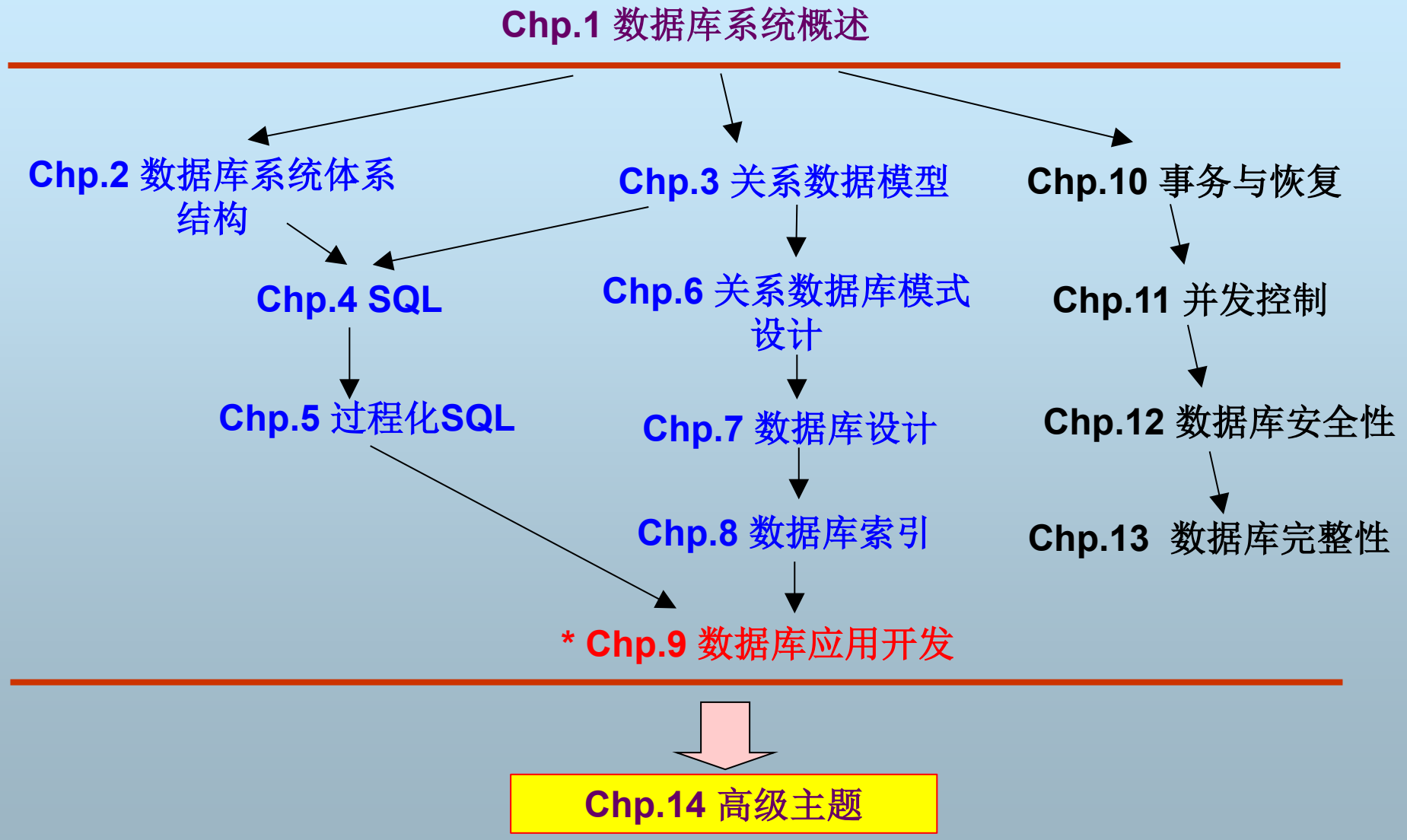
第14章 高级主题

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.10 事务与恢复
Chp.4 SQL
Chp.6 关系数据库模式设计
Chp.11 并发控制
Chp.5 过程化SQL
Chp.7 数据库设计
Chp.12 数据库安全性
Chp.8 数据库索引
Chp.13 数据库完整性
* Chp.9 数据库应用开发
Chp.14 高级主题

主要内容

- 分布式数据库
- 面向对象数据库
- 对象关系数据库
- NoSQL数据库

分布式数据库

金培权（jpq@ustc.edu.cn）

主要内容

- 分布式数据库的概念
- 分布式数据库的特点
- 分布式数据库的模式结构

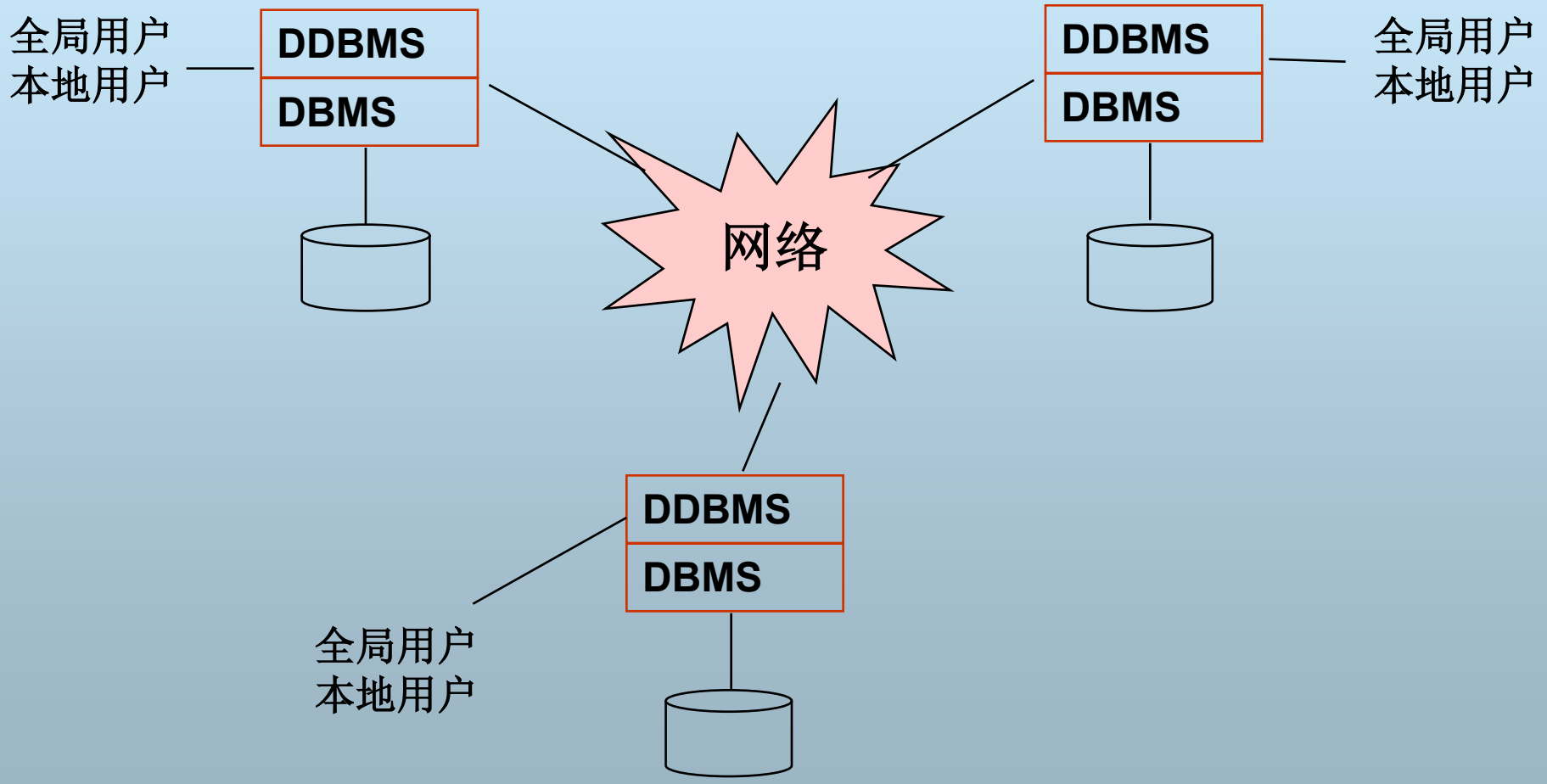
一、分布式数据库的概念

- 1970s，计算机网络技术发展使不同地点的数据库互连成为可能
- 自70年代始，数据库技术逐步成熟
- 分布式管理符合许多企业应用需求
  - 银行、保险、跨国企业、连锁业、交通运输业……
  - 地域上分散，管理上集中

