

# 课程内容

- 数据库系统基本概念（数据模型，体系结构）
- 关系数据库
- 关系数据库标准语言SQL
- 数据库保护
- 关系数据理论
- 数据库设计
- 存储管理与存取方法
- 查询处理和查询优化
- 事务处理技术
- 数据库技术新发展

基础理论

设计理论

实现技术

数据库系统基本原理

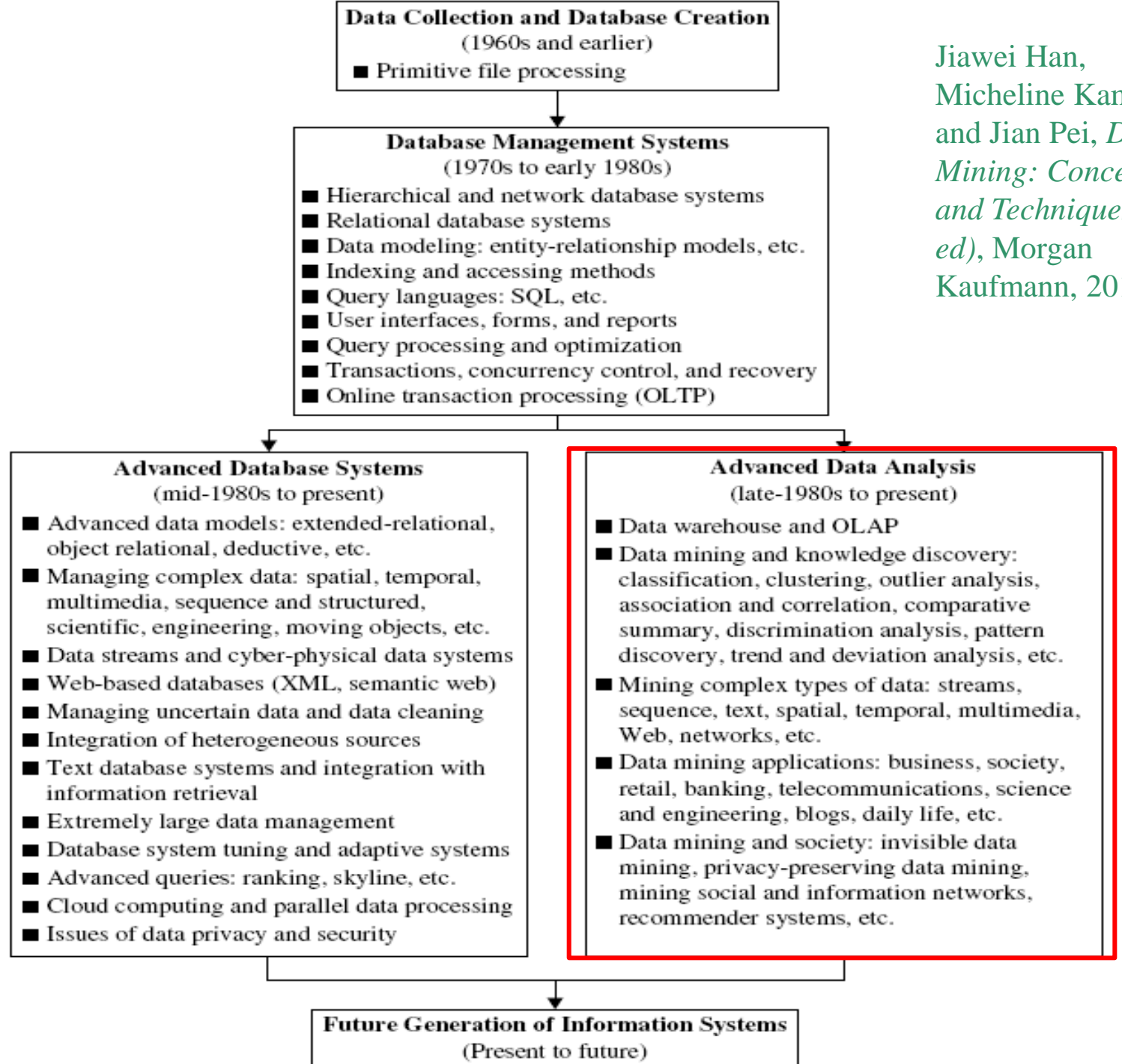
新技术

# 第四部分 数据库新技术

---

- 概述
- 分布式数据库(NewSQL)

# 数据库技术的进化



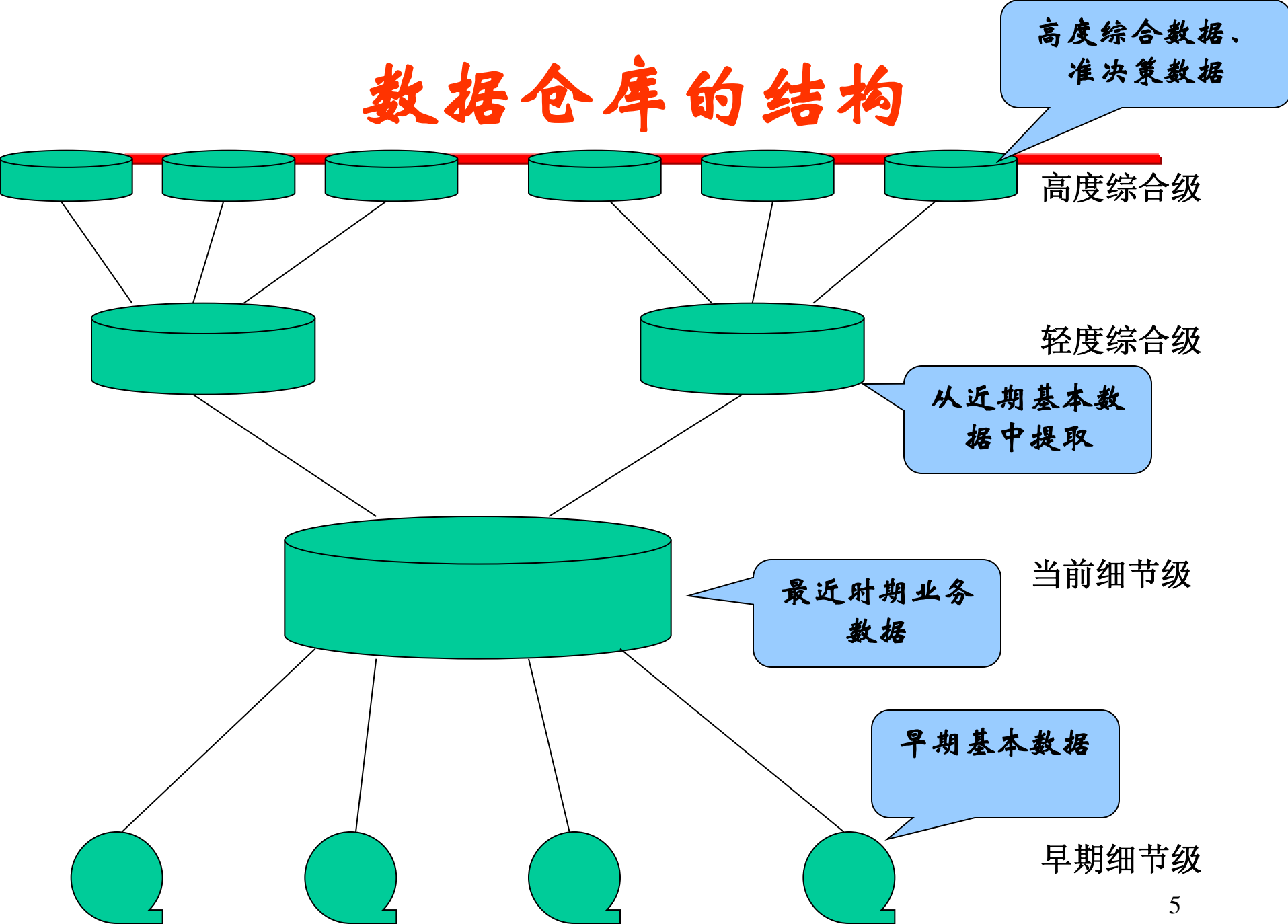
Jiawei Han,  
Micheline Kamber  
and Jian Pei, *Data  
Mining: Concepts  
and Techniques (3<sup>rd</sup>  
ed)*, Morgan  
Kaufmann, 2011

# 什么是数据仓库

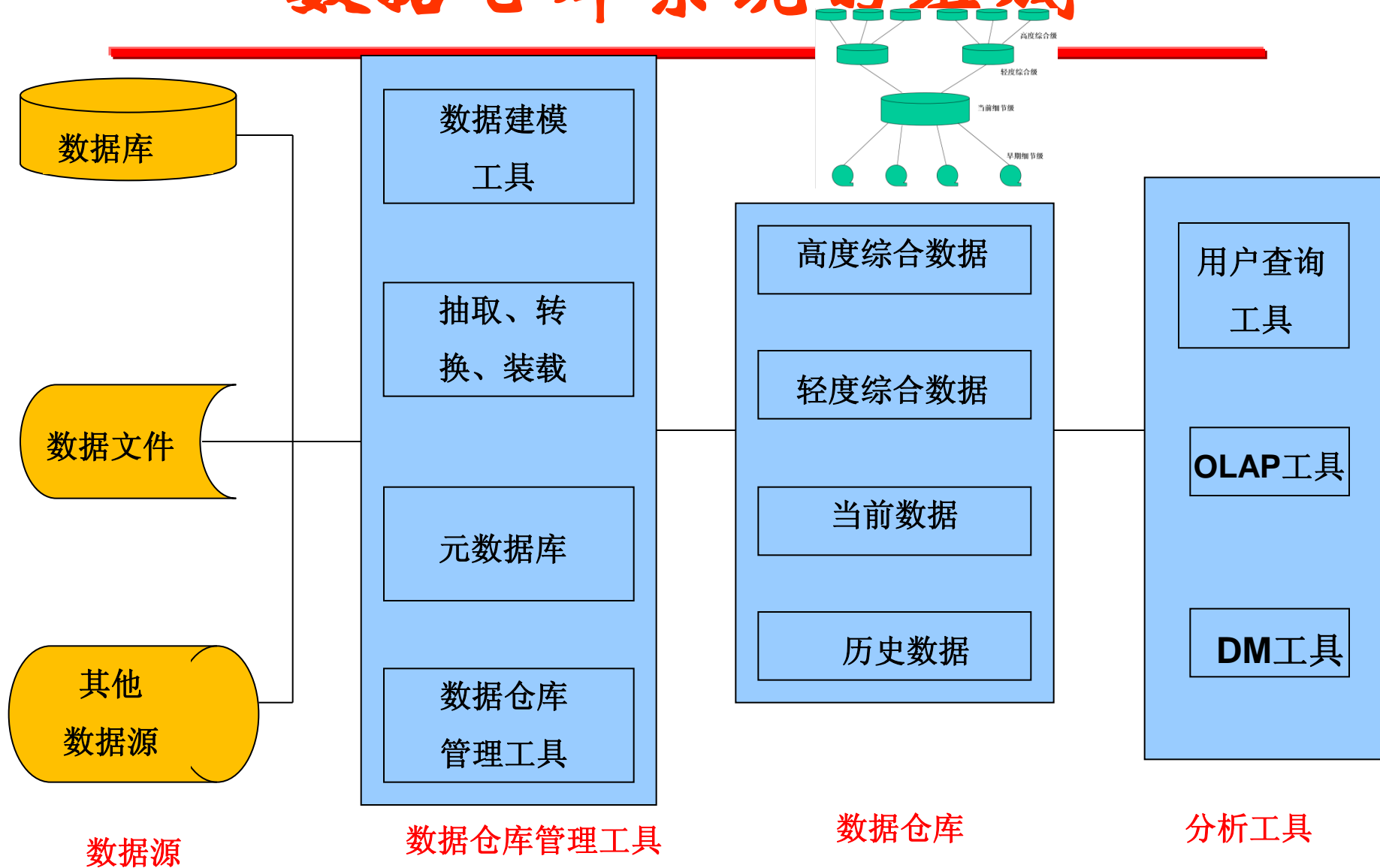
---

- 数据仓库(Data Warehouse)1990年提出, 是支持管理决策过程的、面向主题的、集成的、随时间而增长的持久数据集合
- 数据仓库中的业务
  - 数据仓库上的业务处理称作OLAP( On-Line Analytical Processing), 即联机分析处理
  - 数据库上的业务处理称作OLTP(On-Line Transaction Processing), 即联机事务处理

