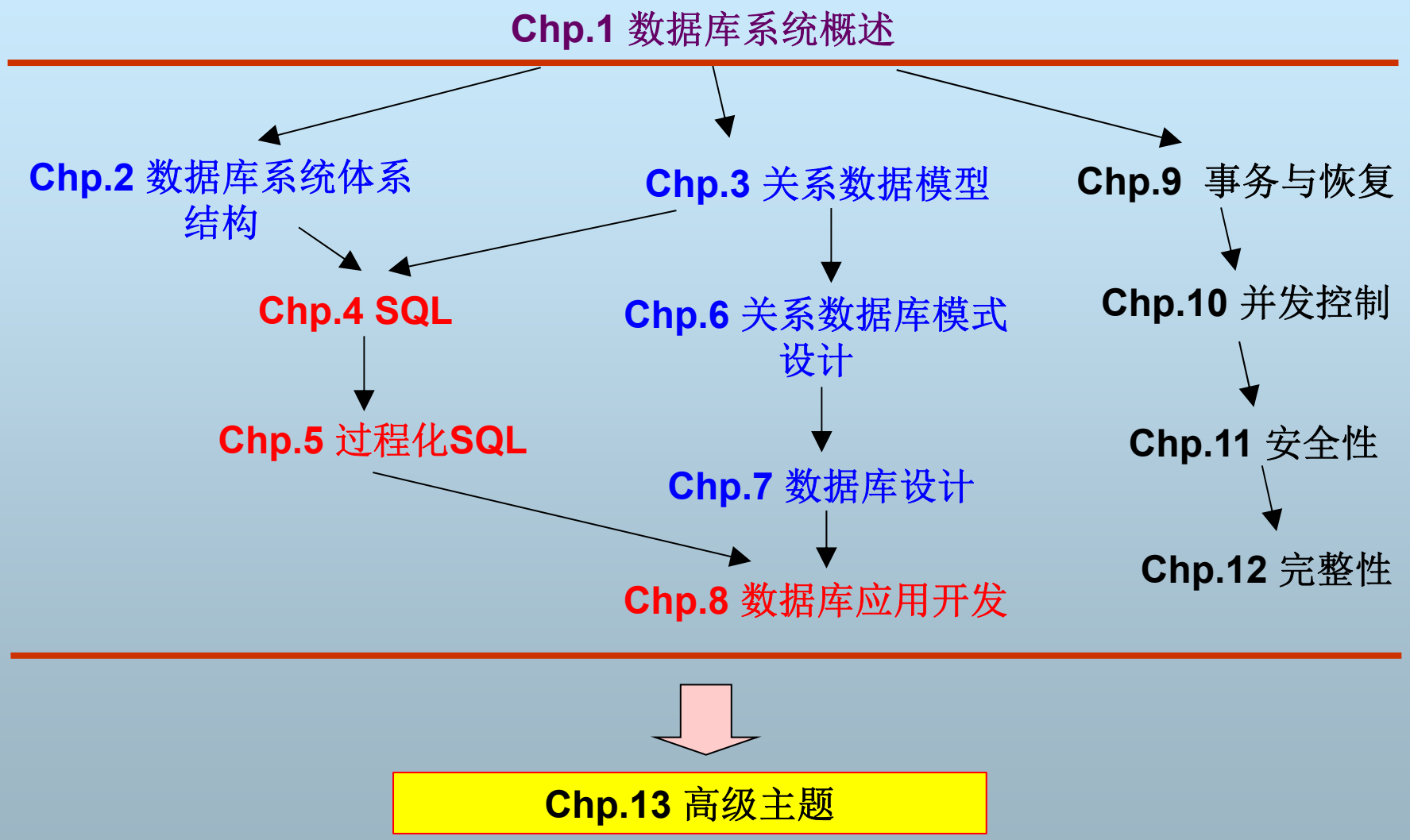
第13章 高级主题

金培权（jpq@ustc.edu.cn）

课程知识结构


Chp.1 数据库系统概述
Chp.2 数据库系统体系结构
Chp.3 关系数据模型
Chp.9 事务与恢复
Chp.4 SQL
Chp.6 关系数据库模式设计
Chp.10 并发控制
Chp.5 过程化SQL
Chp.7 数据库设计
Chp.11 安全性
Chp.8 数据库应用开发
Chp.12 完整性
Chp.13 高级主题

主要内容

- 面向对象数据库
- 对象关系数据库
- **NoSQL数据库**

面向对象数据库和对象关系数据库

金培权（jpq@ustc.edu.cn）

一、OODB的产生与发展

- 传统数据库应用的特征
  - 结构统一：数据格式化，结构相似
  - 面向记录
  - 数据小：记录一般都比较短
  - 原子字段：字段内部是无结构的，符合1NF
- 新的应用不具备传统特征
  - CAD、GIS、OA、多媒体应用、超媒体
  - ……