一、分布式数据库的概念

- 定义：物理上分布而逻辑上集中的数据库系统
  - 每个站点(Site)自身具有完全的本地DBS
  - 所有站点协同工作，组成了逻辑上的统一数据库
  - 站点数据由分布式DBMS(DDBMS)管理——DBMS+分布式扩展模块
  - 本地应用和本地用户(Local)
    - 只访问其所注册的那个站点上的数据
  - 全局应用和全局用户(Global)
    - 访问涉及多个站点上的数据

一、分布式数据库的概念


全局用户
本地用户
DDBMS
DBMS
网络
DDBMS
DBMS
全局用户
本地用户
DDBMS
DBMS
全局用户
本地用户

二、分布式数据库的特点

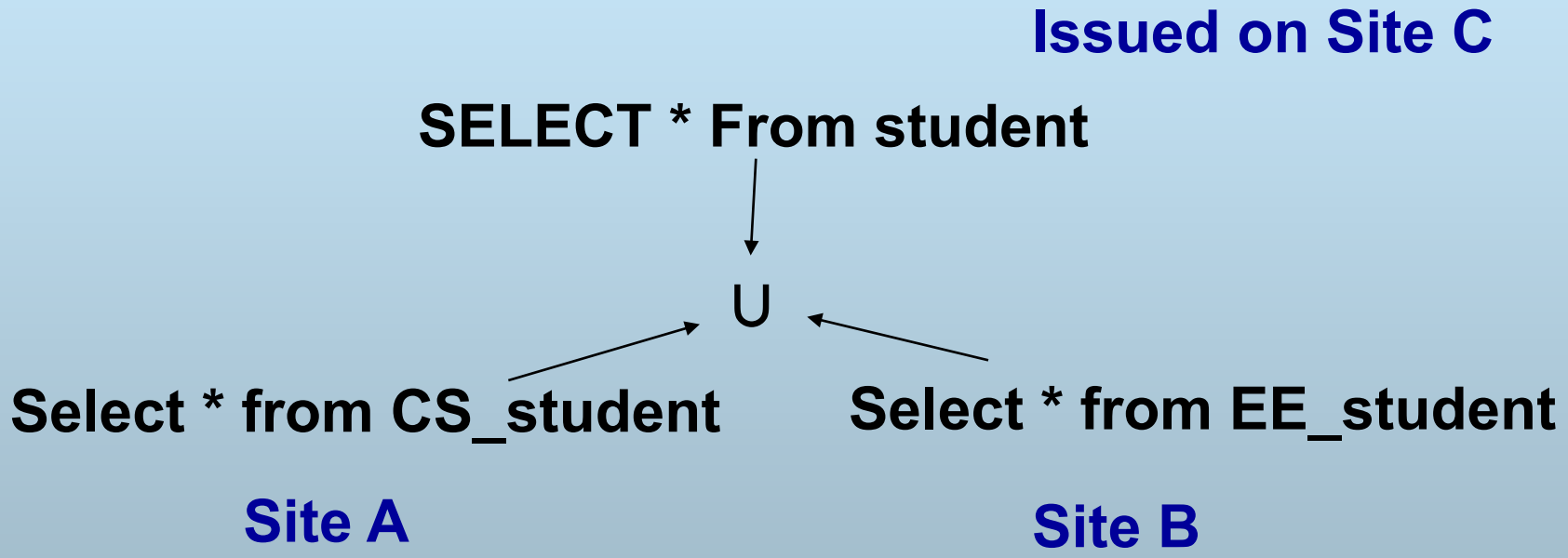
- 特点
  - 物理分布性
  - 逻辑整体性
  - 站点自治性
  - 数据透明性
    - 用户不需要知道数据的物理位置以及如何访问某个特点站点的数据

二、分布式数据库的特点

- 数据透明性
  - 位置透明性
    - 用户无须知道数据的物理存储位置（站点）
  - 复制透明性
    - DDBS通过数据复制来提高系统可用性。但用户不必关系复制了什么数据对象以及副本存储在什么位置
  - 分片透明性
    - 数据分片：将数据分割成不同的片断，例如将CS学生存在站点A，EE学生存在站点B
    - 用户无须知道数据是如何分片的