# 数据仓库的结构

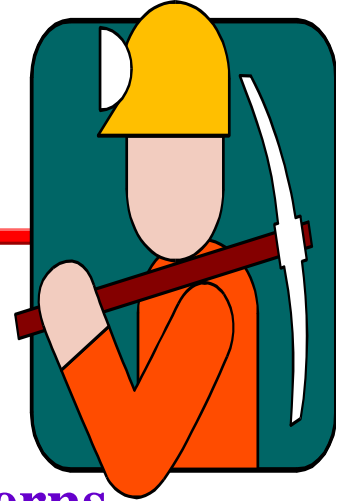


# 数据仓库系统的组成



# What Is Data Mining?

---

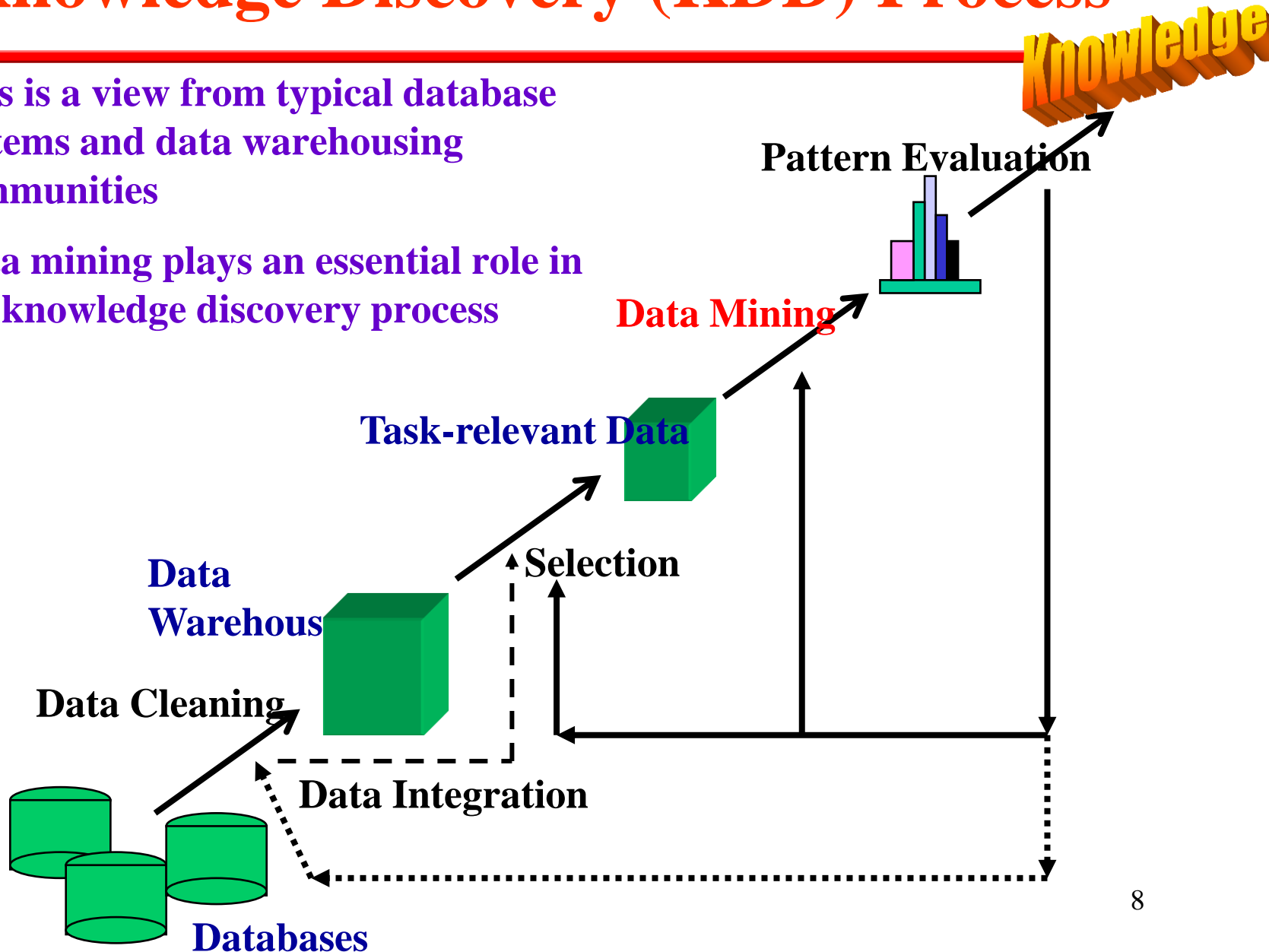


- Data mining (knowledge discovery from data)
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from huge amount of data
- Alternative names
  - Knowledge discovery (mining) in databases (KDD), knowledge extraction, data/pattern analysis, business intelligence, etc.



# Knowledge Discovery (KDD) Process

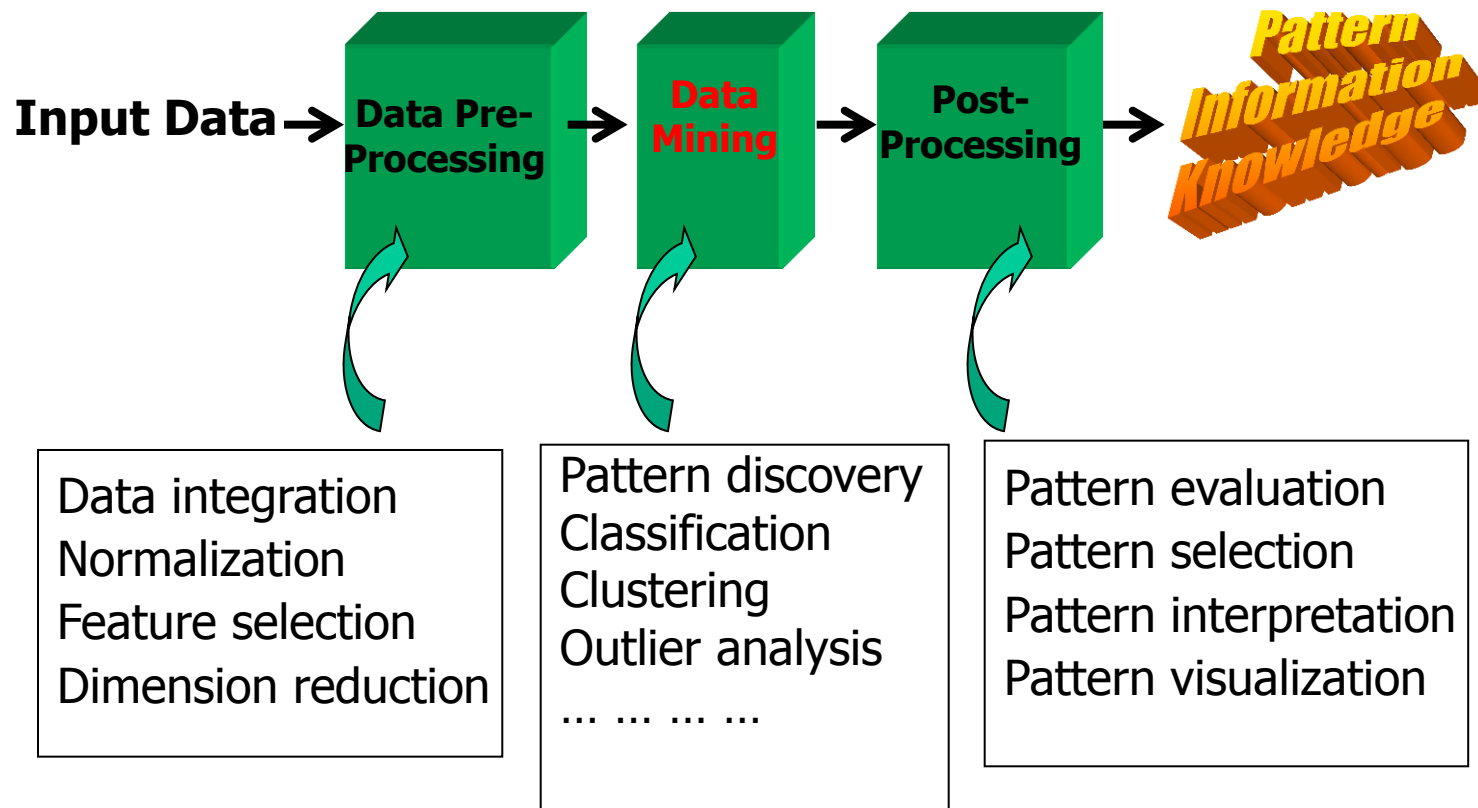
- This is a view from typical database systems and data warehousing communities
- Data mining plays an essential role in the knowledge discovery process





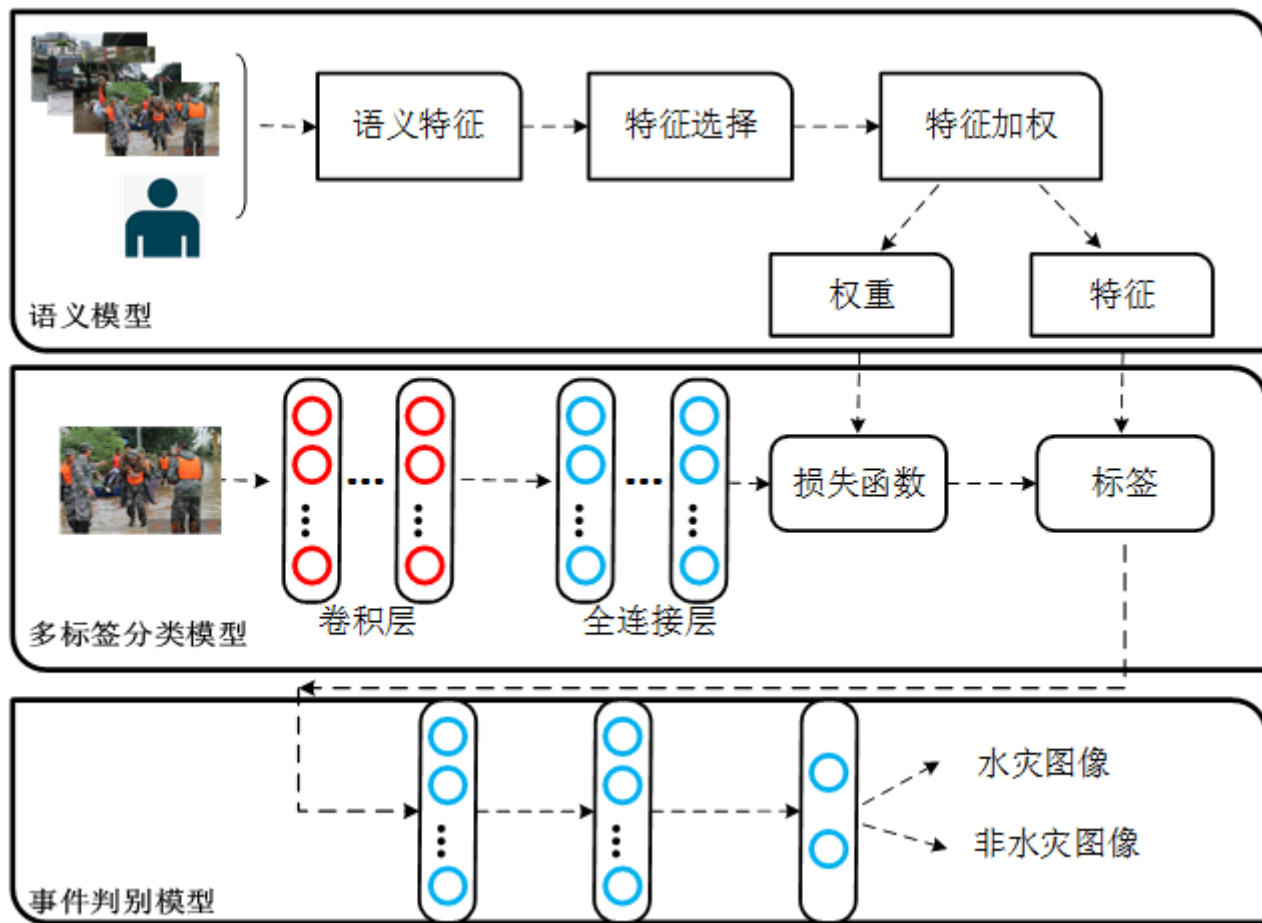
# KDD Process: A View from ML and Statistics

