6	OR90	2002/07/13
E3	S3	OR91	2002/10/27

- 设有如右上表所示的订购单关系，把它命名为 R ，并且进一步设 X 为属性职工号， Y 为属性集{供应商号，订购单号，订购日期}，则当 X 取值为E3时

$Y_x =$ 

{ (S7, OR67, 2002/06/23) ,
 (S4, OR79, 2002/07/29) ,
 (S6, OR90, 2002/07/13) ,
 (S3, OR91, 2002/10/27) }

$\ominus \Pi$ 供应商号, 订购单号, 订购日期 (σ 职工号='E3' (R))

01 关系模型：关系代数-专门的关系运算

4. 除 (Division)

关系R和关系S拥有共同的属性B、C， $R \div S$ 得到的属性值就是关系R包含而关系S不包含的属性，即A属性

关系R

仓库号	供应商号
WH1	S1
WH1	S2
WH1	S3
WH2	S3
WH3	S1
WH3	S2
WH5	S1
WH5	S2
WH5	S4
WH6	S2

被除关系

关系S

仓库号
WH1
WH3
WH5

除关系

÷

=

供应商号

S1

S2

商关系

- 它的含义是：至少向WH1、WH3、WH5供货的供应商号。

4. 除 (Division)

[例6] 设关系 R 、 S 分别为下图的(a)和(b)， $R \div S$ 的结果为图(c)

R

A	B	C
a_1	b_1	c_2
a_2	b_3	c_7
a_3	b_4	c_6
a_1	b_2	c_3
a_4	b_6	c_6
a_2	b_2	c_3
a_1	b_2	c_1

(a)

S

B	C	D
b_1	c_2	d_1
b_2	c_1	d_1
b_2	c_3	d_2

(b)

$R \div S$
A
a_1

(c)

分析

在关系R中，A可以取四个值{a1, a2, a3, a4}

a_1 的象集为 $\{(b_1, c_2), (b_2, c_3), (b_2, c_1)\}$

a_2 的象集为 $\{(b_3, c_7), (b_2, c_3)\}$

a_3 的象集为 $\{(b_4, c_6)\}$

a_4 的象集为 $\{(b_6, c_6)\}$

S在(B, C)上的投影为

$$\{(b_1, c_2), (b_2, c_1), (b_2, c_3)\}$$

只有 a_1 的象集包含了S在(B, C)属性组上的投影

所以 $R \div S = \{a_1\}$

4. 除 (Division)

以学生-课程数据库为例

[例] 查询至少选修1号课程和3号课程的学生号

首先建立一个临时关系 K :

Cno
1
3

关系 R 和关系 S 拥有共同的属性 B 、 C ， $R \div S$ 得到的属性值就是关系 R 包含而关系 S 不包含的属性，即 A 属性

然后求： $\pi_{Sno, Cno}(SC) \div K$

4. 除 (Division)

续 $\pi_{Sno,Cno}(SC)$

95001象集{1, 2, 3}

95002象集{2, 3}

$K=\{1, 3\}$

于是： $\pi_{Sno,Cno}(SC) \div K = \{95001\}$

Sno	Cno
95001	1
95001	2
95001	3
95002	2
95002	3

01

关系模型：数据操作

学号	姓名	性别	年龄	图书证号	所在系	学号	课程号	成绩
S3001	张明	男	22	B20050101	外语	S3001	C1	90
S3002	李静	女	21	B20050102	外语	S3001	C2	95
S4001	赵丽	女	21	B20050301	管理	S3002	C1	84
						S4001	C3	50



学号	姓名	性别	年龄	图书证号	所在系	课程号	成绩
S3001	张明	男	22	B20050101	外语	C1	90
S3001	张明	男	22	B20050101	外语	C2	95
S3002	李静	女	21	B20050102	外语	C1	84
S4001	赵丽	女	21	B20050301	管理	C3	50



投影

学号	姓名	所在系	课程号	成绩
S3001	张明	外语	C1	90
S3001	张明	外语	C2	95
S3002	李静	外语	C1	84
S4001	赵丽	管理	C3	50

- **（自然）连接**：把分散在不同关系中的数据关联在一起查看。我们希望查看学生考试的情况，但考试课表中只有学生的学号，没有学生的姓名及所在系等信息（在学生表中），此时就需要将**学生表**和**考试课表**连接起来。
- **选择**：从关系中选取满足条件的元组，例如从学生表中选出所有的男学生。
- **投影**：从关系中抽取出若干属性形成一个新的关系，例如我们只抽取学生的姓名和所在系。
- **并和差**：并操作就是将两个同类关系中的元组集合合并起来形成新的关系，差操作则是从两个同类关系的元组集合中找出不同的元组集合。

02 结构化查询语言

结构化查询语言（Structured Query Language）简称SQL，是一种数据库查询和程序设计语言，用于查询、更新和管理关系数据库系统。

结构化查询语言是高级的**非过程化编程语言**，允许用户在**高层数据结构上**工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以即使具有完全不同底层结构的不同数据库系统，也可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言语句可以**嵌套**，这使它具有极大的**灵活性**和**强大**的功能。

02 结构化查询语言：SQL历史

- SQL是IBM在其System R系统中首次提出的。1979年ORACLE公司首先提供**商用的SQL**，其后IBM公司在DB2和SQL/DS数据库系统产品中也实现了SQL。
- 1986年10月，美国ANSI采用SQL作为关系数据库管理系统的标准语言（ANSI X3.135-1986），后为国际标准化组织（ISO）采纳为国际标准。
- 1989年，美国ANSI采纳在ANSI X3.135-1989报告中定义的关系数据库管理系统的SQL标准语言，称为ANSI SQL 89，该标准替代ANSI X3.135-1986版本。
- 之后每隔一定时间ISO都会更新新版本的SQL标准，目前最新的版本已经演进到**SQL Server 2021（目前2016是最经典）**。

02 结构化查询语言：SQL构成

按照不同的用途，SQL语言通常被分成三个子集（子语言）：

- ◆ **数据定义语言（DDL：Data Definition Language）**，用于操纵数据库模式，例如数据库对象（**表、视图、索引**等）的**创建**和**删除**。数据定义语言的语句包括动词**CREATE**和**DROP**，之后用数据库对象的类型名词区分要定义的数据库对象，例如**TABLE、VIEW、INDEX**。
- ◆ **数据操作语言（DML：Data Manipulation Language）**，用于对数据库中的数据进行各类操作，包括读取和修改，其语句包括动词**SELECT、INSERT、UPDATE**和**DELETE**。它们分别用于**查找、增加、修改和删除**表中的行。
- ◆ **数据控制语言（DCL：Data Control Language）**，包括除DDL和DML之外的其他杂项语句，这些语句包括对**访问权限和安全级别的控制、事务的控制、连接会话的控制**等。

课后作业题：数据库对象包含了表、视图、索引，三者的优缺点各是什么？

为防止不同用户同时操作同一数据时产生的不良影响，现代的数据库管理系统中都引入了**事务**（Transaction）的概念。**事务由一系列的数据库操作构成**，它必须满足四个特性（被简称为**ACID特性**）：

- （1）**原子性**（Atomicity）：事务包含的所有操作要么全部正确地反映在数据库中，要么全部不反映；
- （2）**一致性**（Consistency）：事务的执行会使数据库从一种一致性的状态达到另一种一致性状态，即事务的执行不会让数据库出现不一致；
- （3）**隔离性**（Isolation）：事务之间是隔离的，每个事务都感觉不到系统中有其他事务在并发地执行；
- （4）**持久性**（Durability）：一个事务成功完成后，它对数据库的改变是永久的，即使系统出现故障也是如此。