一、OODB的产生与发展

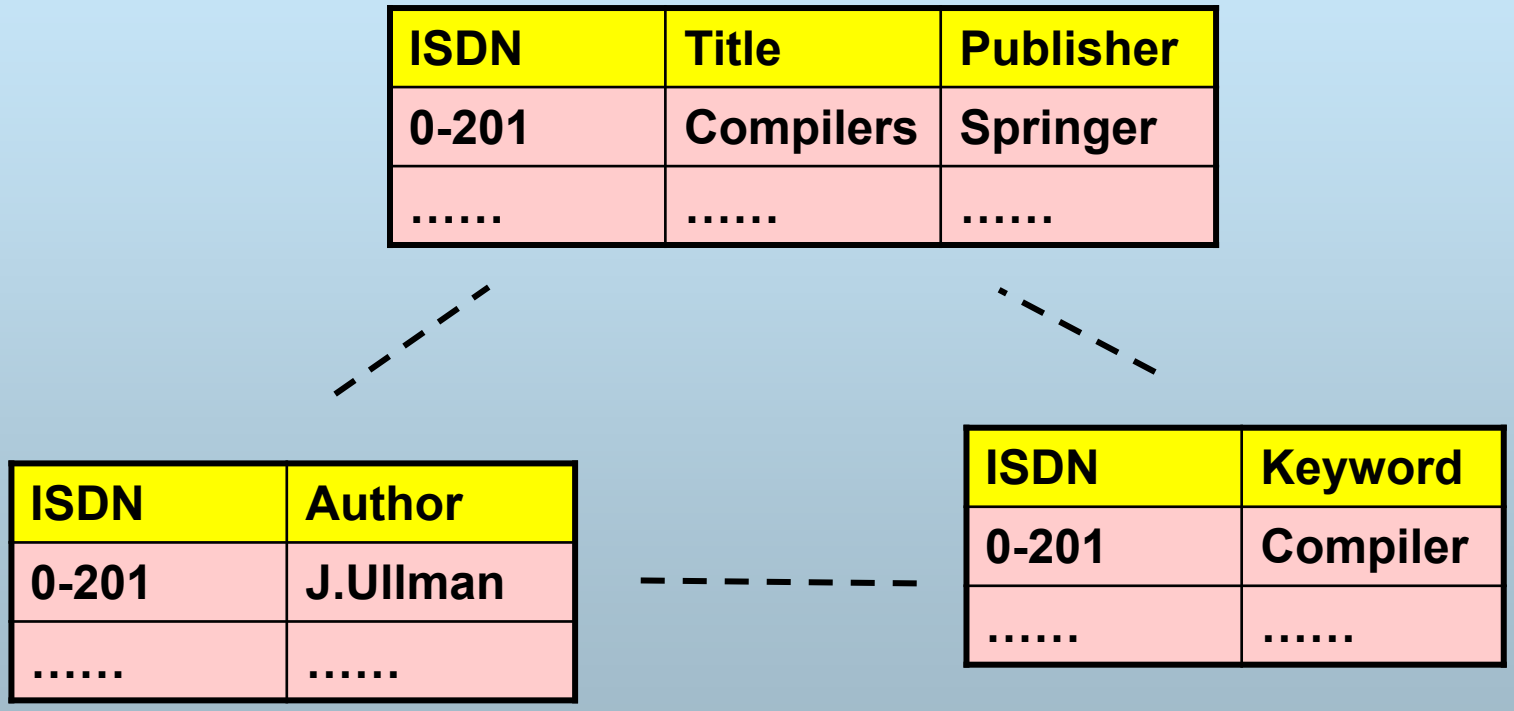
- 关系数据库技术缺乏处理复杂应用的能力

ISDN	Title	Author	Keyword	Publisher
0-201	Compilers	J.Ullman	Compiler	Springer
0-201	Compilers	J.Ullman	Grammar	Springer
0-201	Compilers	A.Aho	Compiler	Springer
0-201	Compilers	A.Aho	Compiler	Springer
0-201	Compilers	J.Hopcroft	Compiler	Springer
0-201	Compilers	J.Hopcroft	Compiler	Springer

问题：由于不能有集合值而导致了冗余设计

一、OODB的产生与发展

- 规范化设计结果



ISDN	Title	Publisher
0-201	Compilers	Springer
……	……	……

ISDN	Author
0-201	J.Ullman
……	……

ISDN	Keyword
0-201	Compiler
……	……

一、OODB的产生与发展

- 问题
  - Book在现实世界中是一个独立对象，而在数据库中被认为地分割成了几个关系

  - 不能表达顺序关系（作者）
  - SQL可以执行分组操作，但不能返回一组结果
    - 例如“查询作者集合及其所写作的书列表”

一、OODB的产生与发展

- 一种自然的解决方法

ISDN	Title	Author	Keyword	Publisher
0-201	Compilers	{J.Ullman, A.Aho, J.Hopcroft}	{Compiler, Grammar}	Springer