---



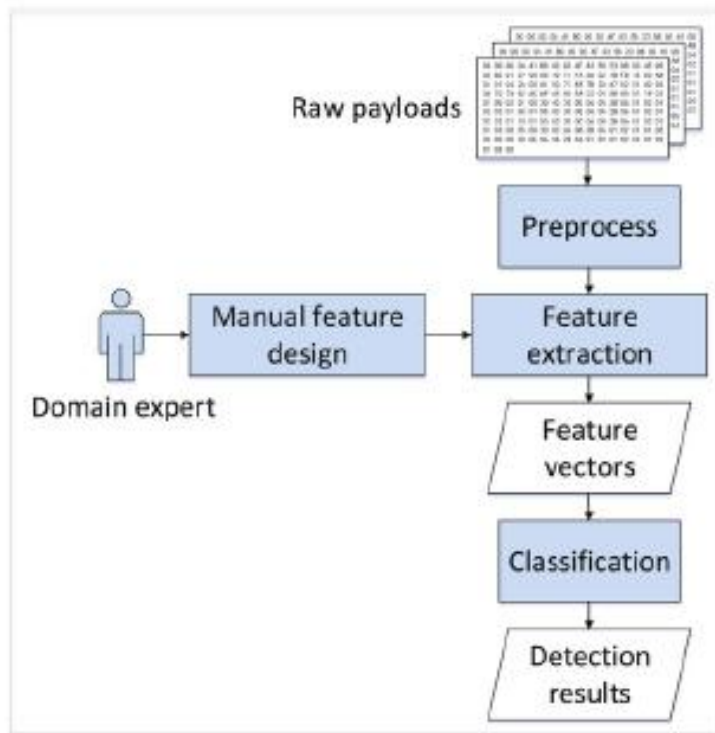
# 基于机器学习的数据分析示例

## • 水灾事件识别方法

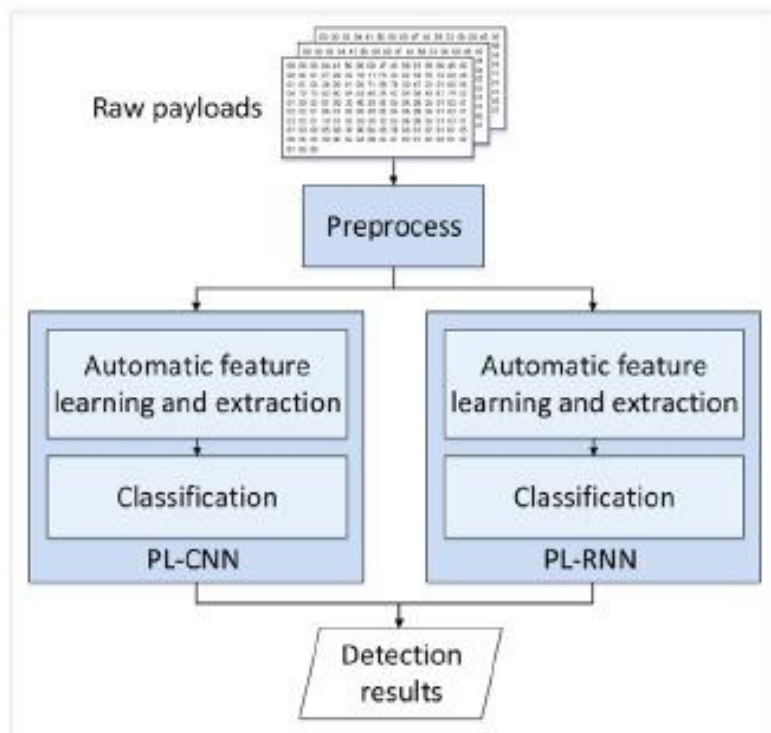


# 基于机器学习的数据分析示例

- 网络异常流量检测



(a) Traditional machine learning methods

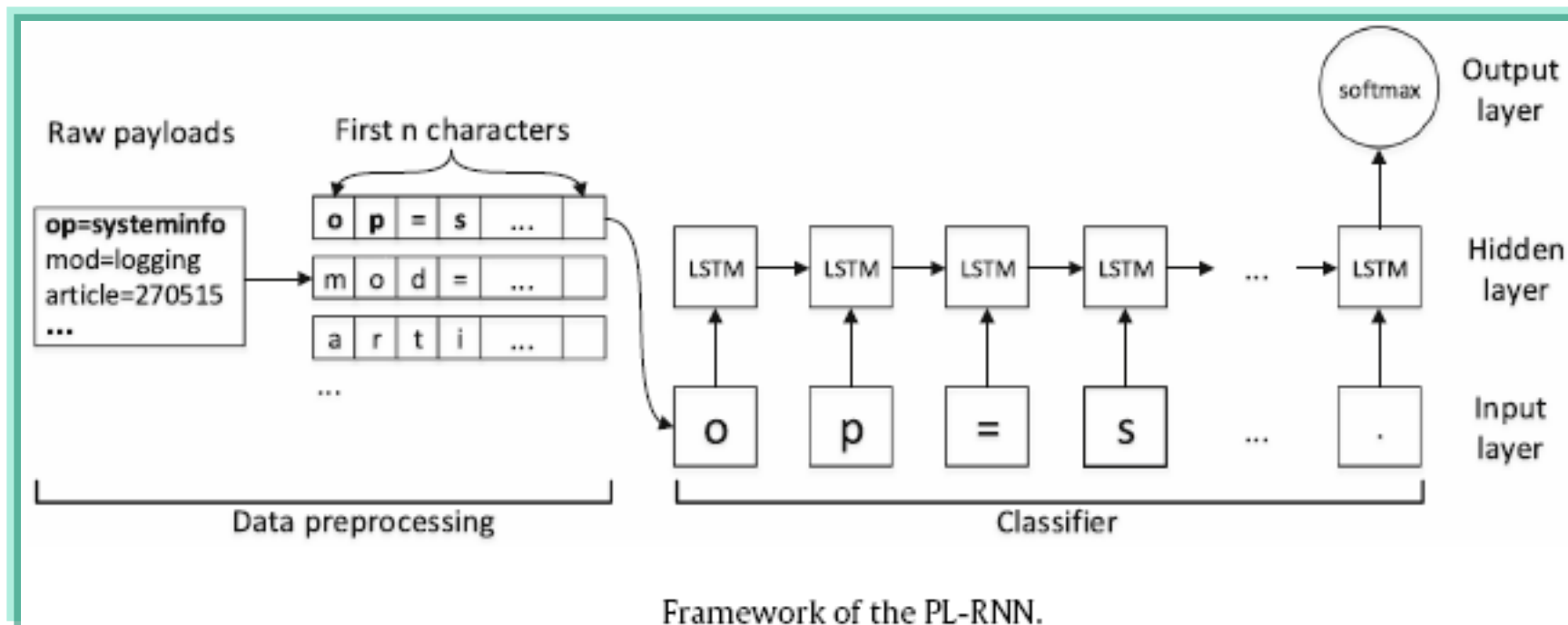


(b) Deep learning methods

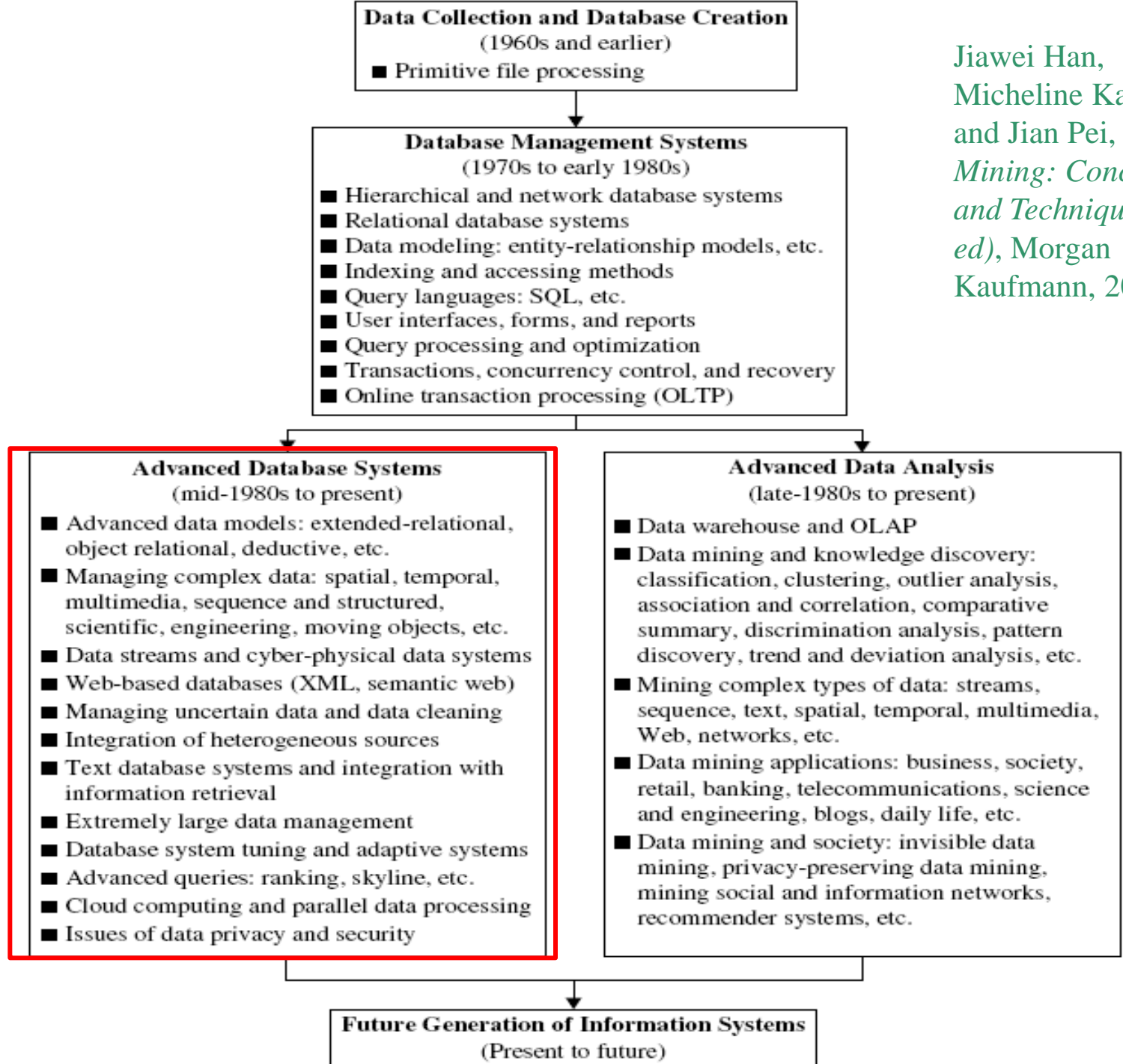
Attack detection process.

# 基于机器学习的数据分析示例

- 网络异常流量检测



# 数据库技术的进化

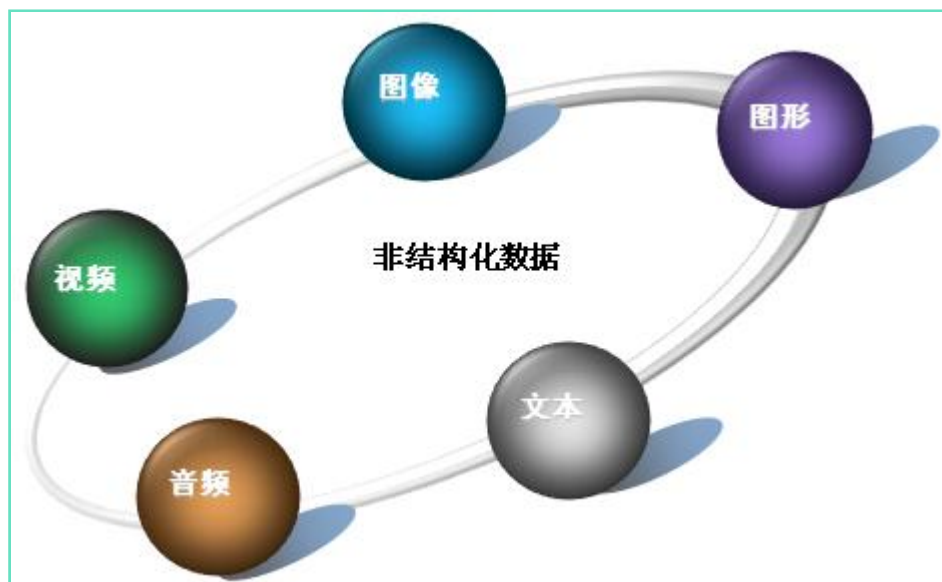


Jiawei Han,  
Micheline Kamber  
and Jian Pei, *Data  
Mining: Concepts  
and Techniques (3<sup>rd</sup>  
ed)*, Morgan  
Kaufmann, 2011