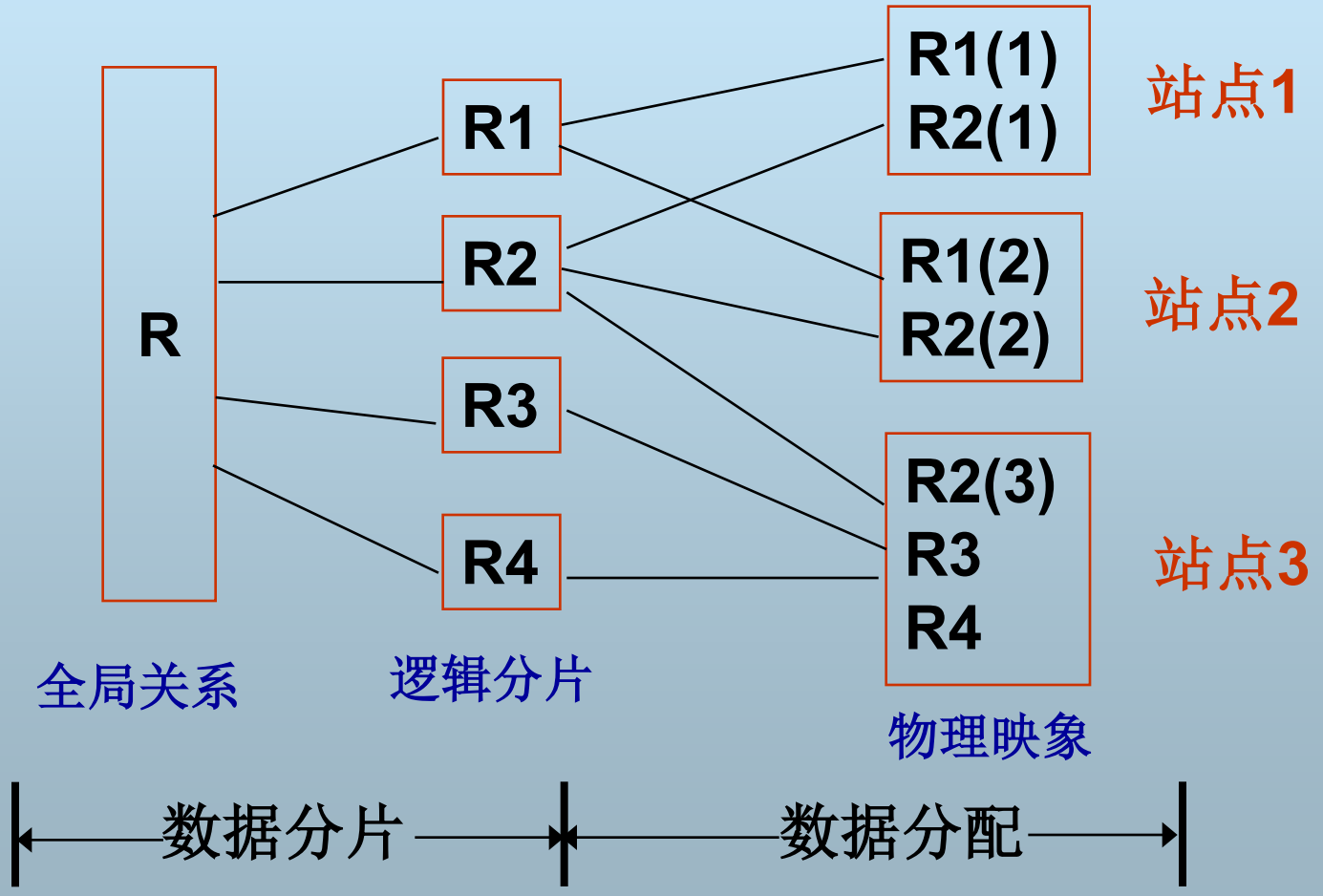
二、分布式数据库的特点

- 分片透明性例子

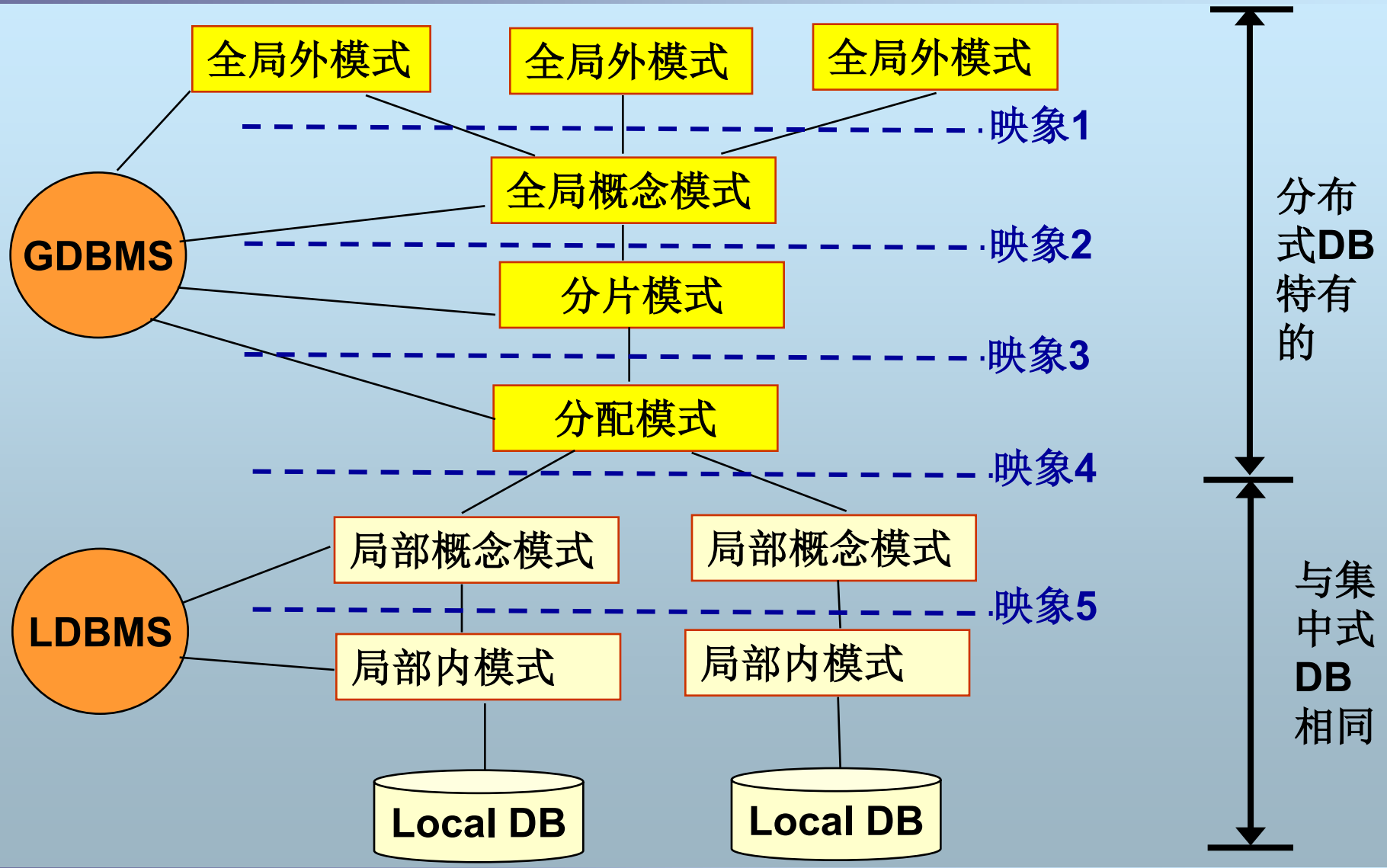


二、分布式数据库的特点

- 从数据分片到数据分配
  - 将分片分散存储到各个站点上



三、分布式数据库模式结构



面向对象数据库和对象关系数据库

金培权（jpq@ustc.edu.cn）

主要内容

- OODB的产生与发展
- OODB的实现方法
- ORDB

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

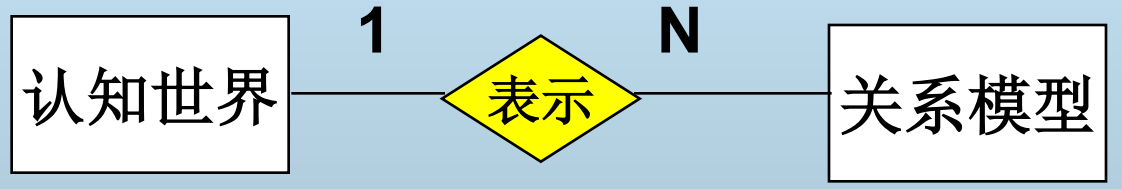
ISDN	Title	Publisher
0-201	Compilers	Springer
……	……	……

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

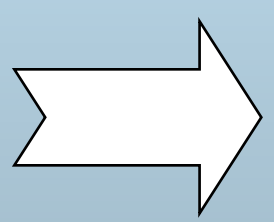
一、OODB的产生与发展

- **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

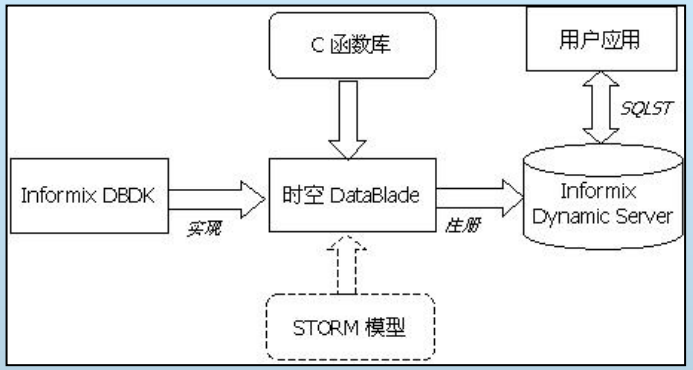
- **ORDB中对象查询例子**
  - **Country(name:char(30), boundary: polygon)**
  - **查询例子“查询中国的邻国名称”**