- 嵌套关系(Nested Relation)
  - 关系中的值可以是一个集合或列表
- 我们所需要的是一个允许我们在关系中自由使用集合和列表的类型系统

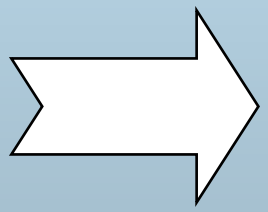
一、OODB的产生与发展

- **OO技术**
  - 提供了复杂类型系统，支持集合、列表等类型

OO概念

数据库系统概念

新的应用需求



面向对象数据库系统：

可以在DBMS中支持复杂数据

二、OODB的实现方法

- 持久化OO语言
  - 与OO语言相关，增加持久化支持
  - 一般只支持C++和Java，早期还支持SmallTalk

例：Versant for Java中持久化对象的例子

```
Person per = new Person(name, age);
……
session.makePersistent(per);
```

- 主要的问题
  - 只支持C++和Java
  - 不支持SQL标准

三、ORDB

- ORDB = RDB + OO
  - 支持对象概念的关系数据库系统
- 关系表的列可以是新的抽象数据类型(ADT)
- 用户可以增加新的ADT以及定义ADT上的操作
- 支持引用类型、继承
- 支持SQL语言

1、ORDB的ADT扩展例子

Employee(name: char(10), salary int, addr: Address)

- **Address是扩充的ADT**

```
Create Type Address (
    street varchar(30),
    city     varchar(30),
    house_number  varchar(20)
)
```

- **Select name, addr.city From Employee**

2、ORDB支持的ADT类型

- Create Type可以创建的ADT（SQL标准）
  - ROW：行类型，类似C结构类型
  - SET：集合类型
  - MultiSET：多值集合（允许重复值）
  - LIST：有序列表
  - REF：引用类型，指向一个复杂Type的指针

3、ORDB中对象查询例子

- **ORDB中对象查询例子**
  - **Country(name:char(30), boundary: polygon)**
  - **查询例子“查询中国的邻国名称”**

```
Select C.name
From Country C, Country D
Where D.name='China' and Meet(C.boundary, D.boundary)
```

使用了扩充的ADT操作Meet，用于回答空间连接查询

NoSQL数据库

金培权（jpq@ustc.edu.cn）

一、NoSQL简介

- **Definition (from http://nosql-database.org)**
  - **Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontal scalable.**
  - **The original intention has been modern Web-scale databases. The movement began early 2009 and is growing rapidly. Often more characteristics apply as: schema-free, easy replication support, simple API, eventually consistent /BASE (not ACID), a huge data amount, and more.**
  - **So the misleading term *"nosql"* (the community now translates it mostly with "not only sql") should be seen as an alias to something like the definition above.**

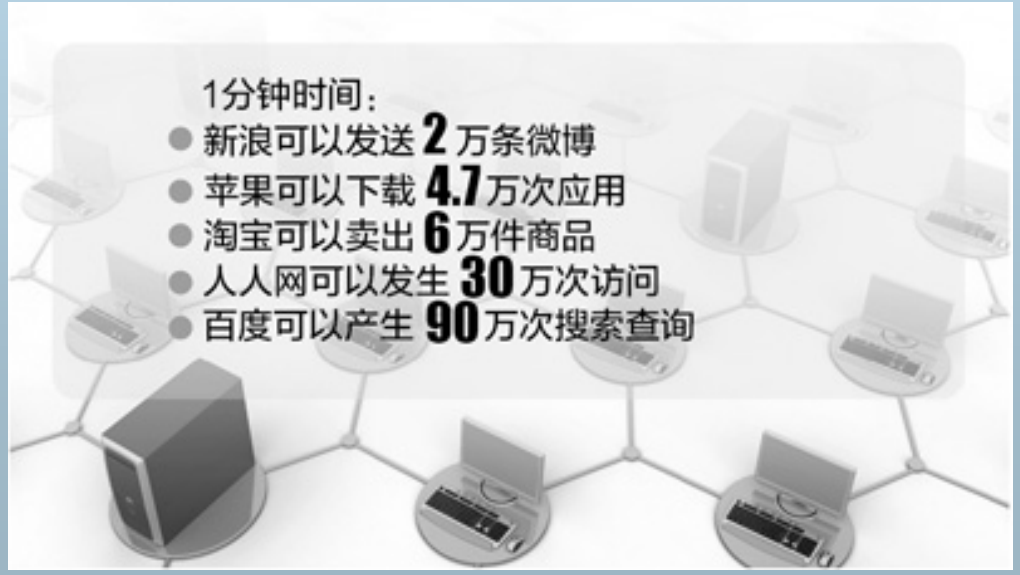
1、NoSQL特点

- **NoSQL特点：**
  - **Non relational**
  - **Scalability**
  - **No pre-defined schema**
  - **CAP not ACID**