- 事务是数据库的逻辑工作单位
 - 事务中包括的所有操作,要么都做,要么都不做

事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。

一致性状态：

数据库中只包含成功事务提交的结果

不一致状态：

数据库中包含失败事务的结果。

银行转帐：从帐号A中取出一万元，存入帐号B。

- 定义一个事务，该事务包括两个操作：

A	B
$A=A-1$	$B=B+1$

- 这两个操作要么全做，要么全不做
 - 全做或者全不做，数据库都处于一致性状态。
 - 如果只做一个操作，数据库就处于不一致性状态。

对并发执行而言,一个事务的执行不能被其他事务干扰。

- 一个事务内部的操作及使用的数据对其他并发事务是隔离的。
- 并发执行的各个事务之间不能互相干扰。

T_1	T_2
① 读A=16	
②	读A=16
③ $A \leftarrow A-1$ 写回A=15	
④	$A \leftarrow A-3$ 写回A=13

T1的修改被T2覆盖了！

- 持续性也称永久性（ Permanence ）
 - 一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。
 - 接下来的其他操作或故障不应该对其执行结果有任何影响。

04

关系数据库管理系统

关系数据库管理系统 (Relational Database Management System : RDBMS) 是管理、操作和维护关系型数据库的一种软件程序。



PART 3 分布式文件系统

分布式文件系统建立在通过网络联系在一起的多台价格相对低廉的服务器上，将要存储的文件按照特定的策略划分成多个片段分散放置在系统中的多台服务器上。

01 分布式文件系统分类

从用途来看，目前主流的分布式文件系统主要有两类：

- ◆ 第一类分布式文件系统主要面向以大文件、块数据顺序读写为特点的数据分析业务，其典型代表是Apache旗下的HDFS。
- ◆ 另一类主要服务于通用文件系统需求并支持标准的可移植操作系统接口（ Portable Operating System Interface of UNIX ，缩写为 POSIX ），其代表包括Ceph和GlusterFS。

这种分类仅表示各种分布式文件系统的专注点有所不同，并非指一种分布式文件系统只能用于某种用途。

02

Hadoop分布式文件系统（HDFS）：特点

HDFS作为Hadoop的分布式文件系统，其功能为数据的存储、管理和出错处理。它是类似于GFS的开源版本，设计的目的是用于**可靠地存储大规模的数据集**，并**提高用户访问数据的效率**。

A

适合大文件
存储和处理

可处理的文件规模可达到百MB乃至数百TB，目前应用已到PB级。

B

集群规模
可动态扩展

存储节点可在运行状态下加入到集群中，集群仍然可以正常地工作。

C

能有效保证
数据一致性

基于“**一次写入，多次读取**”设计，简化处理文件访问方式，当一个文件创建、写入并关闭后就不能再修改。

D

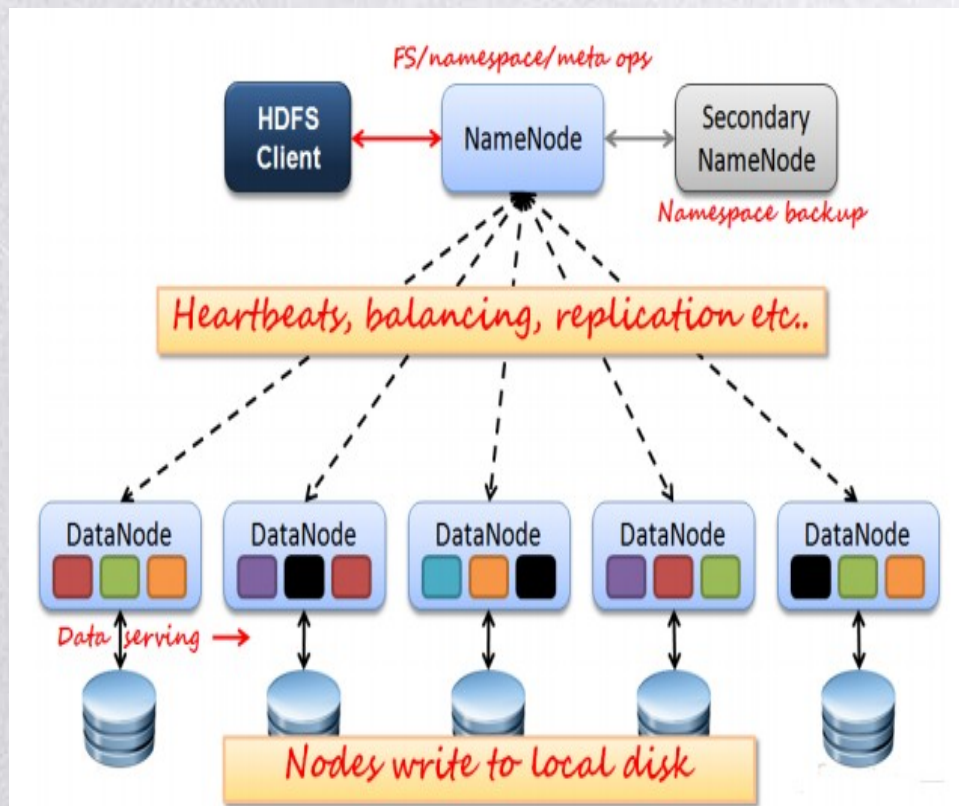
数据的吞吐量大，
跨平台移植性好

采用数据流式读写的方式，用以增加数据的吞吐量。具有很好的跨平台移植性，源代码开放。

02

Hadoop分布式文件系统（HDFS）：架构和操作

HDFS采用的是**单一主服务器的主从结构**，一个HDFS集群通常由一台主服务器和若干台数据服务器构成，有一台后备主服务器用于定期对主服务器存储的元数据进行备份，保障名称空间、元数据等系统信息的完整性。这台后备主服务器只与主服务器进行交互，对系统中的其他节点不可见。



Active Namenode

- 1.主Master(只有一个)
- 2.管理HDFS的名称空间
- 3.管理数据块映射信息
- 4.配置副本策略
- 5.处理客户端读写请求

Standby Namenode

- 1.NameNode的热备
- 2.定期合并fsimage和fsedit, 推送给NameNode
- 3.当Active NameNode出现故障时, 快速切换为新的Active NameNode

Datanode

- 1.Slave (有多个)
- 2.存储实际的数据块
- 3.执行数据块的读写

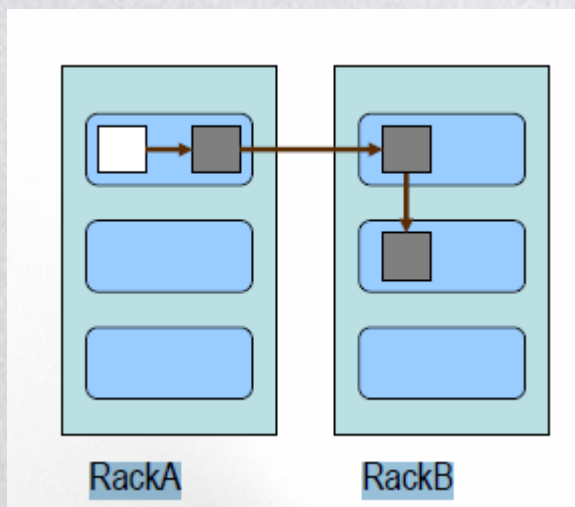
Client

- 1.文件切分
- 2.与NameNode交互, 获取文件位置信息
- 3.与DataNode交互, 读取或者写入数据;
- 4.管理HDFS 和访问HDFS

02

Hadoop分布式文件系统（HDFS）：副本管理

为了提高系统中文件数据的**可靠性**，HDFS系统提供了一种**副本机制**：默认情况下，每一个文件块都会在HDFS系统中拥有**三个副本**，副本数可以在部署集群时手动设置。通常**这三个副本会被放置在不同的数据服务器上**，这样就保证了即便其中某一个副本丢失或者损坏，都可以保证该文件块可以继续使用，甚至还可以**利用其他两个副本来恢复丢失或者损坏的那个副本**。