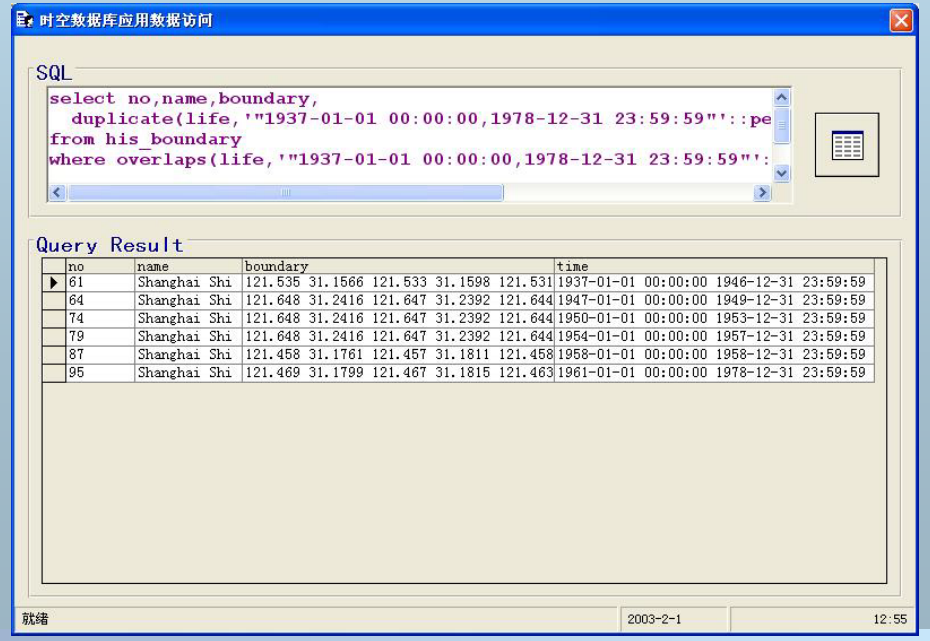
```
Select C.name
From Country C, Country D
Where D.name='China' and Meet(C.boundary, D.boundary)
```