# 新时代数据管理面临的挑战

---

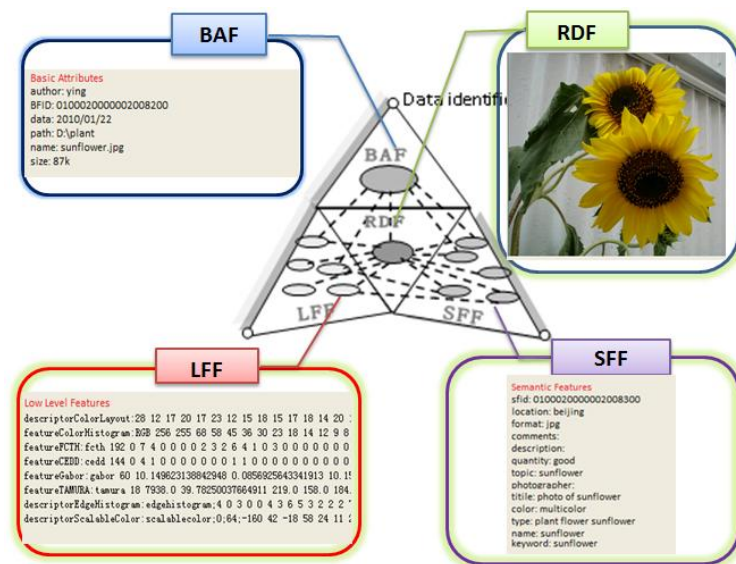
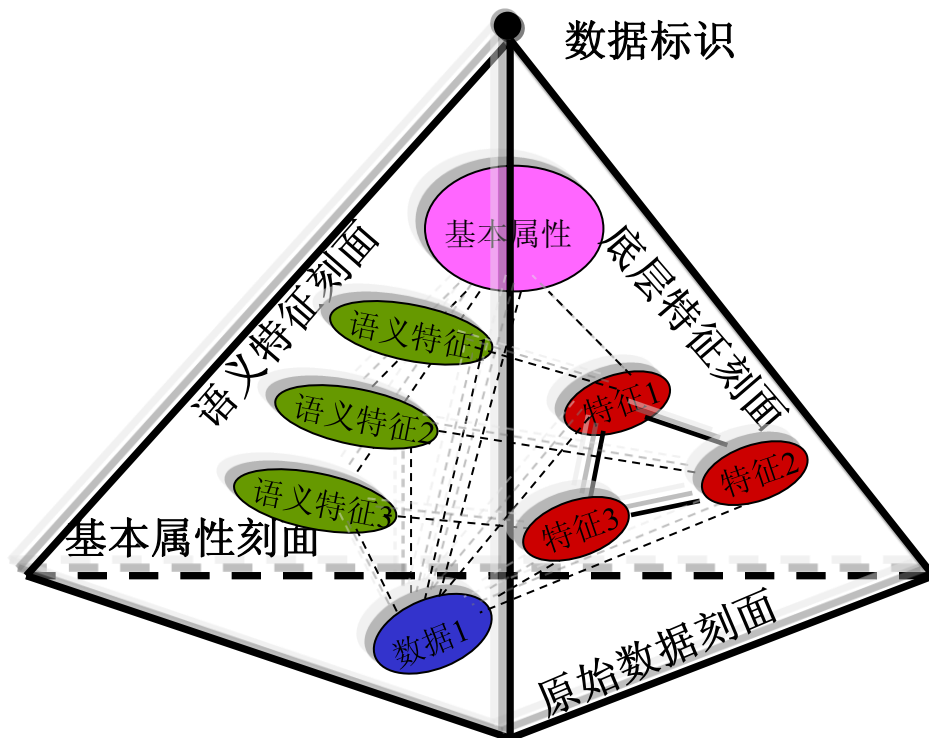
- 数据量：互联网时代，数据量呈指数级飞速增长，从TB到PB或更多
- 用户数：从几千人到几亿人
- 非结构化数据占总数据量的80%以上





# 非结构化数据管理

## 统一数据模型：四面体模型



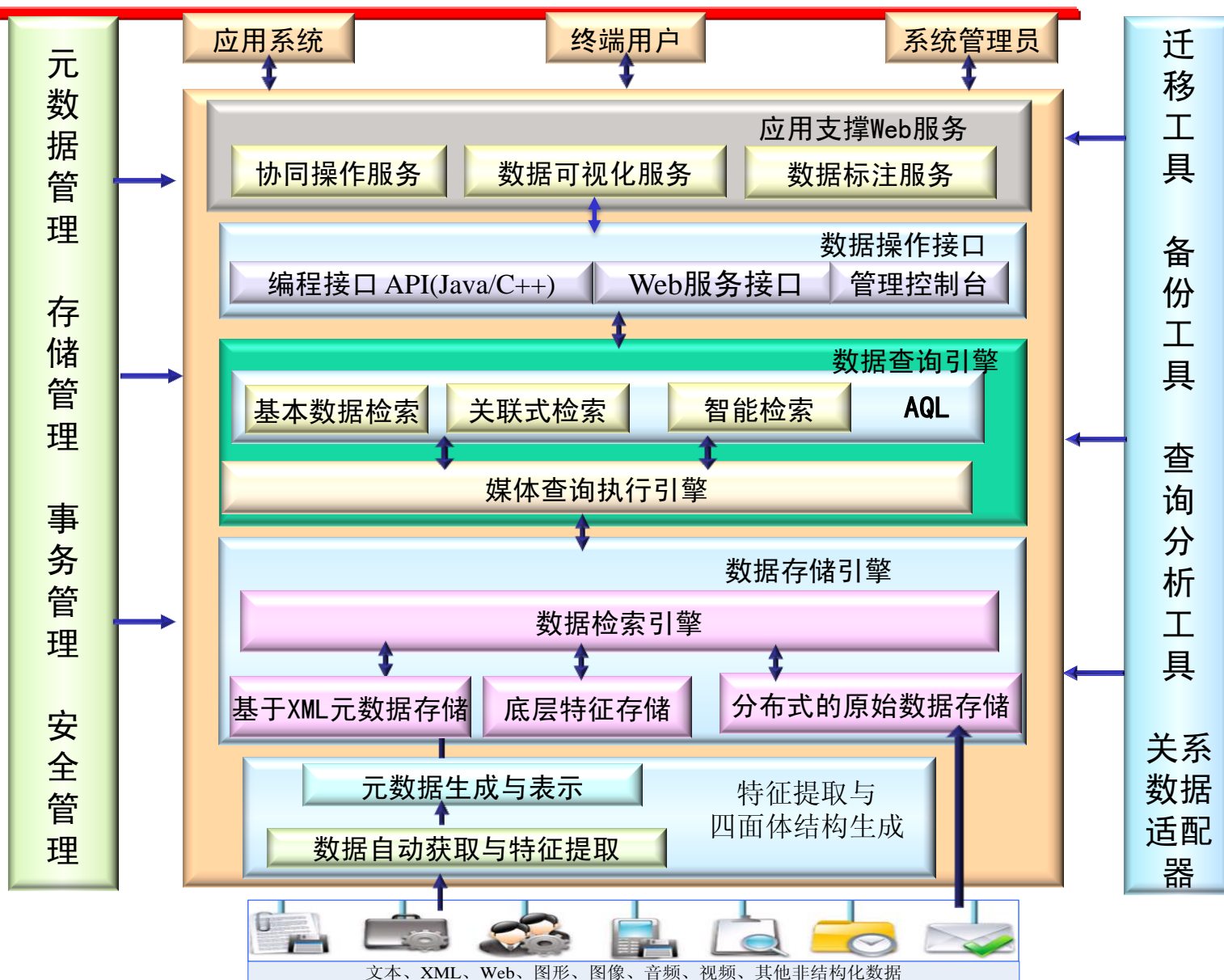
➤ 优点:

- ◆ 统一性
- ◆ 集成性
- ◆ 关联性
- ◆ 扩展性

$Tetrahedron = (V, BA\_FACET, SF\_FACET, LF\_FACET, RD\_FACET, CONUS)$   
 $BA\_FACET = \{Basic\_Attribute\}$   
 $SF\_FACET = \{Semantic\_Feature_j \mid j \in [1, m]\}$   
 $LF\_FACET = \{Lowlevel\_Feature_k \mid k \in [1, m]\}$   
 $RD\_FACET = \{Data_l \mid l \in [1, p]\}$   
 $CONUS = (BA\_FACET \times SF\_FACET \cup BA\_FACET \times LF\_FACET \cup BA\_FACET \times RD\_FACET$   
 $\cup SF\_FACET \times LF\_FACET \cup SF\_FACET \times RD\_FACET \cup LF\_FACET \times RD\_FACET)$