从应用场景来看，HDFS是专门为Hadoop这样的计算引擎而生，更适合**离线批量处理大数据**，例如电商网站对于用户购物习惯的分析。由于HDFS本身设计的特点，它不适合于经常要对文件进行更新、删除的在线业务。

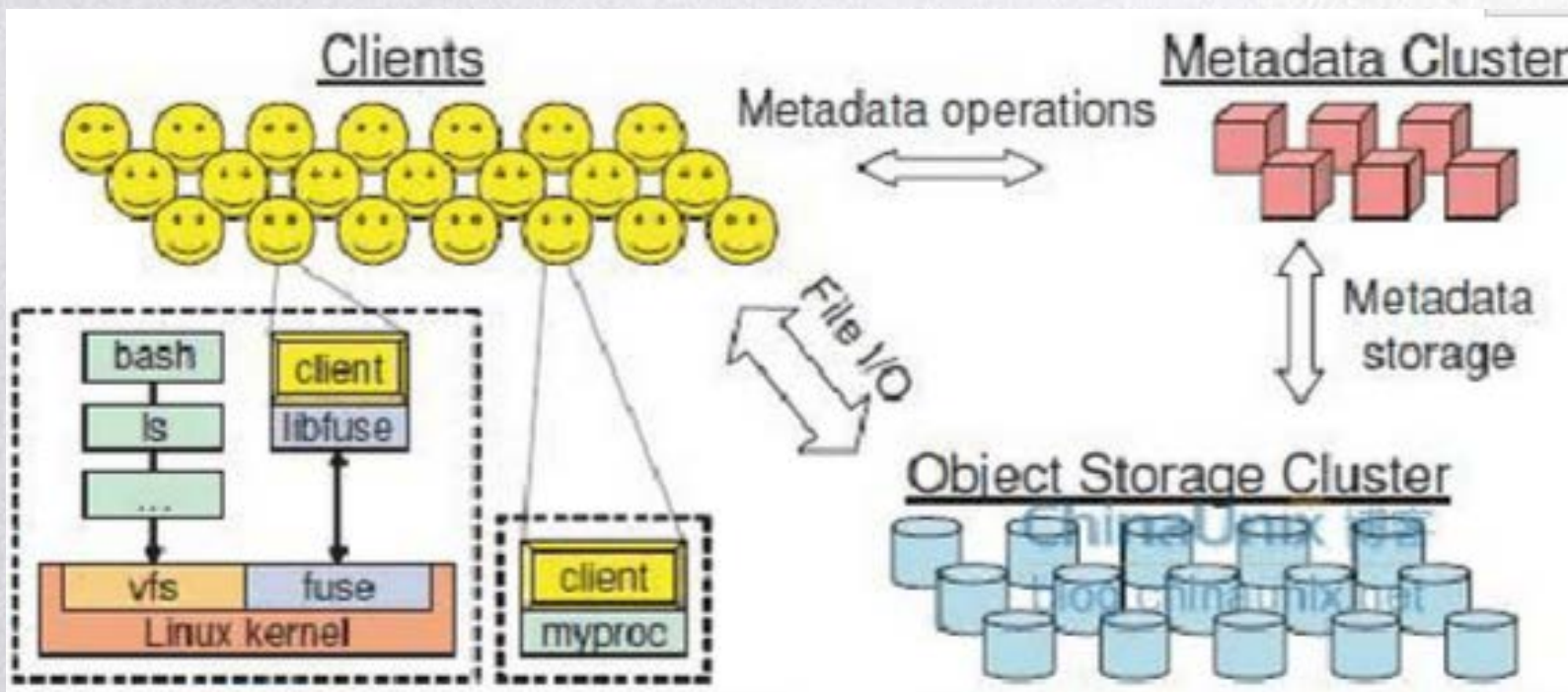
Ceph是一个高可用、易于管理、开源的分布式存储系统，可以同时提供对象存储、块存储以及文件存储服务，其优势包括统一存储能力、可扩展性、可靠性、性能、自动化的维护等等。

Ceph优势均来源于其先进的核心设计思想，可其概括为八个字——“无需查表，算算就好”（数据寻址的方式）。基于这种设计思想，Ceph充分发挥存储设备自身的计算能力，同时消除了对系统单一中心节点的依赖，从而实现了真正的无中心结构。

Ceph项目起源于其创始人Sage Weil在加州大学圣克鲁兹分校攻读博士期间的研究课题。

03 Ceph

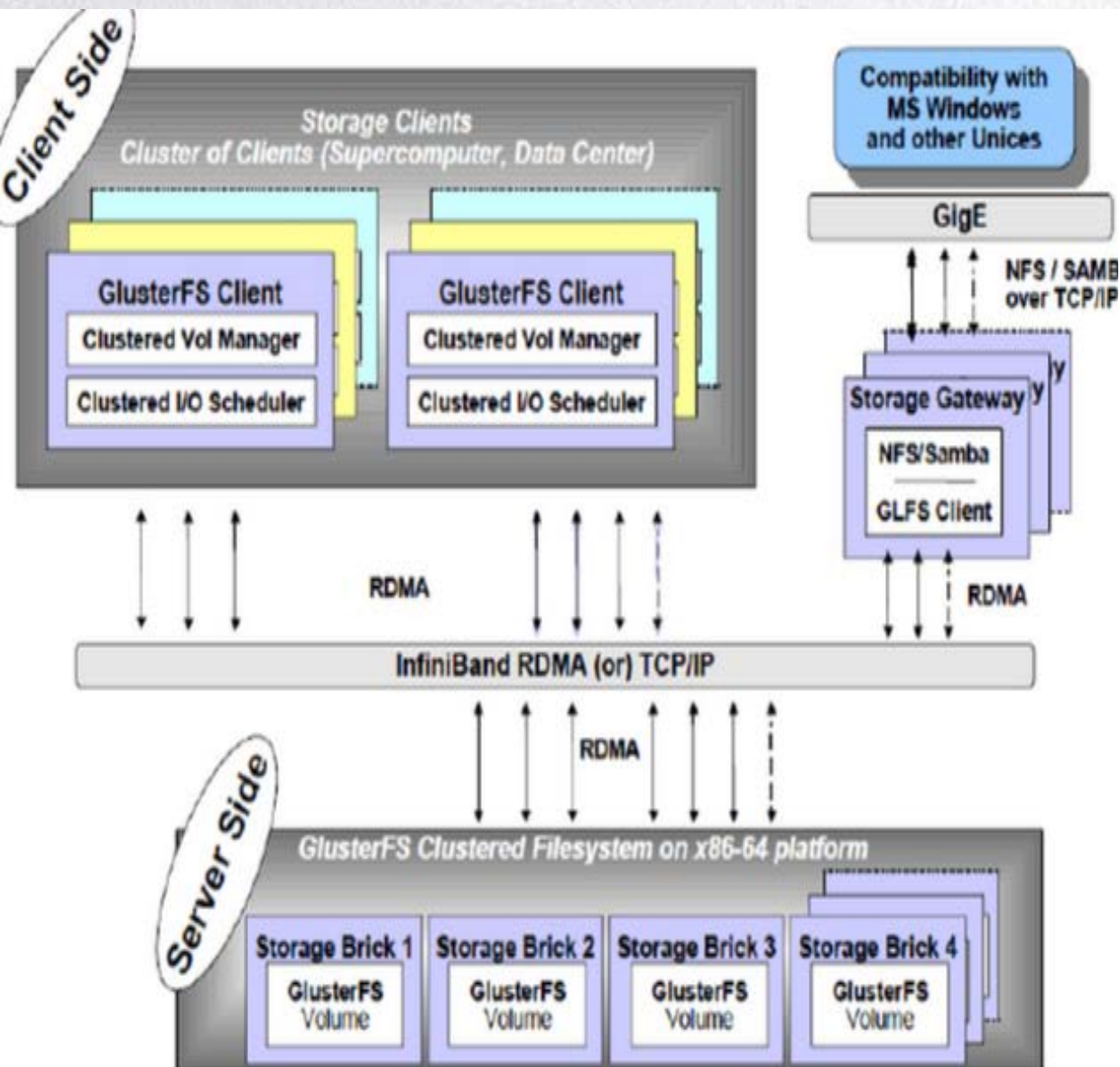
客户端通过与OSD (Object Storage Device) 的直接通讯实现文件操作。在打开一个文件时，客户端会向MDS (Metadata storage) 发送一个请求。MDS把请求的文件名翻译成文件节点 (inode) ，并获得节点号、访问模式、大小以及文件的其他元数据。如果文件存在并且客户端可以获得操作权，则MDS向客户端返回上述文件信息并且赋予客户端操作权。