使用了扩充的ADT操作Meet

4、利用ORDB实现复杂数据管理

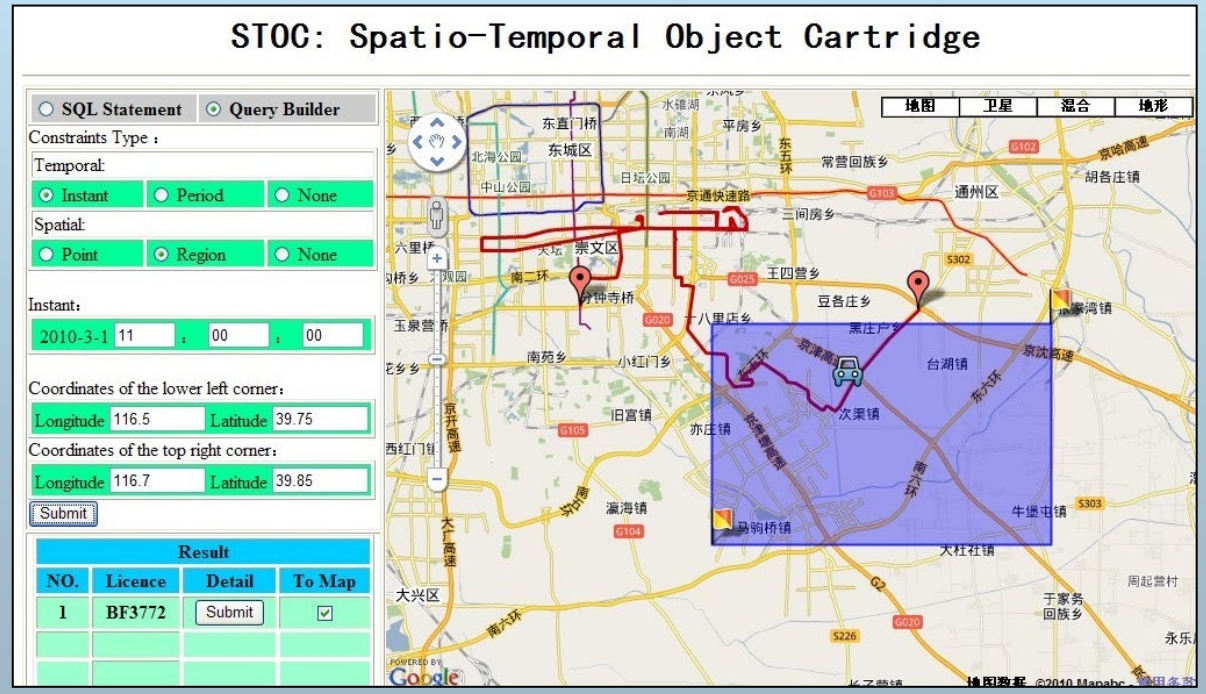
- 基于Informix Datablade的时空数据管理

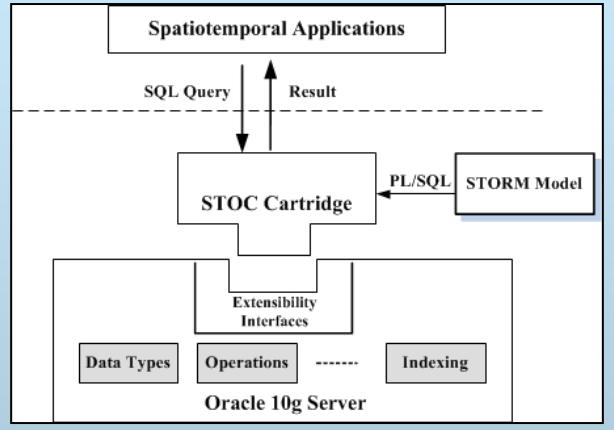




4、利用ORDB实现复杂数据管理

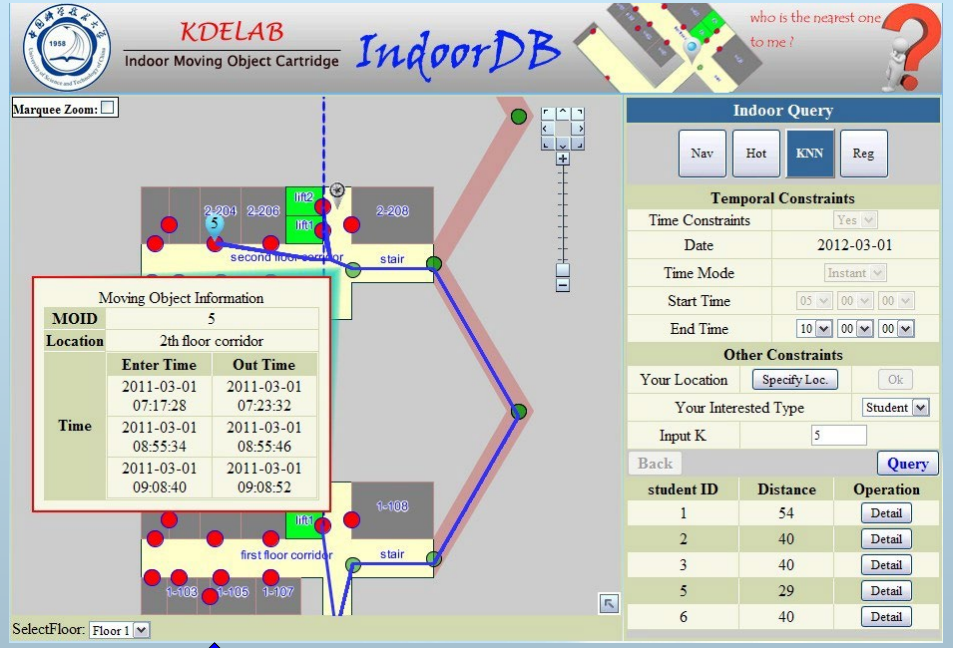
- 基于Oracle Cartridge的时空数据管理

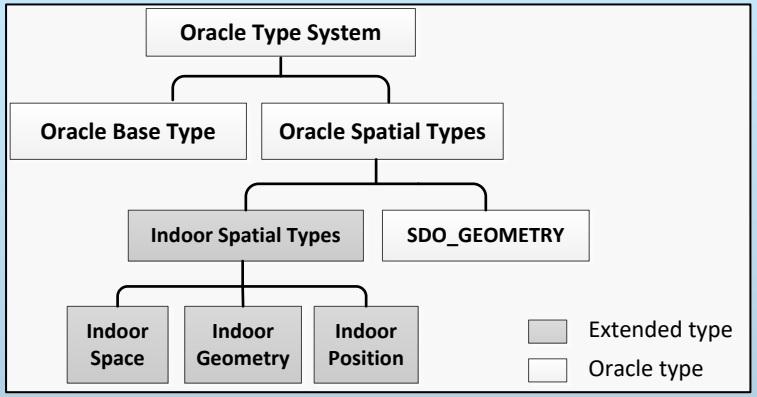




4、利用ORDB实现复杂数据管理

- 基于Oracle Cartridge的室内移动对象管理

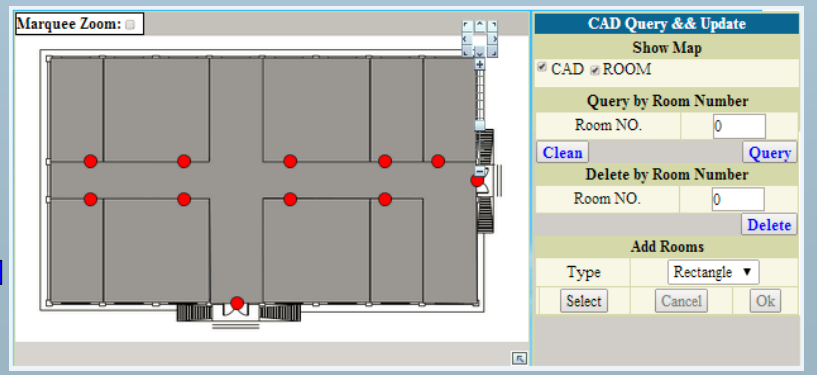




支持室内位置服务

Oracle IndoorDB

从CAD文件中抽取室内空间对象



NoSQL Databases

金培权（jpq@ustc.edu.cn）

主要内容

- NoSQL简介
- NoSQL主要的类型
- NoSQL的分布式系统基础

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.**

一、NoSQL简介

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

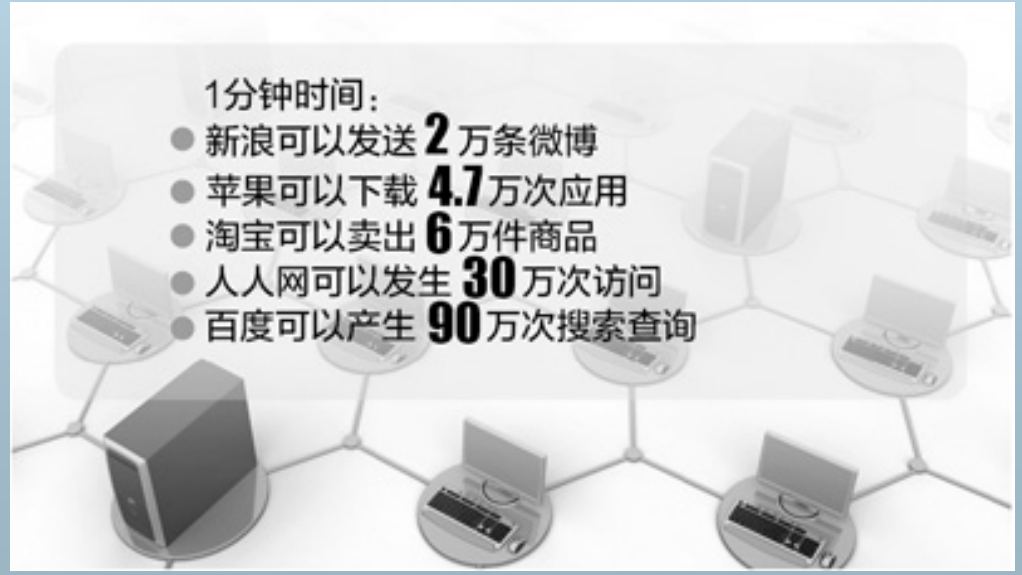
一、NoSQL简介

- 现在已有很多公司使用了NoSQL数据库：
  - Google
  - Facebook
  - Adobe
  - Foursquare
  - LinkedIn
  - 字节、腾讯、阿里、新浪、华为……

1、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　TB→PB →ZB
- 无法满足数据高并发的需求　1K→1M→10M 并发
- 无法满足高可扩展性和高可用性的需求



1、NoSQL兴起的原因

（2）“One size fits all”模式很难适用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等针对在线业务，两者都抛弃了关系模型

1、NoSQL兴起的原因

（3）关系数据库的关键特性包括完善的事务机制和高效的查询机制。这些关键特性在Web 2.0时代出现了变化：

- 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一致性
- 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）

2、NoSQL vs. RDBMS

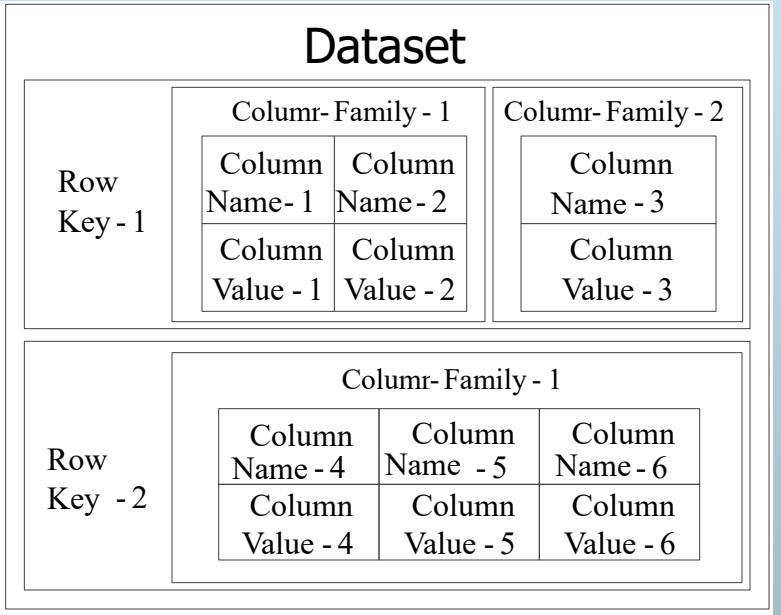
- **RDBMS**
  - 优势：以完善的关系代数理论作为基础，有严格的标准，支持事务ACID，提供严格的数据一致性，借助索引机制可以实现高效的查询，技术成熟，有专业公司的技术支持
  - 劣势：(1)扩展性较差，无法较好支持海量数据存储，(2)采用固定的数据库模式，无法较好支持Web 2.0应用，(3)事务机制影响系统的整体性能等
- **NoSQL**
  - 优势：(1)具有强大的横向扩展能力，支持超大规模数据存储，数据分布和复制容易，(2)灵活的数据模型可以很好地支持Web 2.0应用
  - 劣势：缺乏数学理论基础，复杂查询性能不高，大都不能实现事务强一致性，很难实现数据完整性，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的主要类型