# 非结构化数据管理系统——AUDR

- 组件化、多引擎的体系结构
- 优点：
  - ◆ 可扩展
  - ◆ 可定制

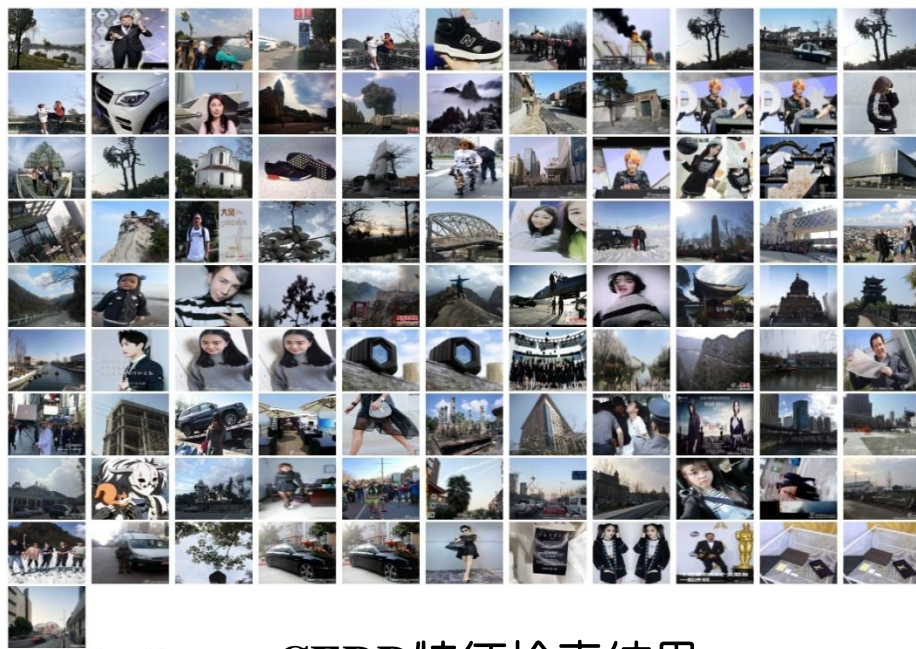




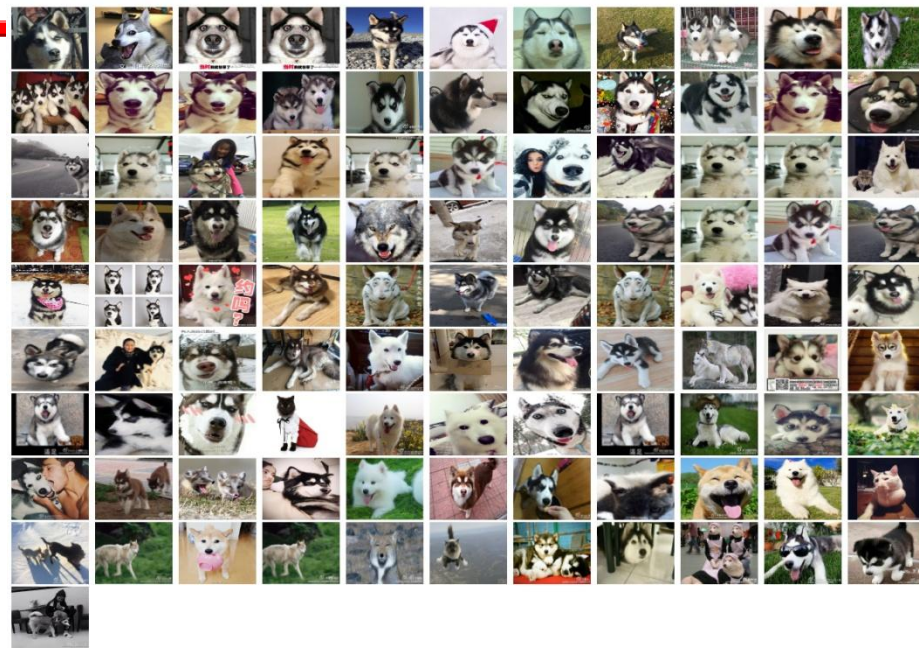
# AUDR 中的图像实例检索



检索用例图



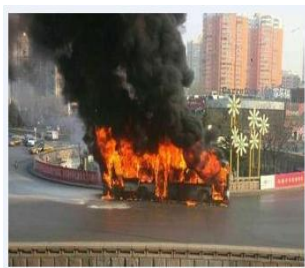
CEDD特征检索结果



VGG-F特征检索结果

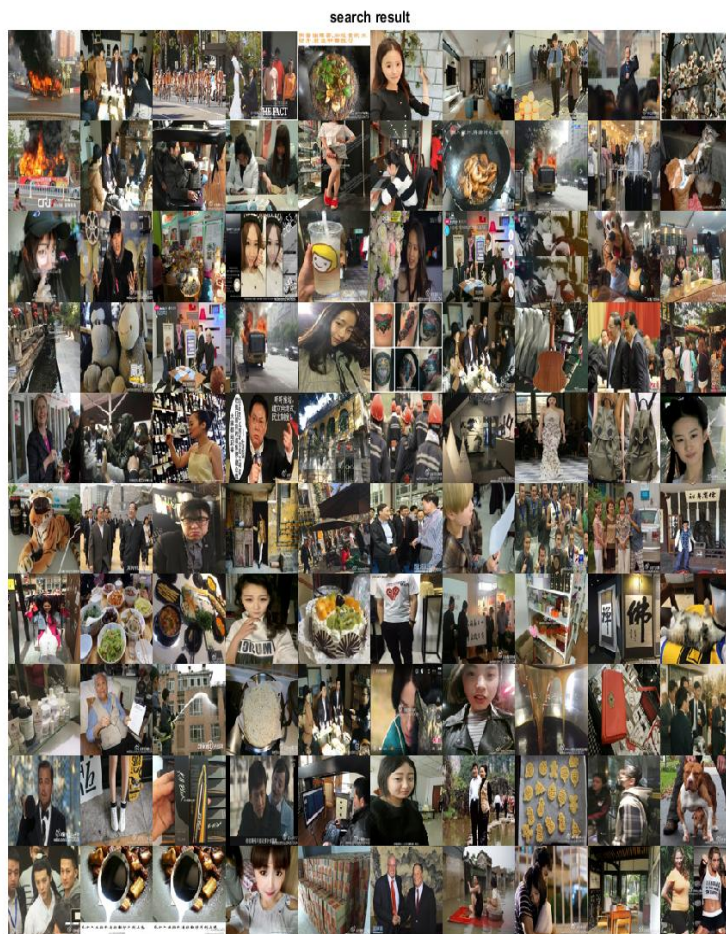


# AUDR 中的图像实例检索



检索用例图

图像库数据规  
10万



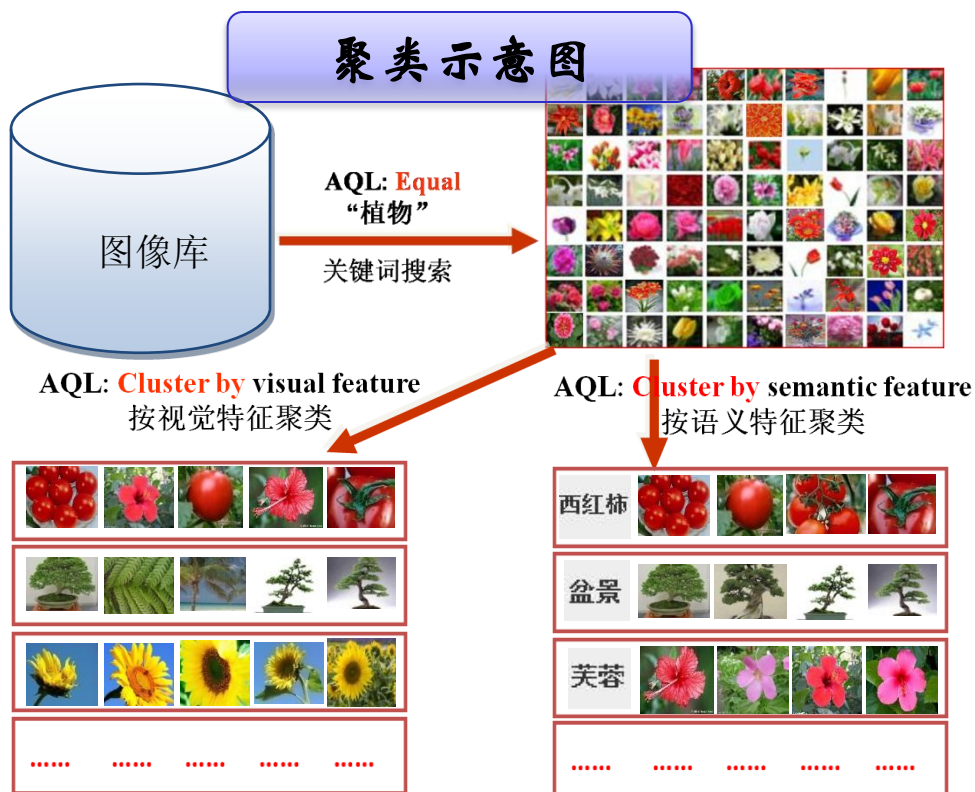
CEDD特征检索结果



VGG-F特征检索结果

# 数据聚类检索

## 聚类示意图



## 按视觉特征聚类结果





# 数据管理技术的新发展

---

- SQL

- 传统关系型数据库，支持SQL操作、事务ACID特性
- 几千用户，TB级数据

- NoSQL

- Not only SQL,非关系型的数据库，水平可扩展、分布式
- 不使用SQL，不支持事务的ACID操作
- HBase、MongoDB等

- NewSQL

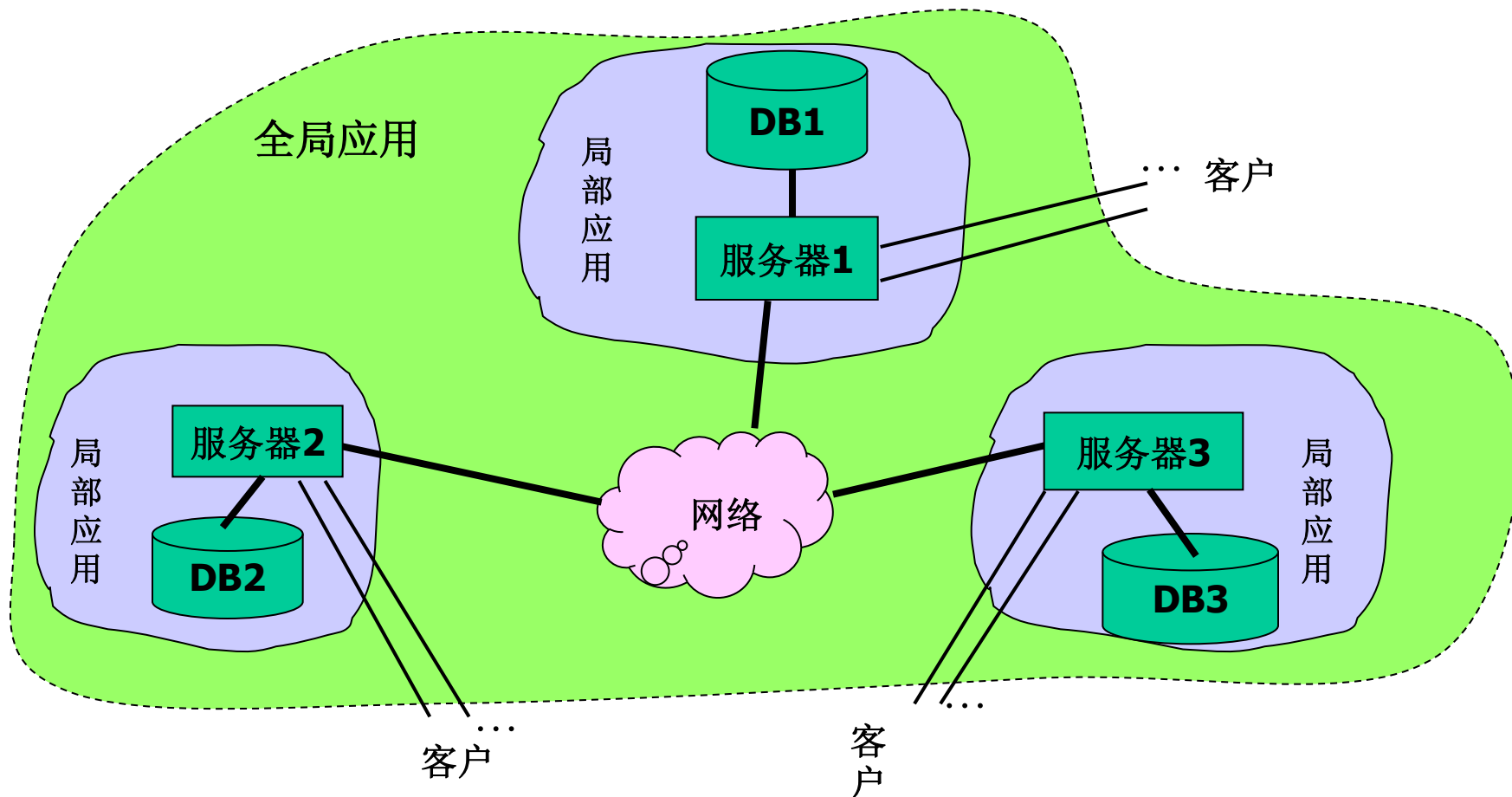
- 新的可扩展/高性能数据库
- 不仅具有NoSQL的海量数据存储管理能力，还保持了传统数据库支持ACID和SQL等特性
- VoltDB、ScaleBase、阿里DRDS等

# 第九章 分布式数据库系统

---

- 分布式数据库系统基本概念
  - 什么是分布式数据库系统
  - 分布式数据库系统的特点
- 分布式数据库系统体系结构
- 分布式数据库系统主要技术

# 什么是分布式数据库系统



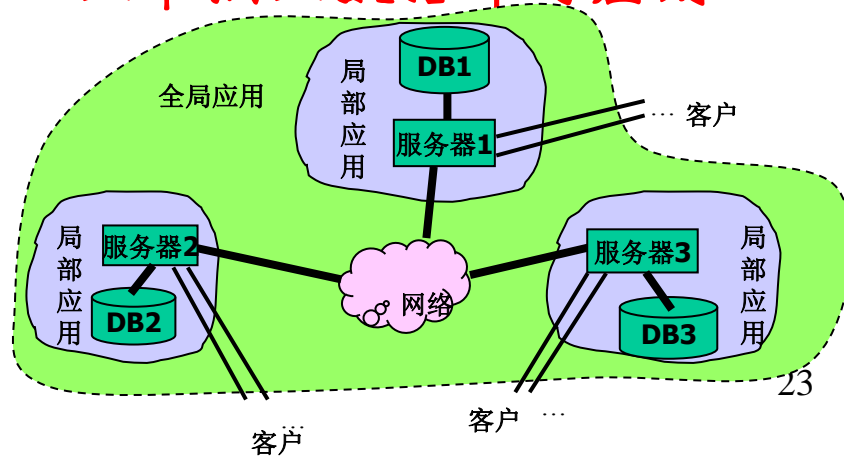
- **分布性**—数据分布存储在网络的各个节点上。
- **逻辑上的整体性**—数据被一种机制联系在一起，构成一个有机整体。

# 什么是分布式数据库系统

- 分布式数据库定义

- 分布式数据库是由一组分布在计算机网络的不同结点上的数据组成，每个结点具有独立处理的能力（称为场地自治），可以执行局部应用，同时每个结点也能通过网络通信支持全局应用。

- 局部应用：只操作一个结点上数据库的应用
    - 全局应用：操作两个或两个以上节点上数据库的应用



# 什么是分布式数据库系统

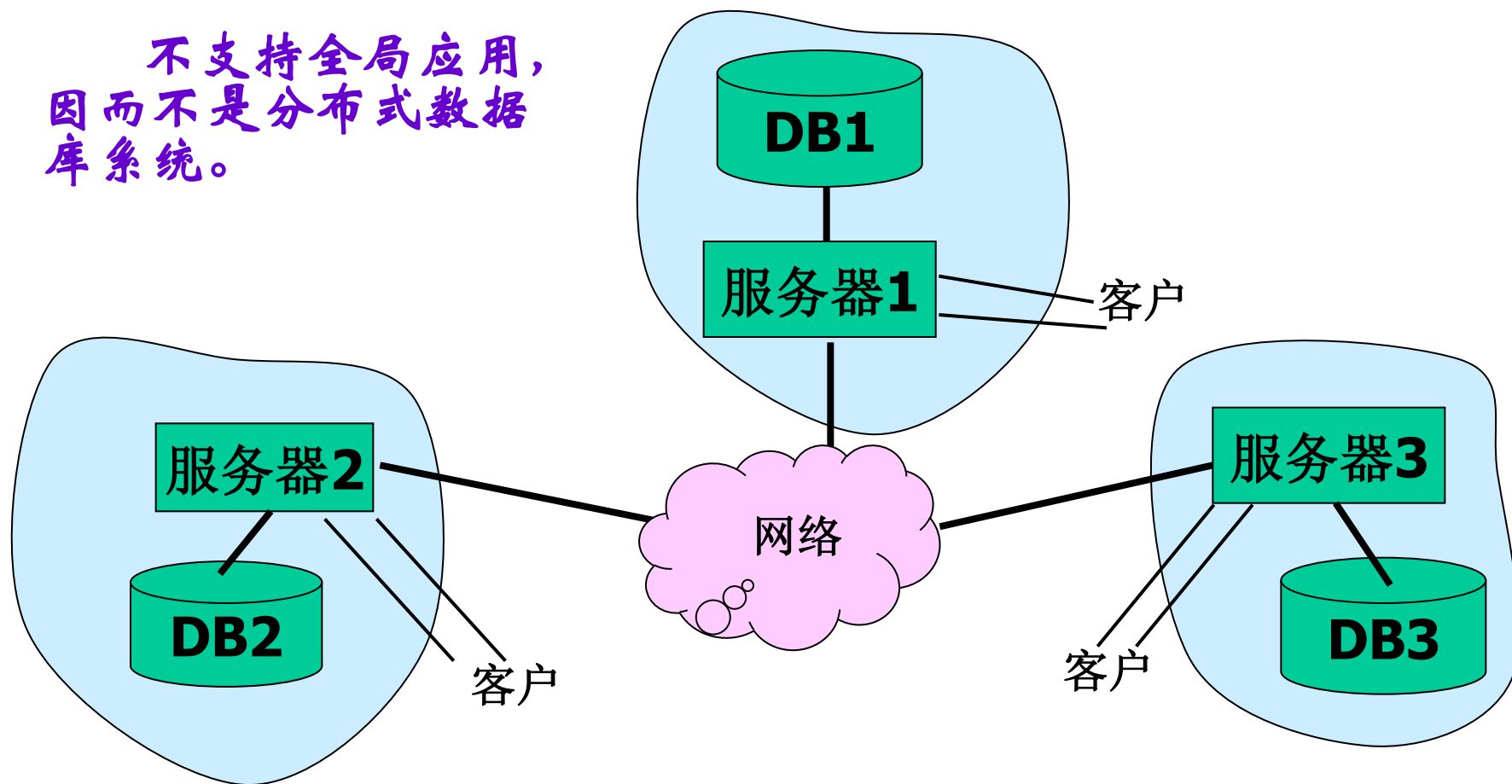
---

- 分布式数据库以“数据分布”为前提，强调场地自治性（局部应用）以及自治场地之间的协作性（全局应用），两者缺一不可。
  - 场地自治性：每个场地有自己的数据库、一组终端、运行局部DBMS，是独立的DBS，具有高度自治性。
  - 自治场地之间的协作性：各结点组成整体。整体性的含义是，从用户角度看，分布式数据库系统逻辑上如同一个集中式数据库一样，用户可以在任何场地执行全局应用。