相对于面向**离线批处理**的HDFS来说，Ceph更偏向于成为一种高性能、高可靠、高扩展性的**实时**分布式存储系统，其对于写入操作特别是随机写入的支持要更好。

GlusterFS是Scale-Out存储解决方案Gluster的核心，它是一个开源的分布式文件系统，具有强大的横向扩展能力，通过扩展能够支持数PB存储容量和处理数千客户端。

GlusterFS借助TCP/IP或InfiniBand RDMA网络将物理分布的存储资源聚集在一起，使用单一全局命名空间来管理数据。GlusterFS基于可堆叠的用户空间设计，可为各种不同的数据负载提供优异的性能。



- Storage Brick: GlusterFS中的存储单元，可以通过**主机名和目录名**来标识。
- Storage Client: 挂载了GlusterFS卷的设备
- RDMA: 远程直接内存访问，支持不通过双方的OS进行直接内存访问。
- RRDNS: round robin DNS，是一种通过DNS轮转返回不同的设备以进行**负载均衡**的方法。
- Self-heal: 用于后台运行**检测副本中文件和目录的不一致性**并解决这些不一致。

GlusterFS支持运行在任何标准IP网络上标准应用程序的标准客户端，用户可以在**全局统一的命名空间**中使用NFS/CIFS等标准协议来访问应用数据。

GlusterFS使得用户可摆脱原有的独立、高成本的**封闭存储系统**，能够利用普通廉价的存储设备来部署可集中管理、横向扩展、虚拟化的存储池，存储容量可扩展至TB/PB级。

GlusterFS由于缺乏一些关键特性，**可靠性也未经过长时间考验**，还不适合应用于需要提供 24 小时不间断服务的产品环境。目前适合应用于**大数据量的离线应用**。

05 分布式文件系统对比

特性	HDFS	Ceph	GlusterFS
元数据服务器	单个 存在单点故障风险	多个 不存在单点故障风险	无 不存在单点故障风险
POSIX兼容	不完全	兼容	兼容
配额限制	支持	支持	不详
文件分割	默认分成64MB块	采用RAID0	不支持
网络支持	仅TCP/IP	多种网络，包括 TCP/IP、Infiniband	多种网络，包括TCP/IP、 Infiniband
元数据	元数据服务器管理全量 元数据	元数据服务器管理少 量元数据	客户端管理全量元数据
商业应用	大量，国内包括中国移动、百度、网易、淘宝、腾讯、华为等（离线批量处理大数据）	非常不成熟，尚不适合生产环境（携程，联通研究院等）	测试和使用案例多为欧美，国内用户很少（适合应用于大数据量的离线应用）

PART 4 新型数据管理与查询系统

NoSQL (Not only SQL) 数据库是对于**非关系型**的一类数据库系统的统称。它针对关系型数据库在管理**键值对**、**文档**、**图**等类型数据上的不足，针对各个类型数据的存储和访问特点而专门设计的数据库管理系统。

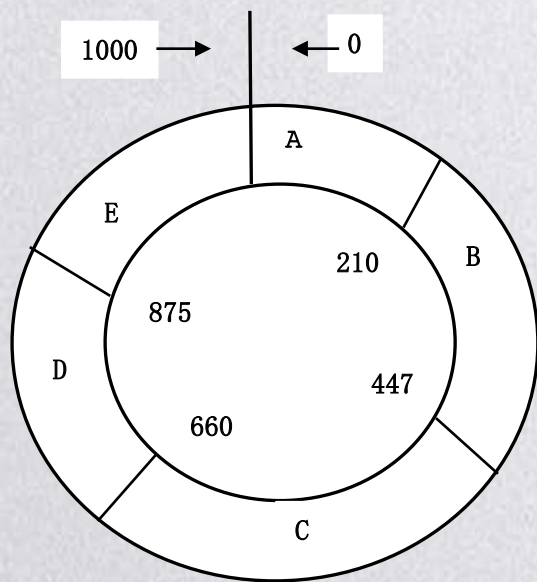
NoSQL数据库设计原则：

- 采用**横向扩展**（Scaling Out）的方式，通过对**大量节点的并行处理**，获得包括读性能和写性能在内的**极高数据处理性能和吞吐能力**。NoSQL数据库需要对**数据进行划分**，以便进行并行查询处理。
- 放弃严格的**ACID**一致性约束，采用放松的一致性约束条件，允许数据暂时出现不一致的情况，并接受最终一致性。
- 对数据进行**容错处理**，一般对数据块进行适当备份，以应对结点失败状况，保证在普适服务器组成的集群上稳定高可靠地运行。