- 键值数据库、列存储数据库、文档数据库和图数据库

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

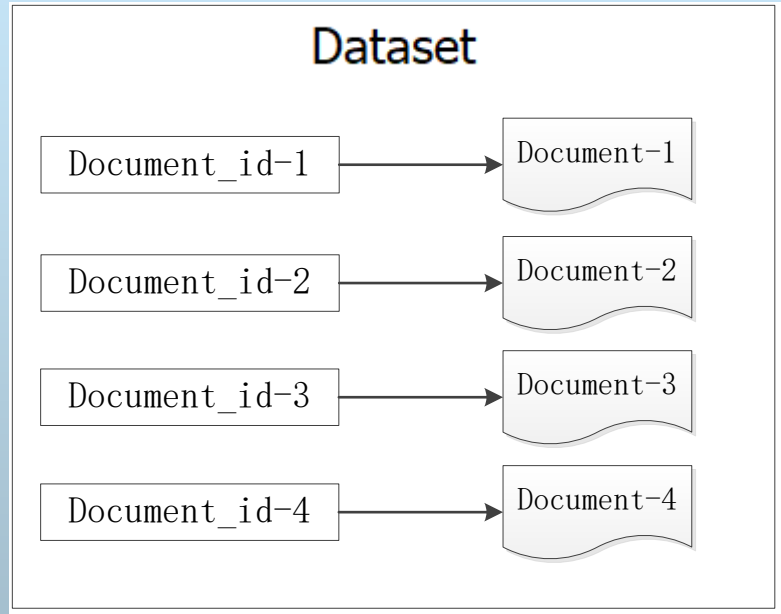
键值数据库



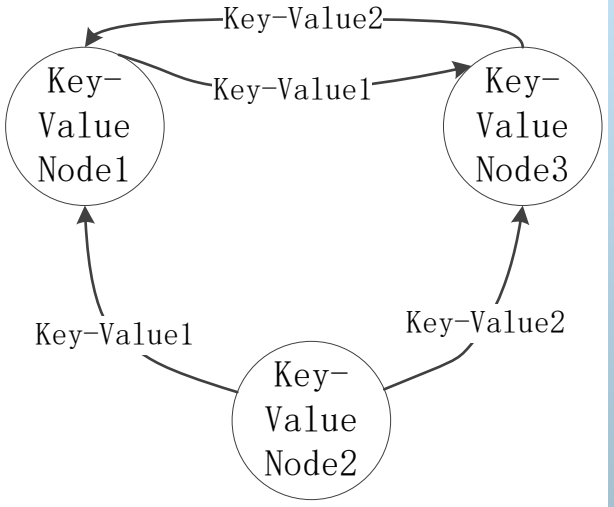
列存储数据库

二、NoSQL的主要类型

- 键值数据库、列存储数据库、文档数据库和图数据库



文档数据库



图数据库

Key-Value结构

- NoSQL数据库大都以“KEY+VALUE”结构为基础进行数据表示和存储
- KEY与VALUE都以字节流（byte string）存储：

```
typedef struct {
    void *data; //字节流指针
    int size; //字节流长度
} DBT;
```

- 数据类型没有限制
- 应用与数据库之间不需转换数据格式
- 不提供KEY和VALUE的内容和结构信息
- 应用必须知道所用的VALUE的含义

二、NoSQL的主要类型

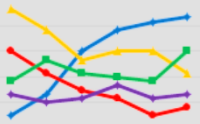
文档数据库	图数据库
 Couchbase MarkLogic mongoDB	Neo4j InfiniteGraph The Distributed Graph Database
键值数据库	**列存储数据库**
redis amazon DynamoDB levelDB riak	accumulo HYPERTABLE INC Cassandra APACHE HBASE Amazon SimpleDB

Overall Rank

DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.



trend chart

423 systems in ranking, December 2024

Rank Dec 2024	Rank Nov 2024	Rank Dec 2023	DBMS	Database Model	Score Dec 2024	Score Nov 2024	Score Dec 2023
1.	1.	1.	Oracle	Relational, Multi-model	1263.79	-53.22	+6.38
2.	2.	2.	MySQL	Relational, Multi-model	1003.76	-14.04	-122.88
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	805.69	+5.88	-98.14
4.	4.	4.	PostgreSQL	Relational, Multi-model	666.37	+12.04	+15.47
5.	5.	5.	MongoDB	Document, Multi-model	400.39	-0.54	-18.76
6.	6.	6.	Redis	Key-value, Multi-model	150.27	+1.63	-8.08
7.	7.	↑ 10.	Snowflake	Relational	147.36	+4.87	+27.48
8.	8.	↓ 7.	Elasticsearch	Multi-model	132.32	+0.68	-5.43
9.	9.	↓ 8.	IBM Db2	Relational, Multi-model	122.78	+1.04	-11.81
10.	10.	↑ 11.	SQLite	Relational	101.72	+2.24	-16.23

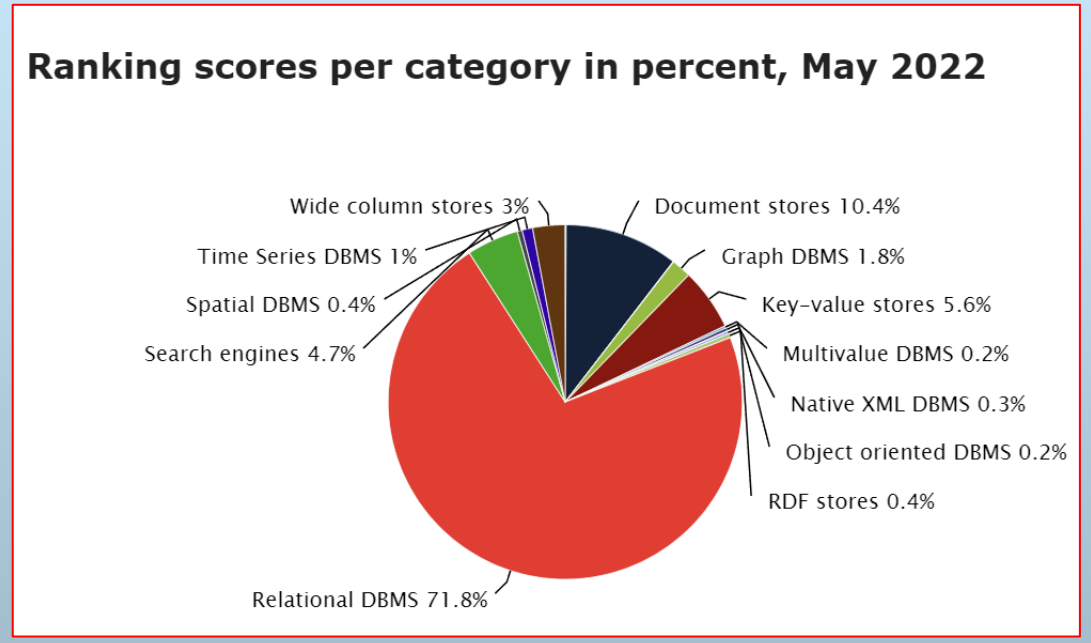
Method of calculating the scores of the DB-Engines Ranking