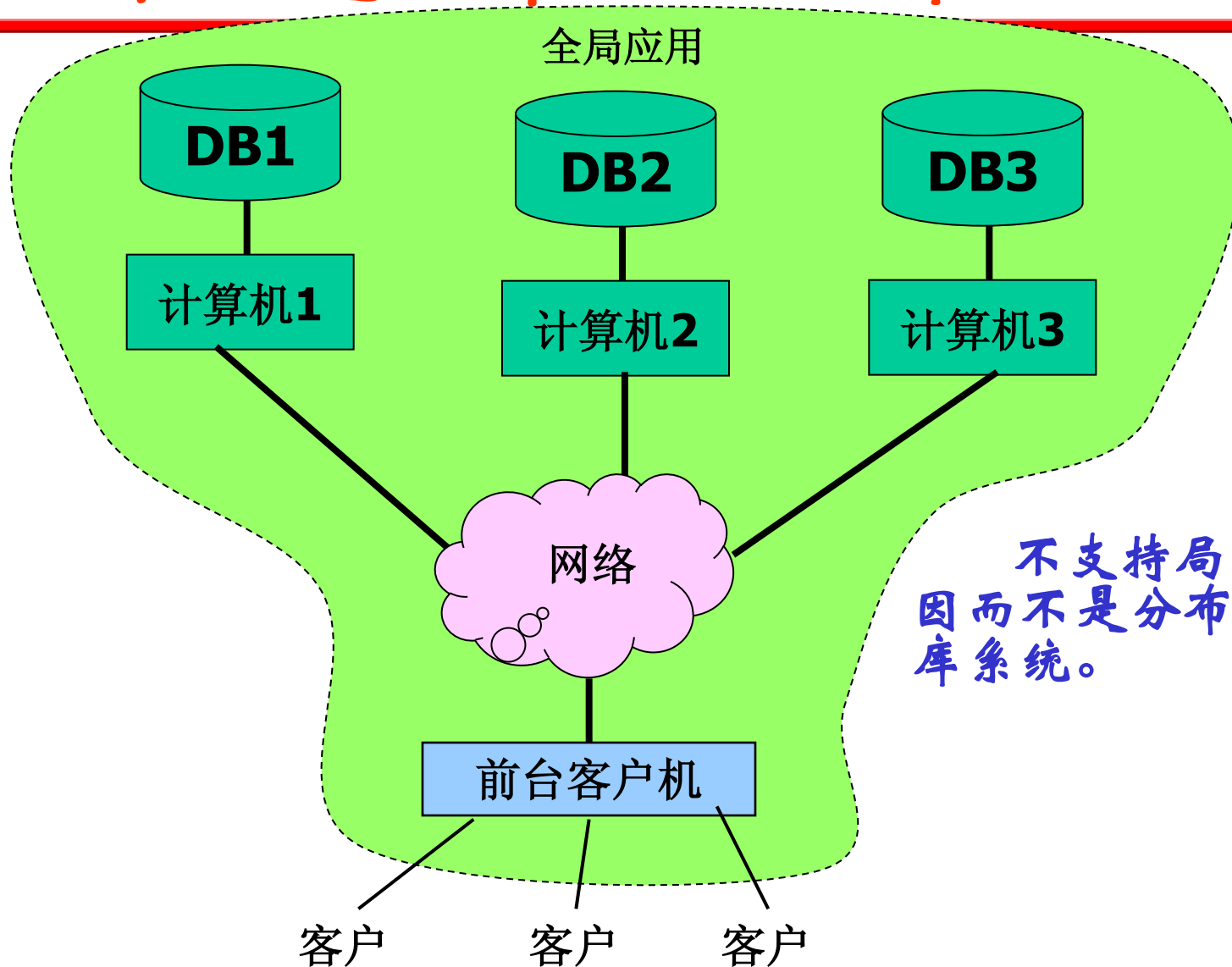


# 什么是分布式数据库系统

不支持全局应用，  
因而不是分布式数据库系统。



# 什么是分布式数据库系统

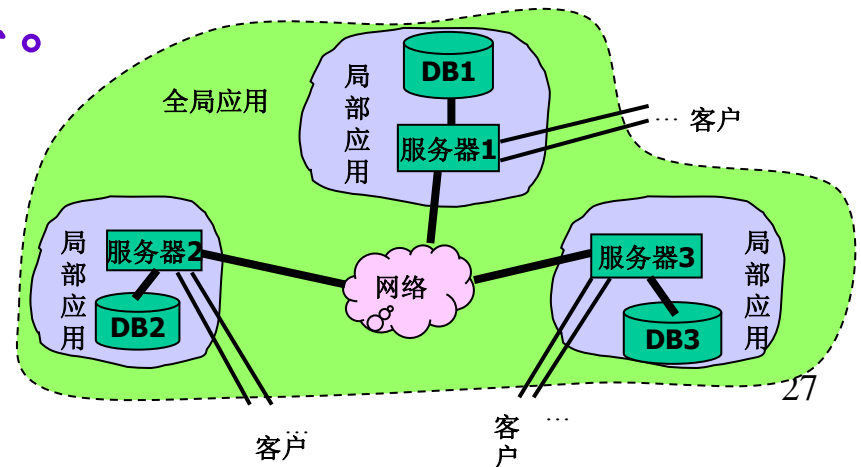


# 分布式数据库系统的特点 (1)

- 分布式数据库系统是在集中式数据库系统技术的基础上发展起来的，它具有自己独特的特征和性质。

## (1) 数据独立性

- 数据的逻辑独立性和物理独立性
- 数据的**分布独立性**（也称**分布透明性**）：数据的逻辑分片、数据物理位置分布的细节、重复副本（冗余数据）一致性问题、局部结点上的数据模型等与用户程序无关。



# 分布式数据库系统的特点 (2)

---

## (2) 集中与自治相结合的控制结构

- 数据的共享有两个层次：
  - 一是**局部共享**，即在局部数据库中存储局部结点各用户的共享数据；
  - 二是**全局共享**。即在分布式数据库系统的各个结点也存储供其他结点的用户共享的数据，支持系统的全局应用。
- 分布式数据库系统常常采用集中和自治相结合的控制结构
  - 各局部的DBMS可以独立的管理局部的数据库，具有**自治功能**。
  - 系统又设有**集中控制结构**，协调各局部DBMS的工作，执行全局应用。

# 分布式数据库系统的特点 (3)

---

## (3) 适当增加数据冗余

- 在分布式数据库系统中适当的增加了冗余数据，在不同的结点存储同一数据的多个副本：
  - 提高系统的可靠性、可用性，当某一结点出现故障时，系统可以对另一结点的相同副本进行操作，不会因为一处故障而造成整个系统的瘫痪；
  - 提高系统性能，系统可以选择用户最近的数据副本来进行操作，减少通信代价，改善整个系统的性能。
- 不利于更新，增加了系统维护代价。

# 分布式数据库系统的特点 (4)

---

## (4) 全局的一致性、可串行性和可恢复性

- 分布式数据库系统除了各局部数据库应满足集中式数据库的一致性、可串行性和可恢复性以外，还应保证数据库的**全局一致性、并行操作的可串行性和系统的全局可恢复性**。

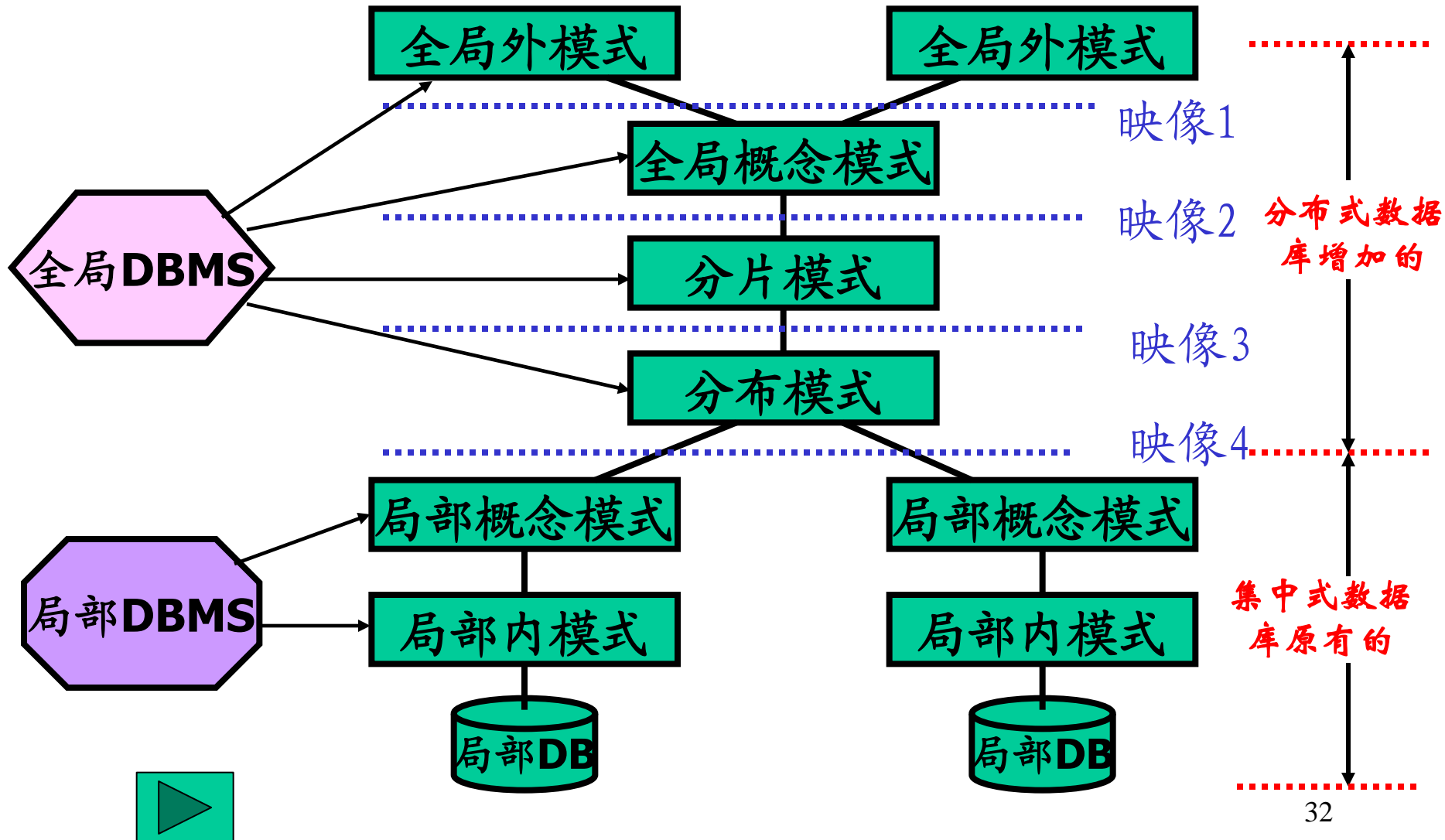


# 分布式数据库系统的体系结构

---

- 分布式数据库系统的模式结构
- 数据分片
- 分布透明性
- 分布式数据库管理系统DDBMS

# 分布式数据库系统的模式结构





# 分布式数据库系统的模式结构

- 全局外模式及全局外模式/全局概念模式映像（映像1）

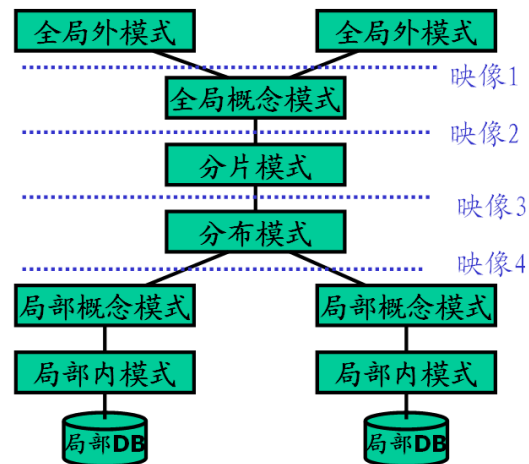
全局外模式全局应用的用户视图，是全局概念模式的子集；映像1定义全局外模式到全局概念模式的映像。