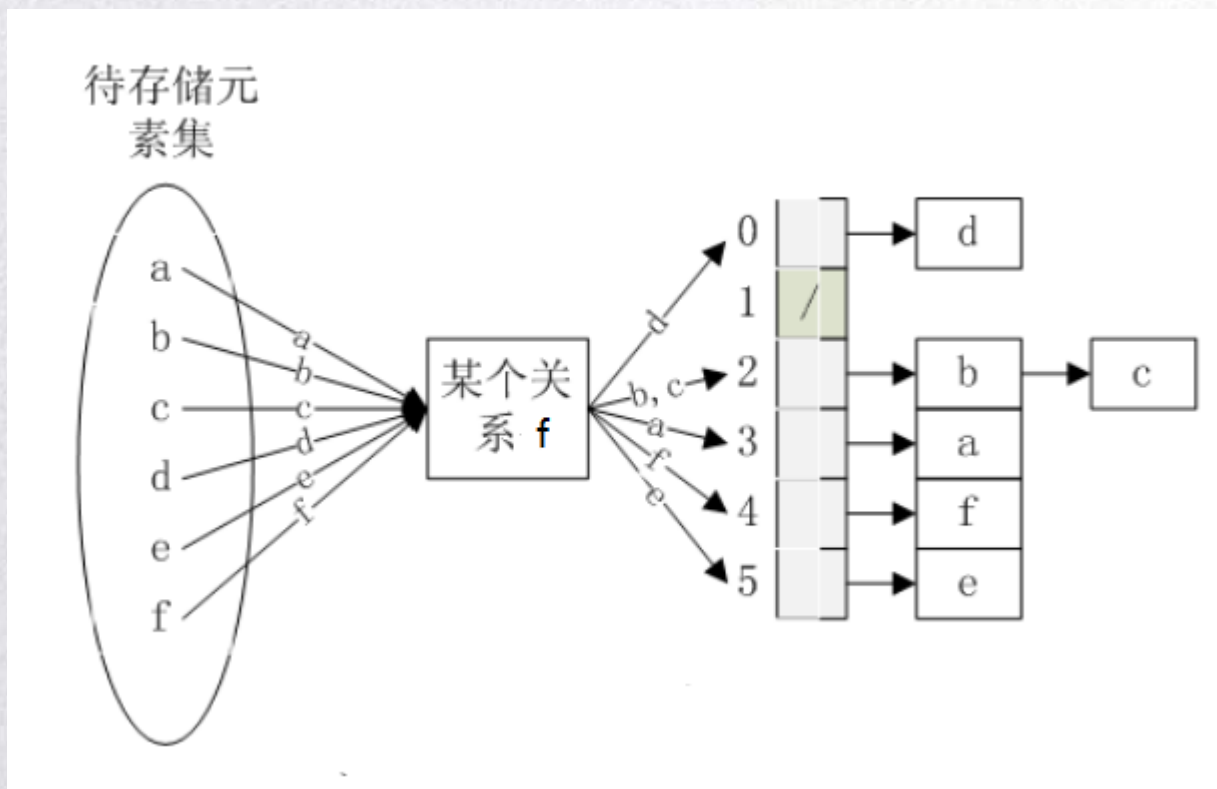
➤ 键值对 (Key-Value) 数据库

这一类数据库主要会使用到一个[哈希表](#)，这个表中有一个特定的键和一个指针指向特定的数据。Key/value模型对于IT系统来说的优势在于简单、易部署。如：

TokyoCabinet/Tyrant, **Redis**, Voldemort, Oracle BDB, **Memcached**。



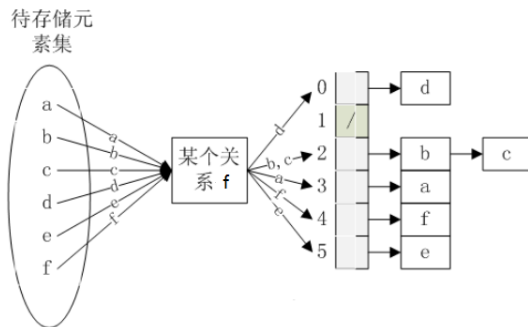
一致性哈希划分数据
 $H(\text{key}) = \text{key} \bmod L$



01

NoSQL数据库

➤ 哈希表



- **哈希表**是一种以 **键-值(key-indexed)** 存储数据的结构，只要输入待查找的键即 key，即可查找到其对应的值。
- **哈希的思路**：如果所有的键都是整数，那么就可以使用一个简单的无序数组来实现：将键作为索引，值即为其对应的值，这样就可以快速访问任意键的值。这是对于简单的键的情况，可将其扩展到可以处理更加复杂的类型的键。
- **使用哈希查找有两个步骤**:
 - 使用哈希函数将被查找的键转换为数组的索引。在理想的情况下，不同的键会被转换为不同的索引值，但是在有些情况下需要处理多个键被哈希到同一个索引值的情况。所以哈希查找的第二个步骤就是**处理冲突**
 - 处理哈希碰撞冲突：**拉链法**和**线性探测法**。

➤ 哈希函数

- 哈希查找第一步就是使用**哈希函数将键映射成索引**。
 - 如果有一个保存M数组，那么就需要一个能够将任意键转换为该数组范围内的索引（ $0 \sim M-1$ ）的哈希函数。哈希函数需要**易于计算**并且能够**均匀分布所有键**。
 - （1）使用手机号码后三位就比前三位作为key更好；
 - （2）使用身份证号码出生年月位数要比使用前几位数要更好。
- 在实际的键并不都是数字，有可能是字符串，还有可能是几个值的组合等，所以需要实现自己的哈希函数。

130xxxx1234
130xxxx2345
138xxxx4829
138xxxx2396
138xxxx8354

易重复分布太集中某几个数字 分布均匀，可用作散列地址

Array2 & M = 6 (4次冲突)

	0	1	2	3	4	5
哈希表	6、12		2、8、14		4、10	

Array2 & M = 7 (0次冲突)

	0	1	2	3	4	5	6
哈希表	14	8	2	10	4	12	6

➤ 哈希函数

➤ 正整数

- 获取正整数哈希值最常用的方法是使用**除留余数法**。即对于大小为素数M的数组，对于任意正整数k，计算k除以M的余数。M一般取素数（**为什么**）。

➤ 字符串

- 将字符串作为键的时候，可将他作为一个大的整数，采用**除留余数法**。将组成字符串的每一个字符取值然后进行哈希。

$$h = s[0] \cdot 31^{L-1} + \dots + s[L-3] \cdot 31^2 + s[L-2] \cdot 31^1 + s[L-1] \cdot 31^0$$

要获取“call”的哈希值，字符串c对应的unicode为99，a对应的unicode为97，l对应的unicode为108，所以字符串“call”的哈希值为 3045982

$$\begin{aligned} &= 99 \cdot 31^3 + 97 \cdot 31^2 + 108 \cdot 31^1 + 108 \cdot 31^0 \\ &= 108 + 31 \cdot (108 + 31 \cdot (97 + 31 \cdot (99))) \end{aligned}$$

01

NoSQL数据库

➤ 哈希函数

➤ 字符串

- 如果对每个字符取哈希值可能会比较耗时，所以可以通过间隔取N个字符来获取哈希值来节省时间，比如，可以获取每8-9个字符来获取哈希值：

如果按照每8个字符取哈希的话，就会得到一样的哈希值。

```
http://www.cs.princeton.edu/introcs/13loop/Hello.java  
http://www.cs.princeton.edu/introcs/13loop/Hello.class  
http://www.cs.princeton.edu/introcs/13loop/Hello.html  
http://www.cs.princeton.edu/introcs/12type/index.html
```