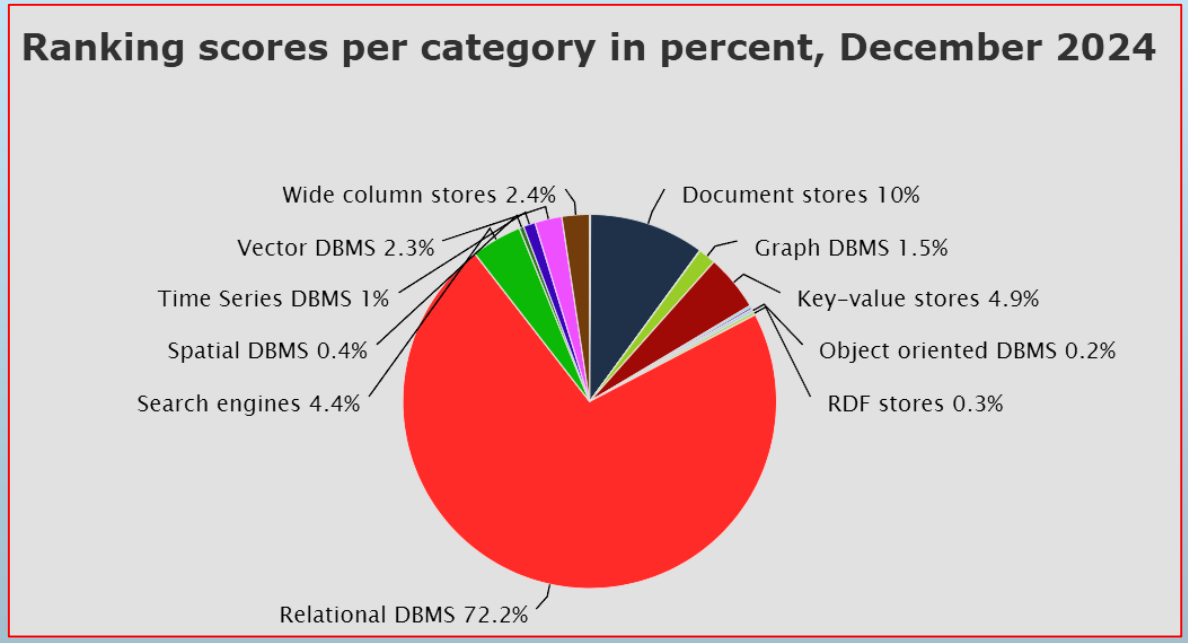
The DB-Engines Ranking is a list of database management systems ranked by their current popularity. We measure the popularity of a system by using the following parameters:

- **Number of mentions of the system on websites**, measured as number of results in search engines queries. At the moment, we use Google and Bing for this measurement. In order to count only relevant results, we are searching for <system name> together with the term database, e.g. "Oracle" and "database".
- **General interest in the system.** For this measurement, we use the frequency of searches in Google Trends.
- **Frequency of technical discussions about the system.** We use the number of related questions and the number of interested users on the well-known IT-related Q&A sites Stack Overflow and DBA Stack Exchange.
- **Number of job offers, in which the system is mentioned.** We use the number of offers on the leading job search engines Indeed and Simply Hired.
- **Number of profiles in professional networks, in which the system is mentioned.** We use the internationally most popular professional network LinkedIn.
- **Relevance in social networks.** We count the number of Twitter (X) tweets, in which the system is mentioned.

Overall Rank (cont.)

- 关系数据库仍是主流，但NoSQL比例在不断增长

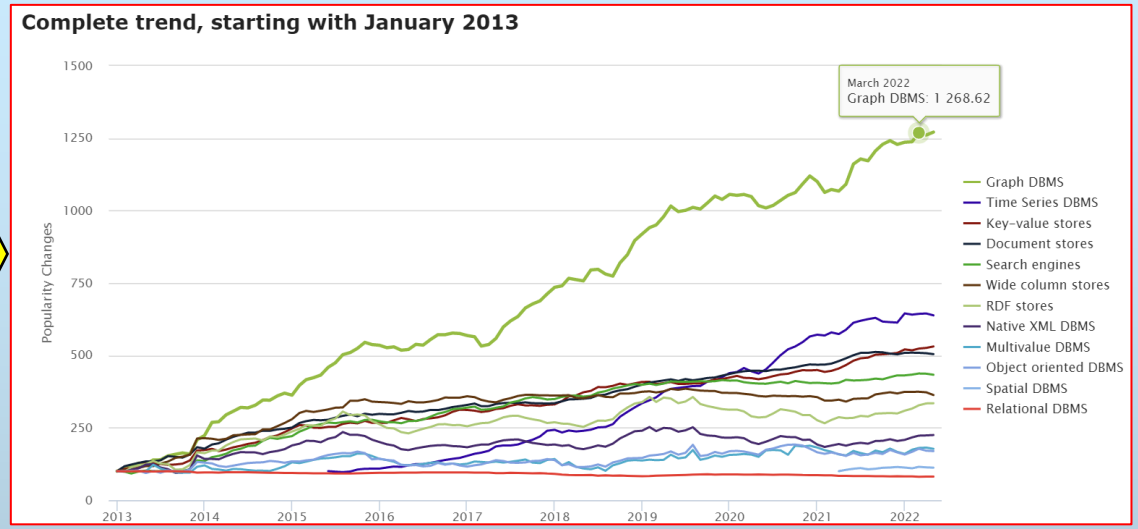




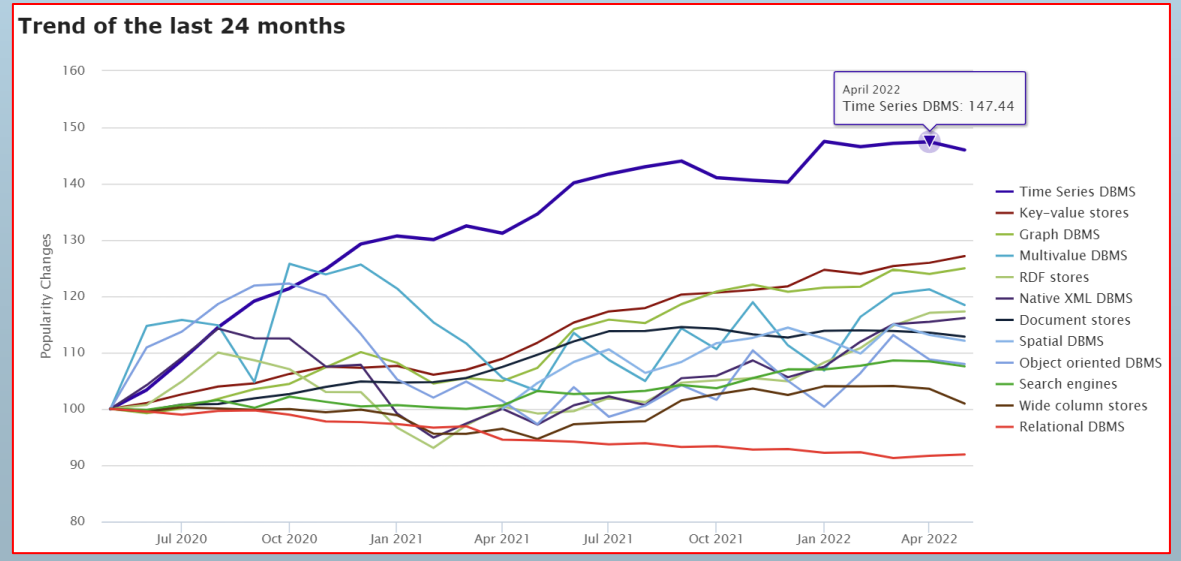
Overall Rank (cont.)