2、NoSQL兴起的原因

（1）RDBMS无法满足Web 2.0的需求：

- 无法满足海量数据的管理需求
- 无法满足数据高并发的需求
- 无法满足高可扩展性和高可用性的需求



2、NoSQL兴起的原因

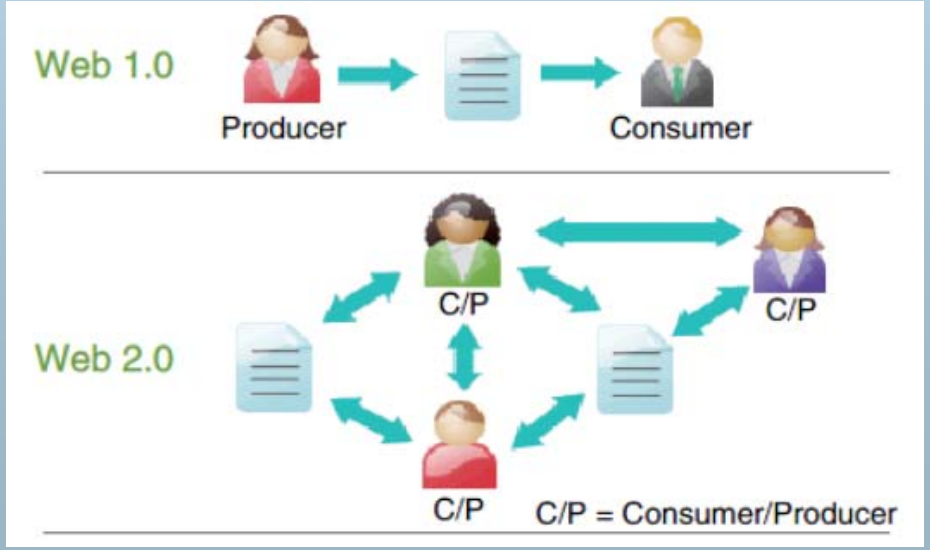
（2）“One size fits all”模式很难适用于截然不同的业务场景

- 关系模型作为统一的数据模型既被用于数据分析（OLAP），也被用于在线业务（OLTP）。但这两者一个强调高吞吐，一个强调低延时，已经演化出完全不同的架构。用同一套模型来抽象显然是不合适的
  - Hadoop就是针对数据分析
  - MongoDB、Redis等是针对在线业务，两者都抛弃了关系模型

2、NoSQL兴起的原因

（3）关系数据库的关键特性在Web 2.0时代出现变化

- **Web 2.0网站系统通常不要求严格的数据库事务**
- **Web 2.0并不要求严格的读写实时性**
- **Web 2.0通常不包含大量复杂的SQL查询（去结构化，存储空间换取更好的查询性能）**



二、NoSQL vs. RDBMS

- **RDBMS**
  - 优势：以完善的关系代数理论作为基础，有严格的标准，支持事务ACID，提供严格的数据一致性，借助索引机制可以实现高效的查询，技术成熟，有专业公司的技术支持
  - 劣势：可扩展性较差，无法较好支持海量数据存储，采用固定的数据库模式，无法较好支持Web 2.0应用，事务机制影响系统的整体性能等
- **NoSQL**
  - 优势：可以支持超大规模数据存储，数据分布和复制容易，灵活的数据模型可以很好地支持Web 2.0应用，具有强大的横向扩展能力等
  - 劣势：缺乏数学理论基础，复杂查询性能不高，大都不能实现事务强一致性，很难实现数据完整性，技术尚不成熟，缺乏专业团队的技术支持，维护较困难，目前处于百花齐放的状态，用户难以选择（120+产品 listed in http://nosql-database.org ）等