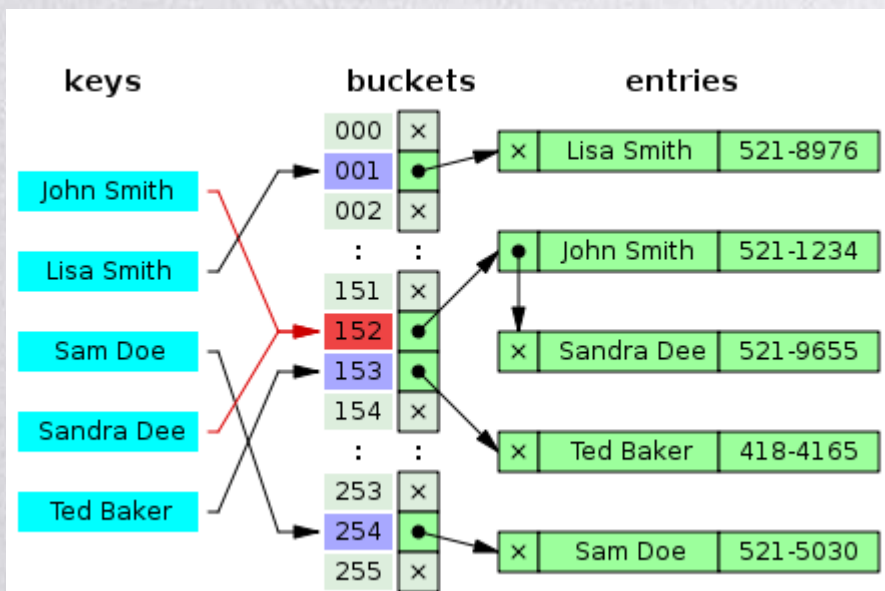
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

哈希碰撞

➤ 避免哈希冲突

拉链法

- 通过哈希函数，可以将键转换为数组的索引(0-M-1)，但是对于两个或者多个键具有相同索引值的情况，需要有一种方法来处理这种冲突。
- 将大小为M 的数组的每一个元素指向一个条链表，链表中的每一个节点都存储散列值为该索引的键值对。



基本思想就是选择足够大的M，使得所有的链表都尽可能的短小，以保证查找的效率。对采用拉链法的哈希实现的查找分为两步，首先是根据散列值找到相应的链表，然后沿着链表顺序找到相应的键。

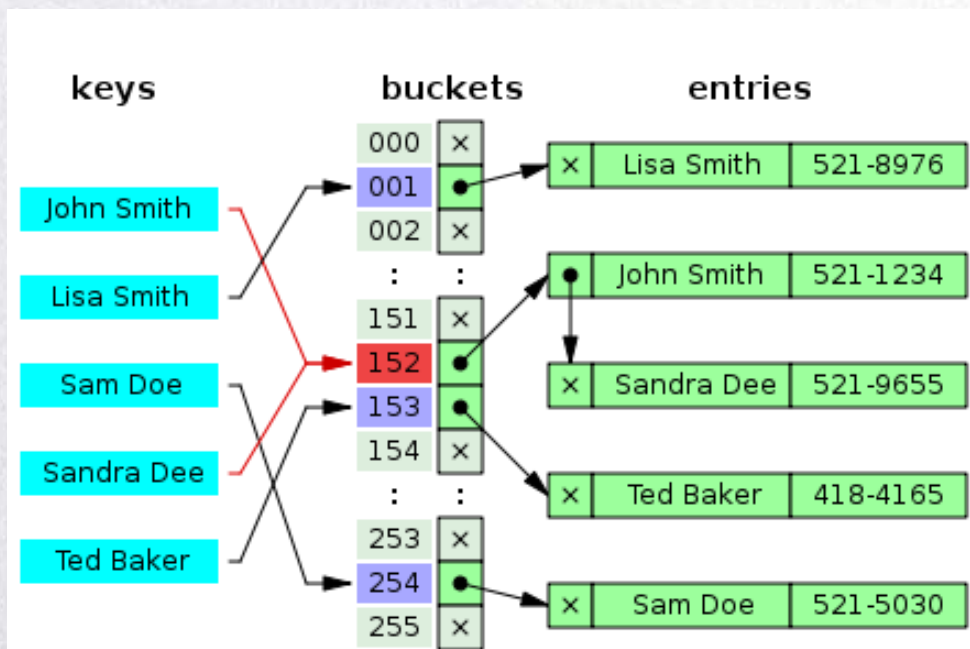
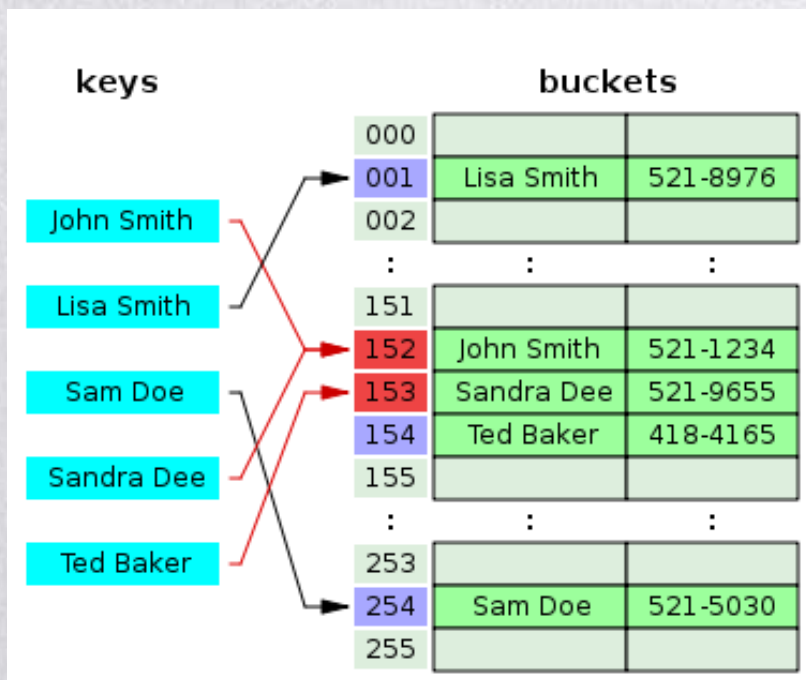
01

NoSQL数据库

➤ 避免哈希冲突

线性探测法

- 线性探测法是开放寻址法解决哈希冲突的一种方法，基本原理为，使用大小为M的数组来保存N个键值对，其中 $M > N$ ，我们需要使用数组中的空位解决碰撞冲突。



1. 对于线性表 (7 , 34 , 55 , 25 , 64 , 46 , 20 , 10 , 38, 49, 21, 25, 18, 15) 进行散列存储时，若使用 $H(K)=K\%9$ 作为散列函数，则其哈希散列表

2. 查找哈希表，解决冲突的方法包括 ()

A. 链地址法 B. 除留余数法 C. 直接地址法 D. 线性探测再散列法

3. 以下那种结构，平均来讲获取任意一个指定值最快？()

A. 二叉排序树 B. 队列 C. 栈 D. 哈希表

二叉排序树中，查找的平均时间复杂度是 $O(\log n)$ ；

对于栈和队列来说，查找就意味着把元素挨个出栈或者出队，故平均时间复杂度是 $O(n)$ ；

哈希表，直接通过关键码查找元素，平均为 $O(1)$ ；

➤ 文档数据库

文档数据库技术是以**键值对存储模型**作为基础模型的NoSQL技术。文档存储数据库以不同标准（如JSON，XML，BSON或YAML）编码的文档形式，来存储**半结构化数据**。每个文档都由唯一的键key或ID来标识。在将文档存储到数据库中之前，无需为文档定义任何模式。