- 全局概念模式

定义分布式数据库中数据的整体逻辑结构，使得数据如同没有分布一样。

- 分片模式及全局概念模式/分片模式映象（映像2）

每一个全局关系可以分为若干互不相交的部分，每一部分称为一个片段。分片模式及映像2定义片段以及全局关系到片段的映象。这种映象是一对多的。



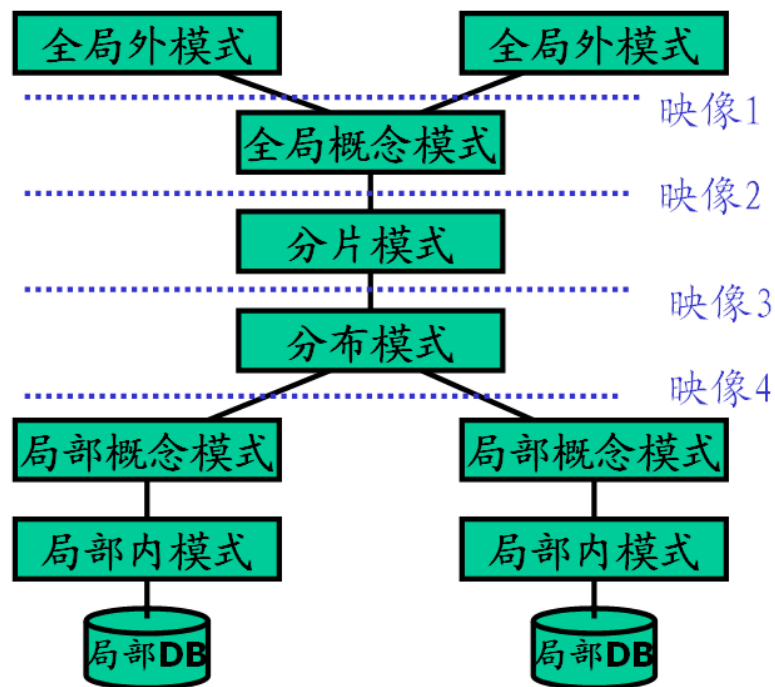
# 分布式数据库系统的模式结构

- 分布模式及分片模式/分布模式映像（映像3）

定义片段的存放结点及片段到节点的映像。分布模式的映像类型确定了分布式数据库是冗余的还是非冗余的。

- 分布模式/局部数据库概念模式映像（映像4）

该映像把存储在局部场地的全局关系或全局关系的片段映像为各局部概念模式。



# 数据分片

---

- 分片的方式有多种：
  - 水平分片，垂直分片——两种基本的分片方式
  - 混合分片，导出分片——较复杂的分片方式

# 数据分片方式

---

- 水平分片

- 将关系依照一定条件按行分为不相交的若干子集，每个子集称为一个水平片段。

- 垂直分片

- 将关系按列分为若干属性子集，每个子集称为一个垂直片段。垂直分片的片段通过连接的方法恢复原关系。因此垂直分片的诸片段通常都包含关系的码。

- 导出分片

- 导出水平分片，分片的条件不是关系本身属性条件，而是其它关系的属性条件。如SC (SNO, CNO, G) 按S关系中学生所在的系分片。

- 混合分片

- 指按上述三种分片方式得到的片段，继续按另一种方式分片。

# 数据分片的约束

无论哪种分片方式，都应满足以下条件：

- 完全性

- 一个全局关系中的数据必须完全划分为若干片段，不允许某些数据属于全局关系但不属于任何片段。

- 不相交性

- 不允许一个全局关系的某些数据既属于该全局关系的某一个片段，又属于另一个片段（垂直分片的码属性除外）。

- 可重构性

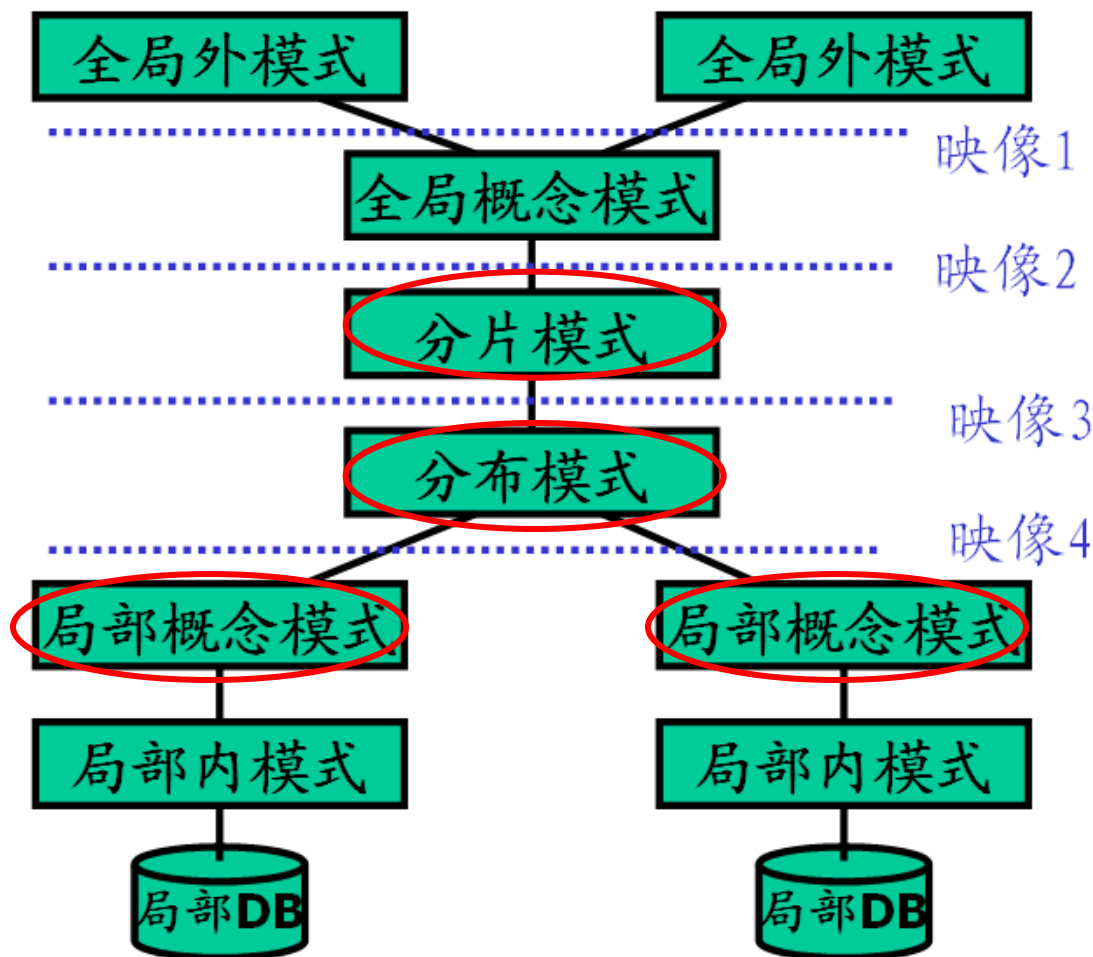
- 可以由片段重构全局关系

- 垂直分片用连接操作重构，例R的码为A，R1，R2，R3为全局关系R的垂直分片，且都包含A，则  $R = R1 \bowtie R2 \bowtie R3$
- 水平分片用并操作重构，例SC\_A, SC\_B为SC的两个水平分片，则  $SC = SC\_A \cup SC\_B$



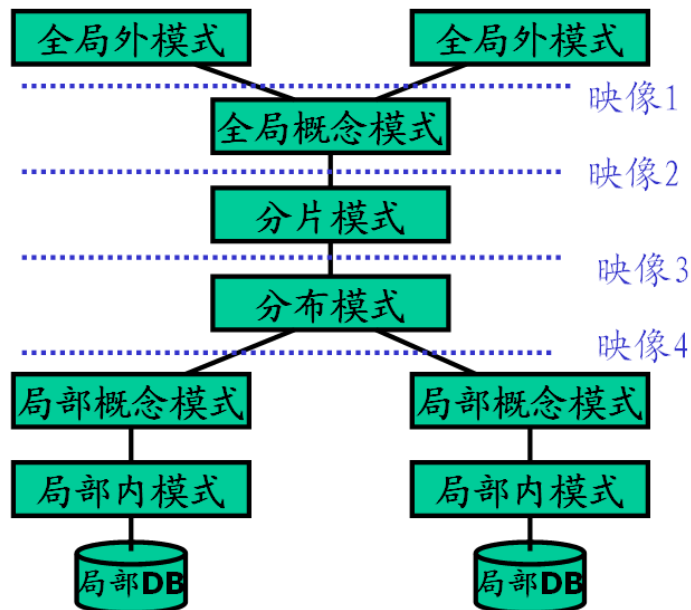
# 分布透明性 (1)

分布透明性（分布独立性）包括：分片透明性，位置透明性，局部数据模型透明性



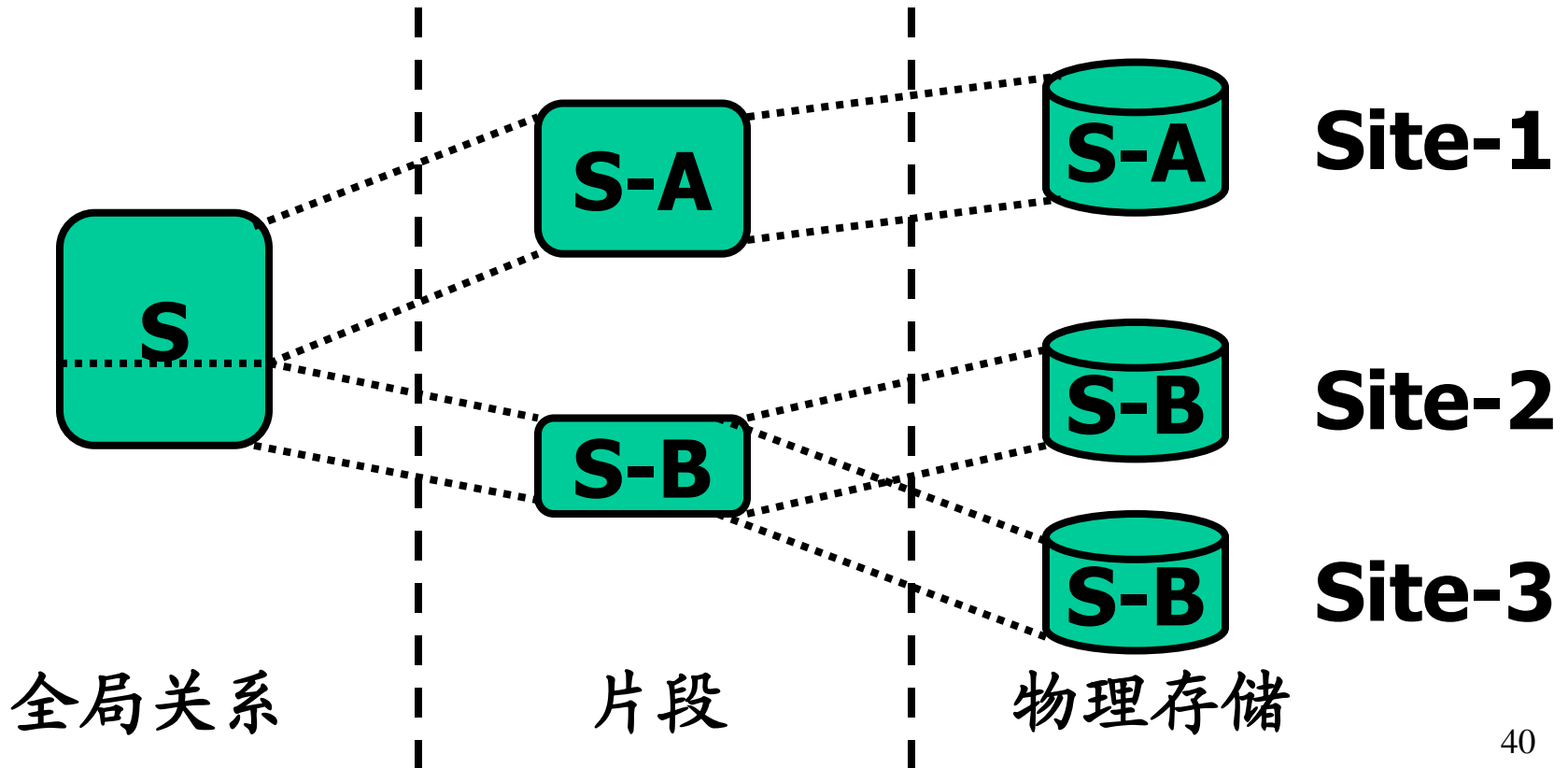
# 分布透明性 (2)