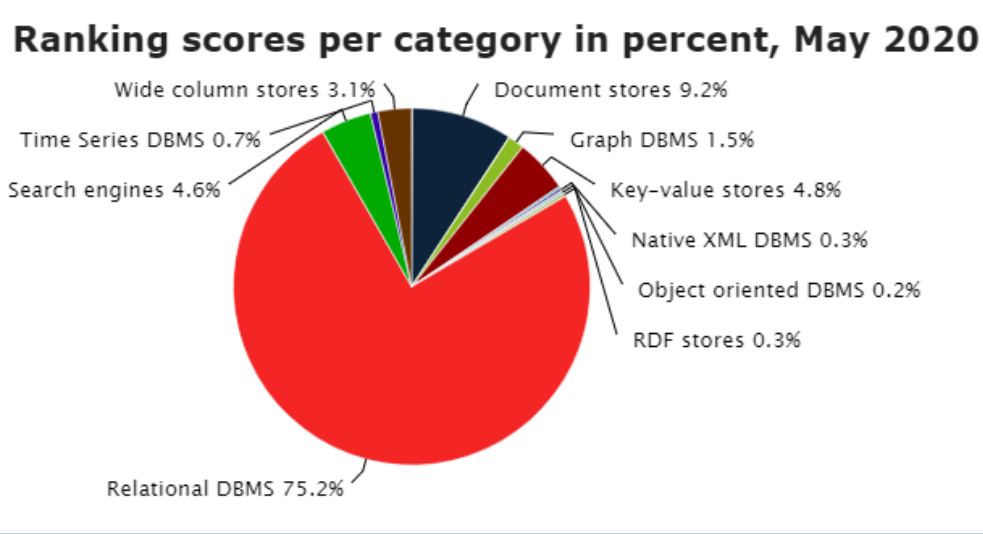
二、NoSQL vs. RDBMS

- RDBSM和NoSQL各有优缺点，彼此无法取代
- RDBMS应用场景
  - 电信、银行等领域的关键业务系统，需要保证强事务一致性
- NoSQL数据库应用场景
  - 互联网企业、传统企业的非关键业务（比如数据分析）
- 采用混合架构
  - 案例：亚马逊公司就使用不同类型的数据库来支撑它的电子商务应用
    - 对于“购物篮”这种临时性数据，采用键值存储会更加高效
    - 当前的产品和订单信息则适合存放在关系数据库中
    - 大量的历史订单信息则适合保存在类似MongoDB的文档数据库中

二、NoSQL vs. RDBMS

- 流行度ranking（2020.5）

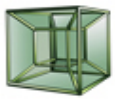
db-engines.com/en/ranking



357 systems in ranking, May 2020

Rank May 2020	Rank Apr 2020	Rank May 2019	DBMS	Database Model	Score May 2020	Score Apr 2020	Score May 2019
1.	1.	1.	Oracle	Relational, Multi-model	1345.44	+0.02	+59.89
2.	2.	2.	MySQL	Relational, Multi-model	1282.64	+14.29	+63.67
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1078.30	-5.12	+6.12
4.	4.	4.	PostgreSQL	Relational, Multi-model	514.80	+4.95	+35.91
5.	5.	5.	MongoDB	Document, Multi-model	438.99	+0.57	+30.92
6.	6.	6.	IBM Db2	Relational, Multi-model	162.64	-2.99	-11.80
7.	7.	7.	Elasticsearch	Search engine, Multi-model	149.13	+0.22	+0.51
8.	8.	8.	Redis	Key-value, Multi-model	143.48	-1.33	-4.93
9.	9.	↑ 11.	SQLite	Relational	123.03	+0.84	+0.14
10.	10.	↓ 9.	Microsoft Access	Relational	119.90	-2.02	-23.88

三、NoSQL的主要类型

文档数据库	图数据库
 Couchbase MarkLogic mongoDB	 InfiniteGraph The Distributed Graph Database
键值数据库	**列存储数据库**
 redis amazon DynamoDB  AEROSPIKE riak	 accumulo HYPERTABLE INC  Cassandra APACHE HBASE Amazon SimpleDB

1、键值数据库 （Key-Value Store）

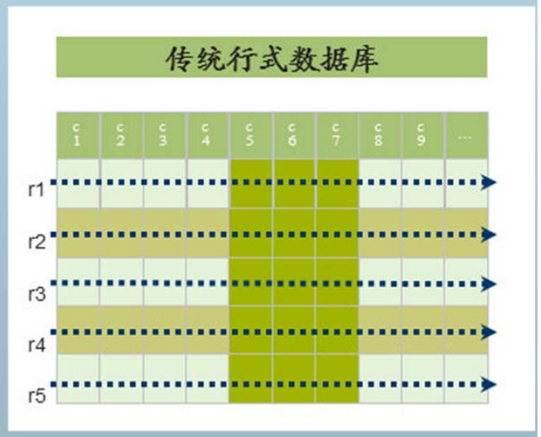
相关产品	Redis、LevelDB、SimpleDB、RocksDB、Memcached等
数据模型	• 键/值对 • 键是一个字符串对象 • 值可以是任意类型的数据，比如整型、字符型、数组、列表、集合等
优点	扩展性好，灵活性好，大量写操作时性能高
缺点	无法存储结构化信息，条件查询效率较低
使用者	百度云数据库（Redis）、GitHub（Riak）、BestBuy（Riak）、Twitter（Redis和Memcached）、StackOverFlow（Redis）、Youtube（Memcached）、Wikipedia（Memcached）
不适用情形	• 不是通过键而是通过值来查：键值数据库根本没有通过值查询的途径 • 需要存储数据之间的关系：在键值数据库中，不能通过两个或两个以上的键来关联数据 • 需要事务的支持：在一些键值数据库中，产生故障时，不可以回滚