➤ 文档数据库

MongoDB是一款分布式文档数据库，它为**大数据量**、**高度并发访问**、**弱一致性要求**的应用而设计。Mongodb具有**高扩展性**，在高负载的情况下，可以通过添加更多的节点，保证系统的查询性能和吞吐能力。

MongoDB数据库支持增加、删除、修改、简单查询等主要的数据库操作以及动态查询，并且可以在复杂属性上建立**索引**，当查询包含该属性的条件时，可以利用索引获得更高的查询性能。

使用文档数据库来存储商品记录

ID	Document
34fd459fs523f3f34d43325	{ "标题" : "iPhone 8 Plus" "特点" : ["屏幕尺寸" : "5.5英寸" "后置摄像头" : "1200万" "存储容量" : "64GB " "运行内存" : "6GB " "操作系统" : "iOS "] "价格" : 5999元 }

01 NoSQL数据库

➤ 列族（Column Family）数据库

通常是用来应对**分布式存储海量数据**。数据存储的基本单位是一个**列**，它具有一个名称和一个值。由列的集合组成的每一行，通过行-键标识来标示，列组合在一起成为列族。与关系数据库不同，列族数据库不需要**在每行中都有固定的模式和固定数量的列**。**列族数据库还可以存储非结构数据**。

HBase表存储结构

HBase表		
RowKey	ColumnFamily-1	ColumnFamily-2
记录1	列1.....列n	列1，列2，列3
记录2	列1，列2	
记录3	列1.....列5	列1



➤ 图数据库

图数据库是用于专门存储具有**节点和边的图结构数据**的一类数据库，并以**节点和边作为基本数据模型**。节点可以代表数据模型中的重要的实体或信息条目，节点之间的关系以边的形式表示。

六度分隔 (Six Degrees of Separation) 理论

“你和任何一个陌生人之间所间隔的人不会超五个，也就是说，最多通过六个人你就能够认识任何一个陌生人。”



某交通保险欺诈关系图

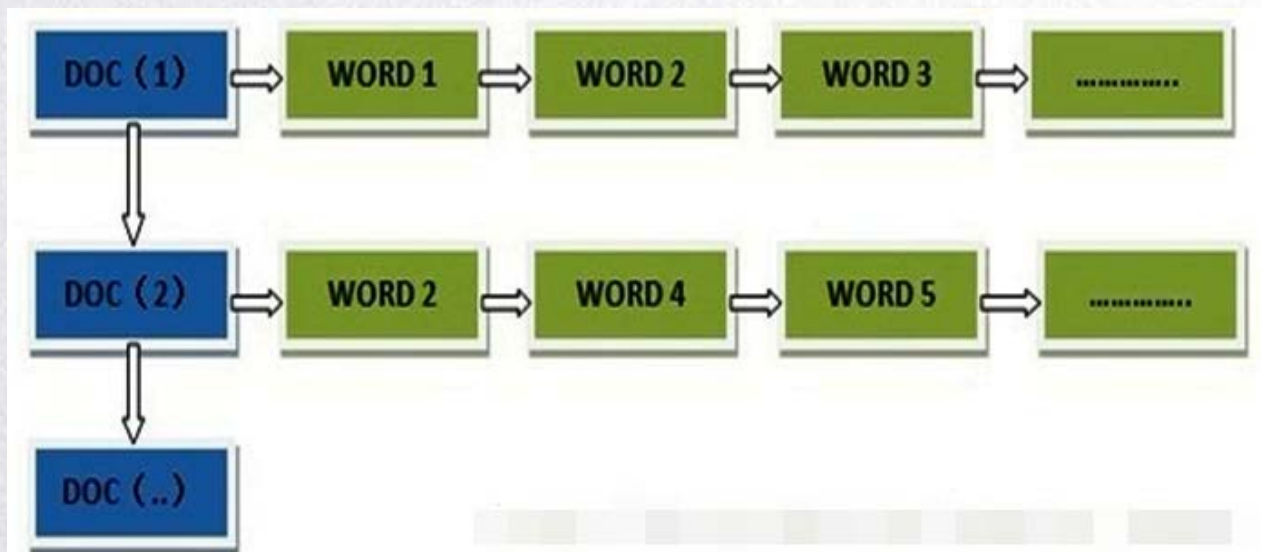
分类	相关产品	典型应用场景	数据模型	优点	缺点
键值对数据库	Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB	内容缓存, 主要用于处理大量数据的高访问负载	Key 指向 Value 的键值对, 通常用hash table 来实现	查找速度快	数据无结构化
列族数据库	Cassandra, HBase, Riak	分布式的文件系统	以列簇式存储, 将同一列数据存在一起	查找速度快, 可扩展性强, 更容易进行分布式扩展	功能相对局限
文档数据库	CouchDB, MongoDB	Web应用 (与 Key-Value类似, Value是半结构化)	Key-Value对应的键值对, Value为半结构化数据	数据结构要求不严格, 表结构可变	查询性能不高, 缺乏统一的查询语法
图数据库	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	需对整个图做计算, 不容易做分布式集群方案

互联网公司最先遇到大数据难题，需要为海量互联网网页构建**倒排列表**。2004年，Google公司提出**MapReduce**技术，作为面向大数据分析和处理的**并行计算模型**，引起了工业界和学术界的广泛关注。**Hadoop**技术很快也影响了数据库研究领域，有面向简单的键值对读写事务型负载的NoSQL系统（如**HBase**等），也有面向数据分析任务的**Hive系统**。

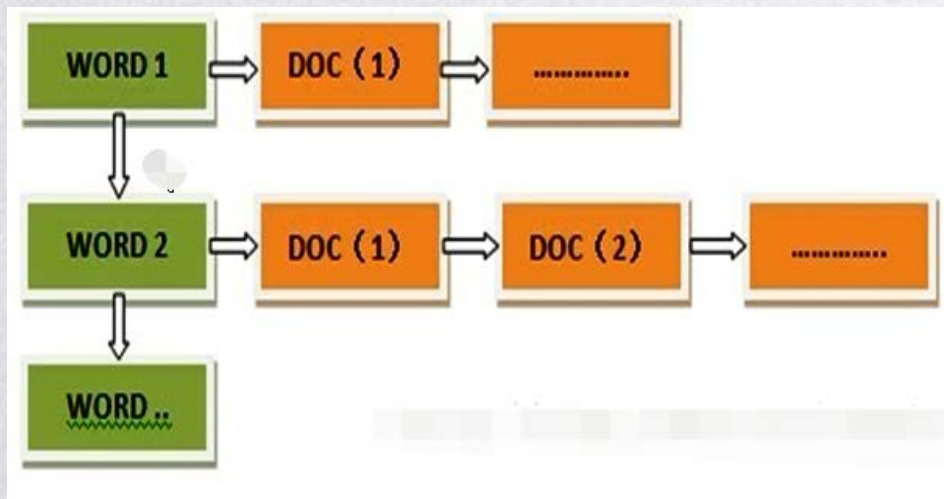
来自互联网领域或者其他领域很多大数据创新公司，并没有止步于**Hive**。最近五六年间做出了很多努力，开发了多个**SQL on Hadoop系统**，以提升这些系统的性能。这些系统借鉴了20世纪90年代以来在并行数据库方面所积累的一些先进技术，大幅度提升了SQL on Hadoop系统的性能。

02 SQL on Hadoop系统

正向索引



倒排列表



02 SQL on Hadoop系统

倒排列表

文档编号	文档内容
1	谷歌地图之父跳槽Facebook
2	谷歌地图之父加盟Facebook
3	谷歌地图创始人拉斯离开谷歌加盟Facebook
4	谷歌地图之父跳槽Facebook与Wave项目取消有关
5	谷歌地图之父拉斯加盟社交网站Facebook