- 发展趋势

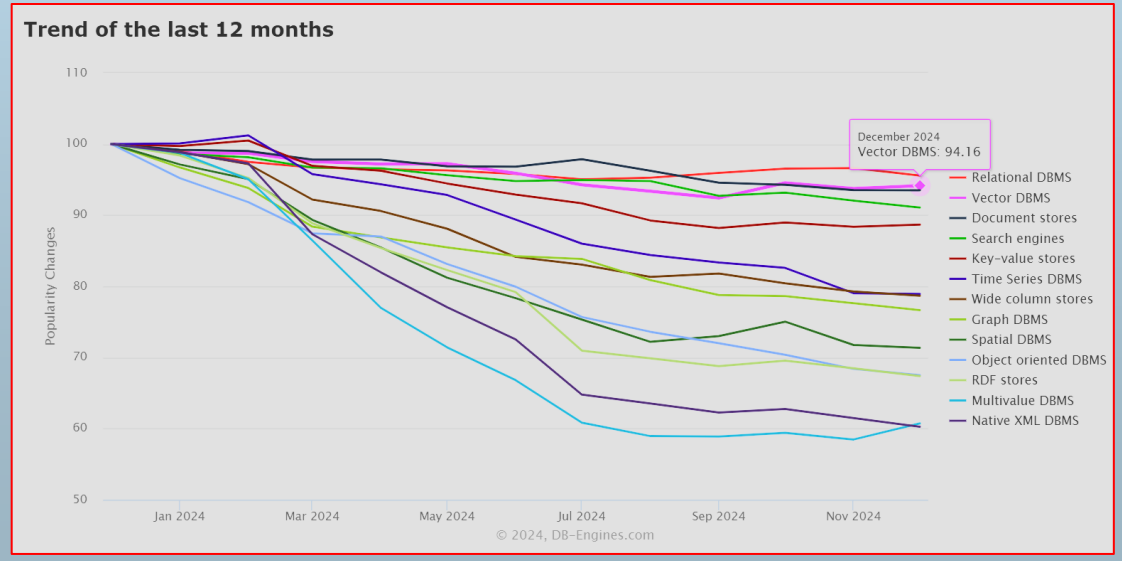
图数据库



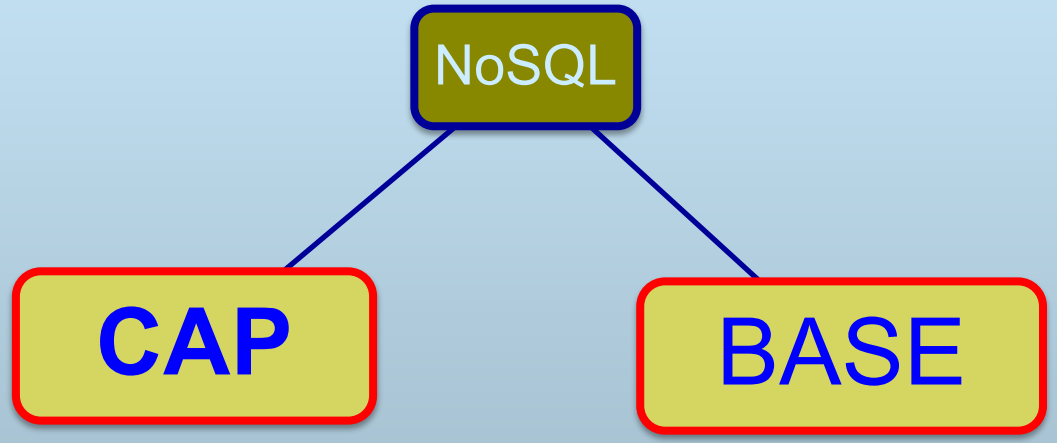
时序数据库



向量数据库



三、NoSQL的分布式系统基础



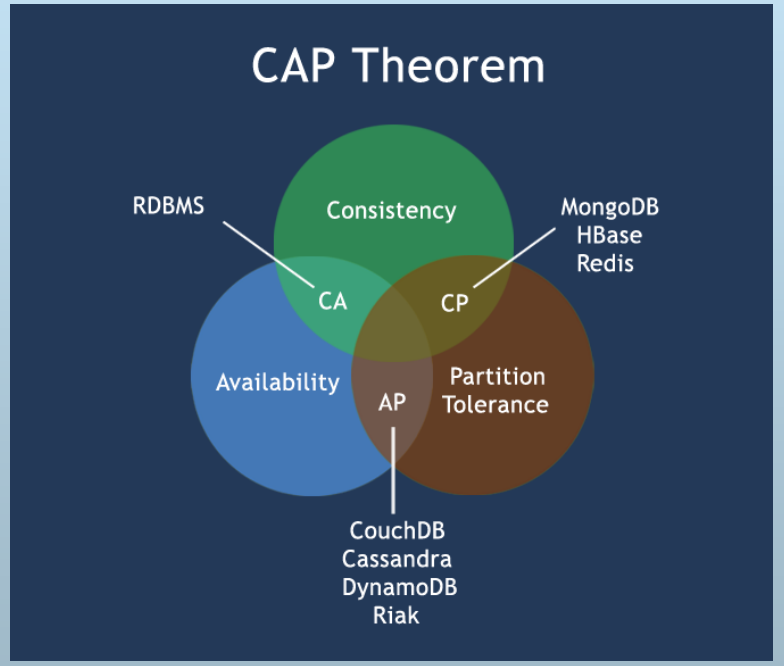
Note：基本都是分布式系统中的技术，跟数据库系统关系不大

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。

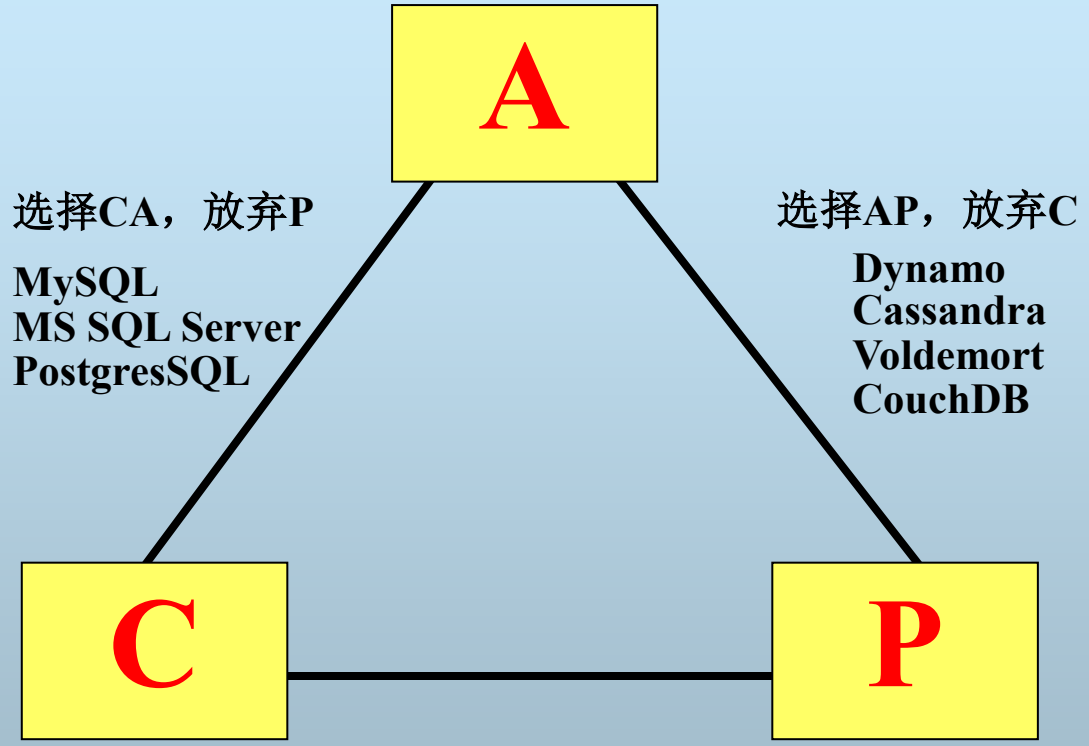
1、CAP

- **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**urable)	

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

- 分布式数据库
- 面向对象数据库
- 对象关系数据库
- NoSQL数据库