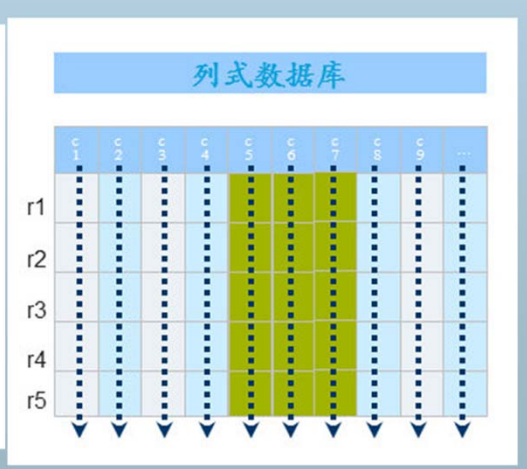
Key_1	Value _1
Key_2	Value _2
Key_3	Value _1
Key_4	Value _3
Key_5	Value _2
Key_6	Value _1
Key_7	Value _4
Key_8	Value _3

键值数据库

2、列存储数据库（Column Store）

相关产品	BigTable、HBase、Cassandra、HadoopDB、GreenPlum、PNUTS
数据模型	列族 （Column Family）
优点	查找速度快，可扩展性强，容易进行分布式扩展，复杂性低
缺点	功能较少，大都不支持强事务一致性
使用者	EBay（Cassandra）、NASA（Cassandra）、Twitter（Cassandra and HBase）、Facebook（HBase）、Yahoo!（HBase）
不适用情形	需要ACID事务支持的情形，Cassandra等产品就不适用





3、文档数据库（Document Store）

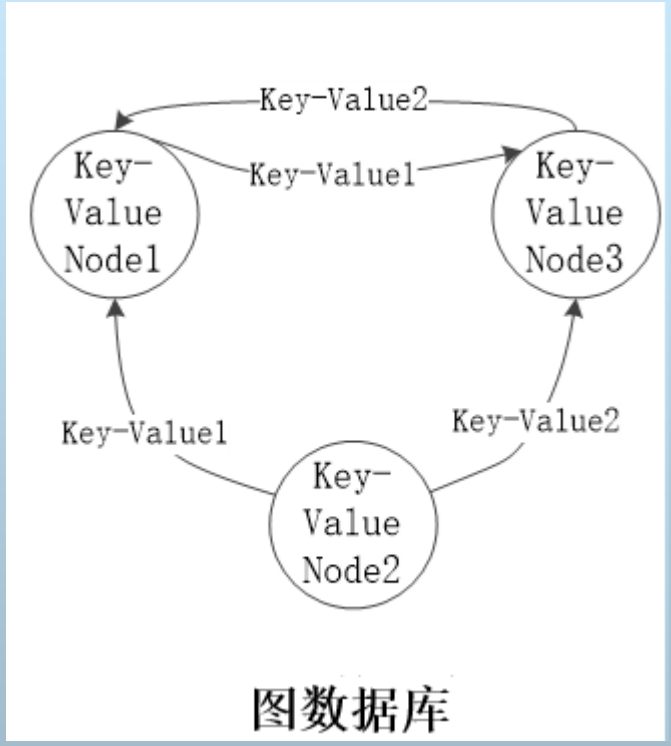
相关产品	MongoDB、CouchDB、Terrastore、ThruDB、RavenDB、SisoDB、RaptorDB、CloudKit、Perservere、Jackrabbit、**SequoiaDB** *（rank 34 in Document Stores，247 in all DBMSs）*
数据模型	• 键/值 • 值（value）是版本化的文档
优点	• 性能好（高并发），灵活性高，复杂性低，数据结构灵活 • 提供嵌入式文档功能，将经常查询的数据存储在同一个文档中 • 既可以根据键来构建索引，也可以根据内容构建索引
缺点	缺乏统一的查询语法
使用者	百度云数据库（MongoDB）、SAP（MongoDB）、Codecademy（MongoDB）、Foursquare（MongoDB）、NBC News（RavenDB）
不适用情形	在不同的文档上添加事务。文档数据库并不支持文档间的事务，如果对这方面有需求则不应该选用这个解决方案

RDBMS		MongoDB
Database	⇨	Database
Table, View	⇨	Collection
Row	⇨	Document (BSON)
Column	⇨	Field
Index	⇨	Index
Join	⇨	Embedded Document
Foreign Key	⇨	Reference
Partition	⇨	Shard (数据水平切分到不同物理节点)

```
{ "_id" : "37010"
    "city" :        "ADAMS",
    "pop" :         2660,
    "state" :       "TN",
    "councilman" : { name: "John Smith"
                     address: "13 Scenic Way"
                   }
}
```