单词ID	单词	倒排列表 (DocID)
1	谷歌	1,2,3,4,5
2	地图	1,2,3,4,5
3	之父	1,2,4,5
4	跳槽	1,4
5	Facebook	1,2,3,4,5
6	加盟	2,3,5
7	创始人	3
8	拉斯	3,5
9	离开	3
10	与	4
11	Wave	4
12	项目	4
13	取消	4
14	有关	4
15	社交	5
16	网站	5

带有单词频率信息的倒排索引



单词ID	单词	倒排列表 (DocID)
1	谷歌	1,2,3,4,5
2	地图	1,2,3,4,5
3	之父	1,2,4,5
4	跳槽	1,4
5	Facebook	1,2,3,4,5
6	加盟	2,3,5
7	创始人	3
8	拉斯	3,5
9	离开	3
10	与	4
11	Wave	4
12	项目	4
13	取消	4
14	有关	4
15	社交	5
16	网站	5

含某个单词

实际搜

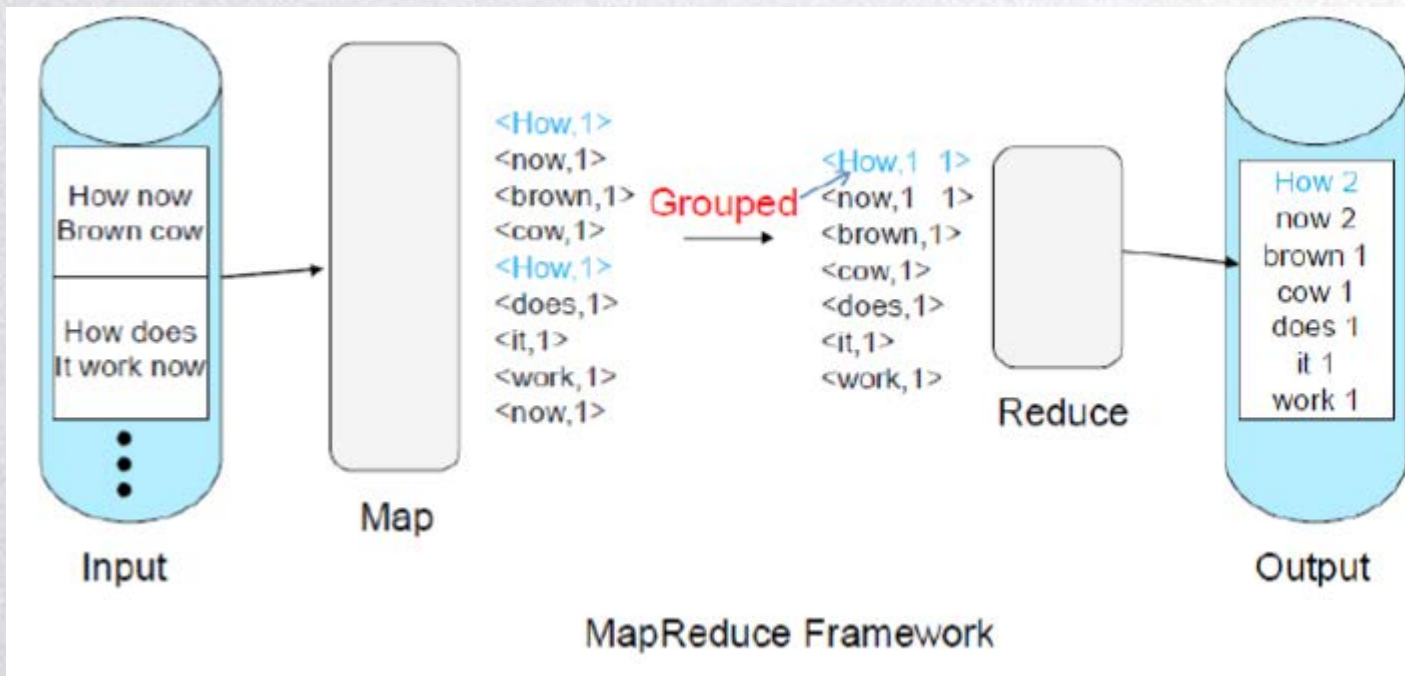
文档编号	文档内容
1	谷歌地图之父跳槽Facebook
2	谷歌地图之父加盟Facebook
3	谷歌地图创始人拉斯离开谷歌加盟Facebook
4	谷歌地图之父跳槽Facebook与Wave项目取消有关
5	谷歌地图之父拉斯加盟社交网站Facebook

倒排列表 (DocID;TF;<POS>)

1	谷歌	5	(1;1;<1>), (2;1;<1>), (3;2;<1;6>), (4;1;<1>), (5;1;<1>)
2	地图	5	(1;1;<2>), (2;1;<2>), (3;1;<2>), (4;1;<2>), (5;1;<2>)
3	之父	4	<1;1;<3>), (2;1;<3>), (4;1;<3>), (5;1;<3>)
4	跳槽	2	(1;1;<4>), (4;1;<4>)
5	Facebook	5	(1;1;<5>), (2;1;<5>), (3;1;<8>), (4;1;<5>), (5;1;<8>)
6	加盟	3	(2;1;<4>), (3;1;<7>), (5;1;<5>)
7	创始人	1	(3;1;<3>)
8	拉斯	2	(3;1;<4>), (5;1;<4>)
9	离开	1	(3;1;<5>)
10	与	1	(4;1;<6>)

大数据计算：应对规模化

增大m，提高f：以MapReduce为例



$$\text{Speedup}_{\text{fixed-time}} = \frac{\text{Sequential Time of Solving Sacred Workload}}{\text{Paralled Time of Solving Scaled Workload}} = (1-f) + mf$$

PART 5 习题和参考文献

知识点：

- 知识点1：数据管理技术的发展历程
- 知识点2：关系数据模型与关系数据库技术
- 知识点3：分布式文件系统
- 知识点4：NoSQL数据库
- 知识点5：Sql on Hadoop技术

课程重点、难点：

- 重点1：数据管理技术的内涵与边界
- 重点2：关系数据模型
- 难点1：关系数据库技术与文件系统技术的异同
- 难点2：NoSQL数据库

- 3.1 数据管理技术的发展历程是怎样的？
- 3.2 关系数据库的特点是什么？
- 3.3 NoSQL数据库的特点是什么？
- 3.4 Sql-on-Hadoop技术与数据库技术的差异在哪？
- 3.5 请列举典型的分布式文件系统，并简要描述。
- 3.6 请针对学生课程成绩查询的场景，设计主要的关系数据表结构，并描述对应的SQL语句。

王珊，萨师煊.数据库系统概论（第5版）》.高等教育出版社，2014.

[美] Gavin Powell著，[译]沈洁 王洪波 赵恒.数据库设计入门经典[Beginning Database Design].清华大学出版社,2007.

[美]Dan Sullivan著，[译]爱飞翔.NoSQL实践指南：基本原则、设计准则及实用技巧.机械工业出版社,2016.

The background features a collage of blue geometric shapes, including squares and diamonds, some of which are semi-transparent, revealing a city skyline (likely Chicago) and a body of water. A large, solid blue triangle with a white dashed line is positioned in the bottom right corner.

THANKS