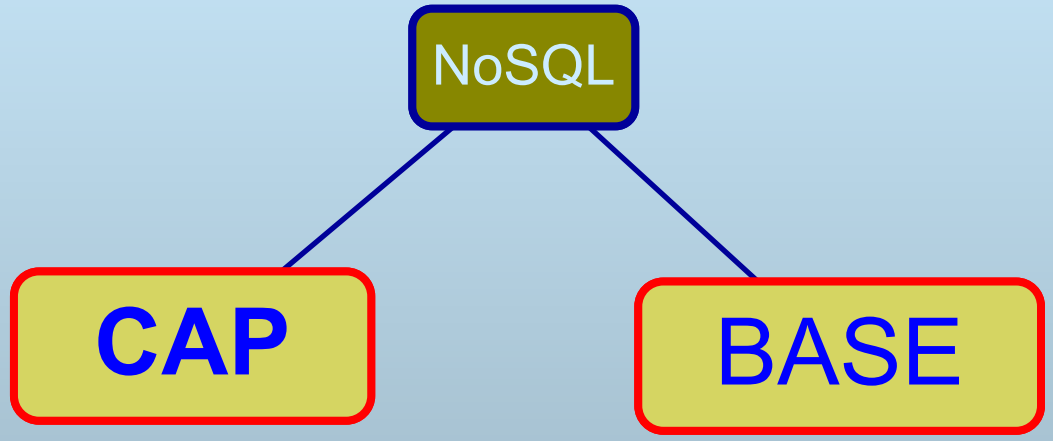
4、图数据库（Graph Store）

相关产品	Neo4J、OrientDB、InfoGrid、InfiniteGraph、GraphDB
数据模型	图结构
典型应用	专门用于处理具有高度相互关联关系的数据，比较适合于社交网络、模式识别、依赖分析、推荐系统以及路径寻找等问题
优点	灵活性高，支持复杂的图算法，可用于构建复杂的关系图谱
缺点	复杂性高，只能支持一定的数据规模
使用者	Adobe（Neo4J）、Cisco（Neo4J）、T-Mobile（Neo4J）



图数据库

四、NoSQL的分布式系统基础



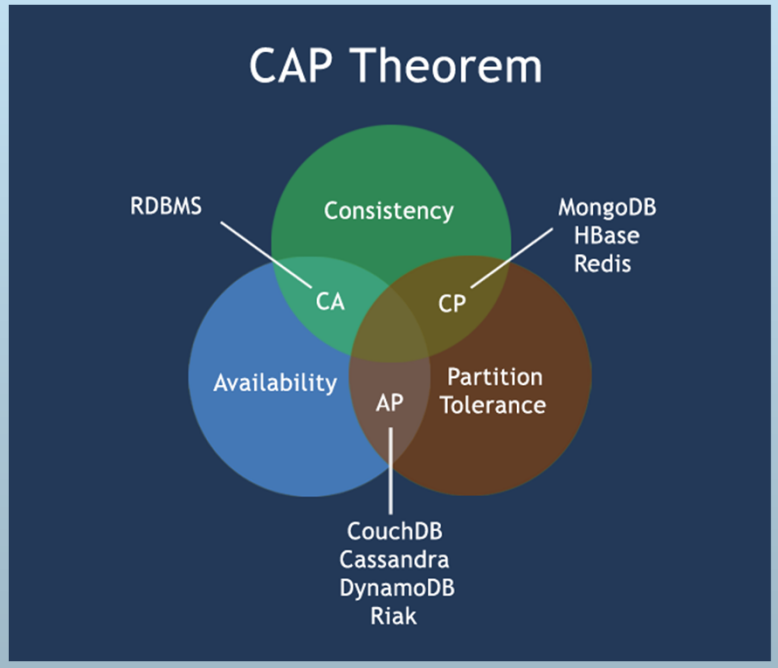
Note：基本都是分布式系统中的技术，跟数据库系统关系不大

1、CAP

- **C（Consistency）：一致性——** ***all nodes see the same data at the same time***
  - 是指任何一个读操作总是能够读到之前完成的写操作的结果，也就是在分布式环境中，多点的数据是一致的，或者说，所有节点在同一时间具有相同的数据
- **A:（Availability）：可用性——*reads and writes always succeed***
  - 是指快速获取数据，可以在确定的时间内返回操作结果，保证每个请求不管成功或者失败都有响应
- **P（Tolerance of Network Partition）：分区容忍性——*the system continues to operate despite arbitrary message loss or failure of part of the system***
  - 是指当出现网络分区的情况时（即系统中的一部分节点无法和其他节点进行通信），分离的系统也能够正常运行，也就是说，系统中任意信息的丢失或失败不会影响系统的继续运作。

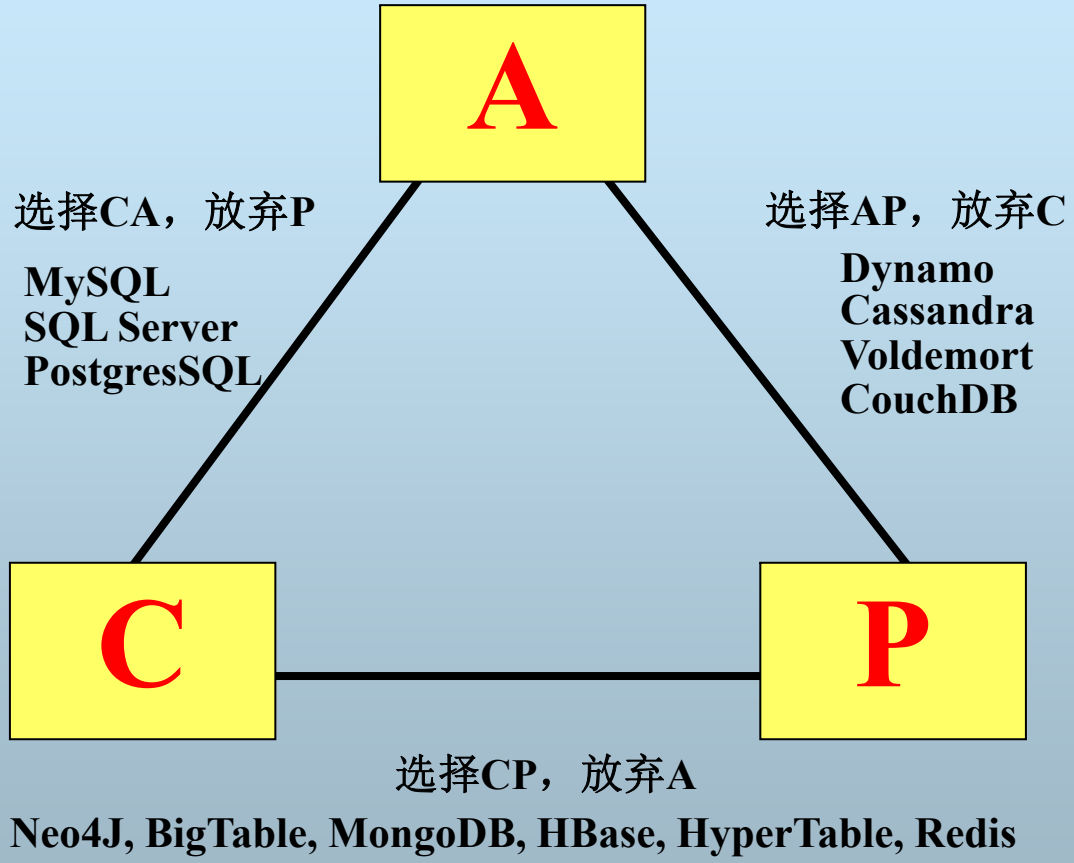
1、CAP

- **Brewer's Theorem (CAP Theorem):** 一个分布式系统不可能同时满足一致性、可用性和分区容忍性这三个需求，最多只能同时满足其中两个 **(Brewer, 2000; Gilbert, 2002)**



Brewer, Eric A. (2000): *Towards Robust Distributed Systems*. Keynote at the ACM Symposium on Principles of Distributed Computing (PODC).
Gilbert, S., & Lynch, N. (2002): *Brewers Conjunction and the Feasibility of Consistent, Available, Partition-Tolerant Web Services*. ACM SIGACT News, p. 33(2).

1、CAP



不同产品在CAP理论下的不同设计原则

2、BASE

- **BASE（Basically Available, Soft-state, Eventual consistency (Pritchett, 2008)**
  - **是对CAP理论的延伸**

ACID	BASE
原子性(**A**tomicity)	基本可用(**B**asically **A**vailable)
一致性(**C**onsistency)	软状态/柔性事务(**S**oft state)
隔离性(**I**solation)	最终一致性 (**E**ventual consistency)
持久性 (**D**urability)	

BASE vs. ACID

Dan Pritchett. (2008): *BASE: An ACID Alternative*. ACM *Queue*, Vol.6(3): 48-55

2、BASE

- **Basically Available**
  - 基本可用，是指一个分布式系统的一部分发生问题变得不可用时，其他部分仍然可以正常使用。也即允许损失部分可用性。
- **Soft-state**
  - “软状态（Soft-state）”是与“硬状态（Hard-state）”相对应的一种提法。数据库保存的数据是“硬状态”时，可以保证数据一致性，即保证数据一直是正确的。“软状态”是指状态可以有一段时间不同步，具有一定的滞后性

2、BASE

- **Eventual consistency**
  - 一致性的类型包括强一致性和弱一致性。对于强一致性而言，当执行完一次更新操作后，后续的其他读操作就可以保证读到更新后的最新数据。如果不能保证后续访问读到的都是更新后的最新数据，那么就是弱一致性。
  - 最终一致性是弱一致性的一种特例，允许后续的访问操作可以暂时读不到更新后的数据，但是经过一段时间之后，必须最终读到更新后的数据。
    - 最常见的实现最终一致性的系统是DNS（域名系统）。一个域名更新操作根据配置的形式被分发出去，并结合有过期机制的缓存；最终所有的客户端可以看到最新的值。

本章小结

- 面向对象数据库
- 对象关系数据库
- **NoSQL数据库**