- **分片透明性**——用户或应用程序只对全局关系进行操作而不必考虑关系的分片。它是分布透明性的最高层次。
- **位置透明性**——用户或应用程序不必了解片段的存储场地也不必关心各数据副本的一致性。
- **局部数据模型透明性**——用户或应用程序不必了解局部场地上使用的是哪种数据模型。模型的转换以及查询语言等的转换均由分布模式/局部概念模式（映象4）完成。



# 分布透明性示例

- 示例：设有全局关系  $S$  ( $S\#, SN, SD, SA$ )，它被划分为两个片段  $S-A$  (本科生)， $S-B$  (研究生)， $S-B$  有两个副本。





# 分布透明性示例

执行查询操作：检索所有学生的学号和年龄。

- 若系统具有分片透明性，则执行操作：

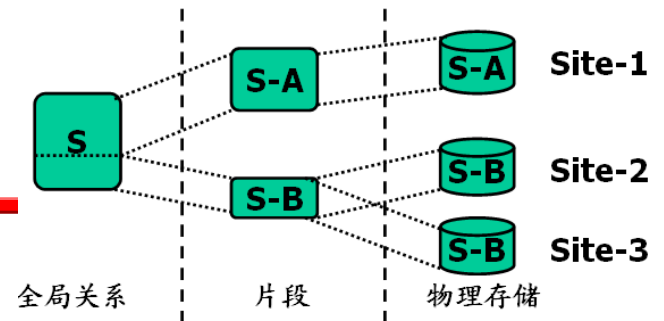
```
SELECT S#,SA  
FROM S
```

- 若系统具有位置透明性，不具有分片透明性，则可执行操作：

```
SELECT S#, SA  
FROM S_A  
UNION  
SELECT S#, SA  
FROM S_B
```

- 若系统只具有局部数据模型的透明性，不具有位置透明性与分片透明性，在执行操作：

```
SELECT S#, SA  
FROM S_A AT SITE-1  
UNION  
SELECT S#, SA  
FROM S_B AT SITE-3
```

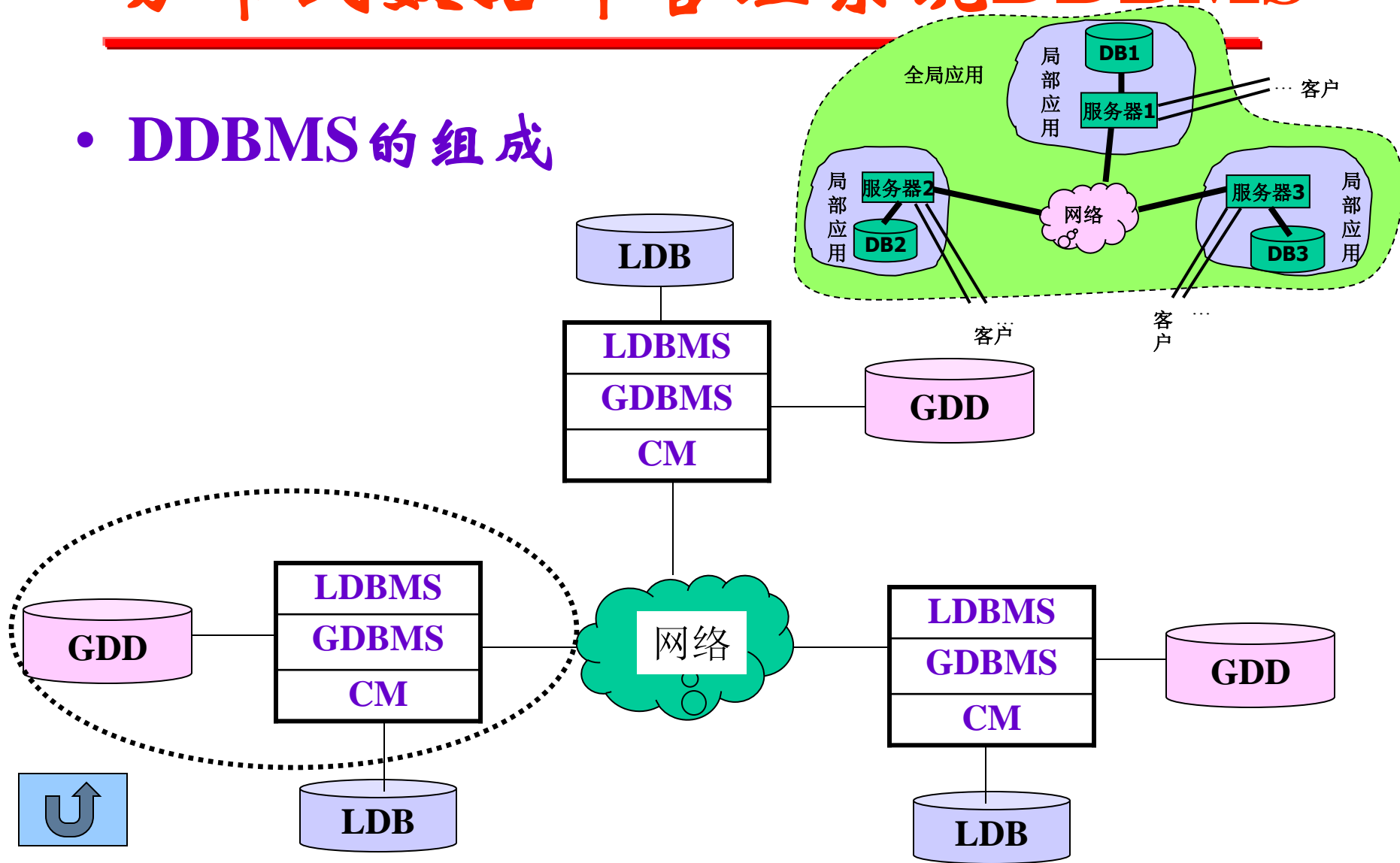


$S(S\#, SN, SD, SA)$

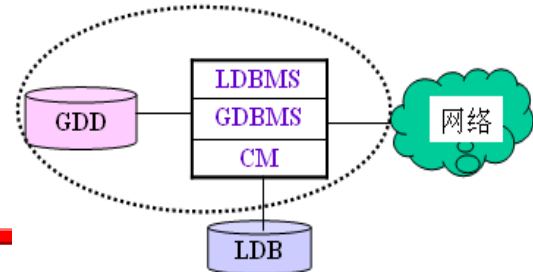


# 分布式数据库管理系统DDBMS

- DDBMS的组成



# DDBMS的组成



## — LDBMS(局部场地上的数据库管理系统)

- 功能：建立和管理局部数据库，提供场地自治能力，执行局部应用及全局查询的子查询。

## — GDBMS（全局数据库管理系统）

- 功能：提供分布透明性，协调全局事务的执行，协调各局部DBMS以完成全局应用，并保证数据库的全局一致性，执行并发控制，实现更新同步，提供全局恢复功能。

## — GDD（全局数据字典）

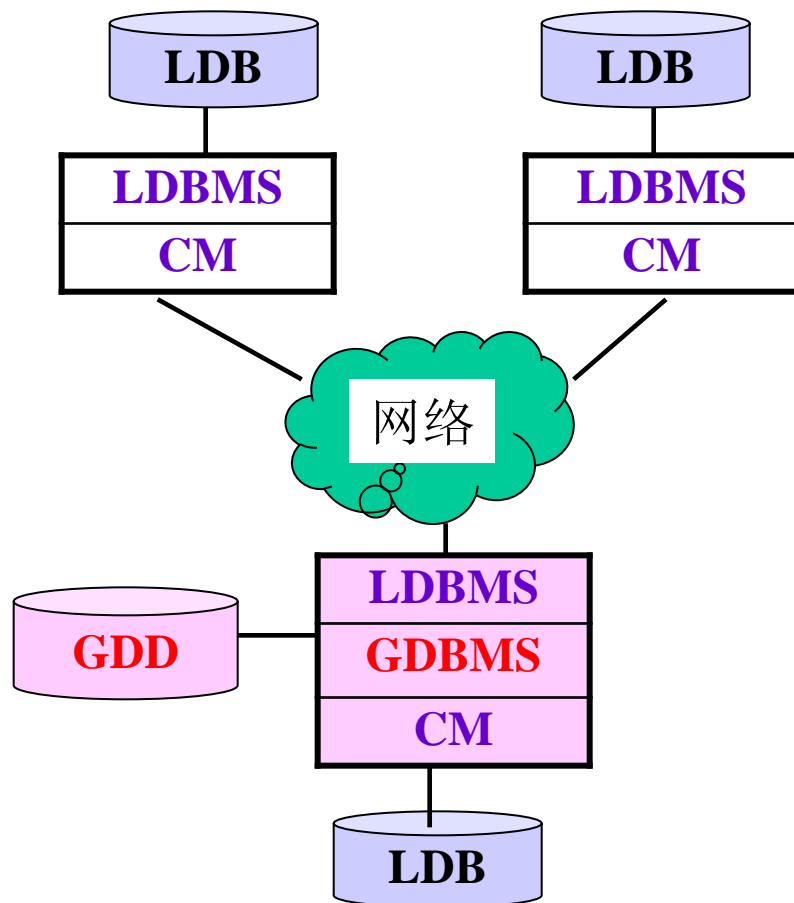
- 存放全局概念模式、分片模式、分布模式的定义以及各模式之间映象的定义。
- 存放有关用户存取权限的定义，以保证全局用户的合法权限和数据库的安全性。
- 存放数据完整性约束条件的定义。

## — CM（通信管理）

- 在分布式数据库各场地之间传递消息和数据，完成通信功能。

# DDBMS的分类 (1)

- 按全局控制方式分类
  - 全局控制集中的DDBMS



- 特点：** GDBMS集中在某一结点上，GDD只有一个，也放在该结点上。
- 优点：**
  - 控制简单，容易设计实现
- 缺点：**
  - 易形成瓶颈，并且一旦该结点出现故障，整个系统将瘫痪。

# DDBMS 的分类 (2)

---

## — 全局控制分散的DDBMS

- **特点：** GDBMS分散在每一个结点上。GDD 也在每个结点上有一份。这类结构称为**完全分布的DDBMS**。
- **优点：** 结点独立，自治性强，单个结点出现问题不会使系统瘫痪；
- **缺点：** 全局控制的协调机制和一致性维护都比较复杂。

## — 全局控制部分分散的DDBMS

- 根据应用的需要将全局数据库管理器和全局数据字典分散在某些结点上。介于上述两者之间的体系结构。

# DDBMS 的分类 (3)

---

- 按局部DBMS的类型分类
  - 同构型DDBMS
    - 每个结点的局部数据库具有相同的DBMS，即使硬件与操作系统不相同。
  - 异构型DDBMS
    - 各结点的局部数据库具有不同的DBMS



# 分布式数据库系统的主要技术

---

- 分布式查询处理和优化
- 分布事务管理

# 分布式查询处理和优化

---

分布式数据库系统中查询处理较集中式数据库复杂，查询优化较集中数据库更重要，效果更显著。

- 分布式查询类型与处理过程
- 查询优化要解决的问题
- 查询优化的目标
- 连接查询的优化



# 分布式查询类型

---

- 在分布式数据库系统中，查询可分为三类：局部查询，远程查询，全局查询。
  - 局部查询和远程查询只涉及单个结点的数据（本地的或远程的），可以采用集中式数据库的处理技术；
  - 全局查询涉及到多个结点的数据，十分复杂。

# 分布式查询处理过程

---

- 为了执行全局查询和确定一个好的查询策略，要做许多判断、计算工作。
  - 查询分解
    - 把全局查询分解为若干子查询，每个子查询只涉及某一个结点的数据,可由局部DBMS处理。必须选择查询开销最省的那些结点（物理片段）。
  - 选择操作执行的次序
    - 主要是确定涉及不同结点上关系的连接和并操作的次序。
  - 选择执行操作的方法
    - 包括选择存取路径、选择某种操作的算法以及连接的执行方法。

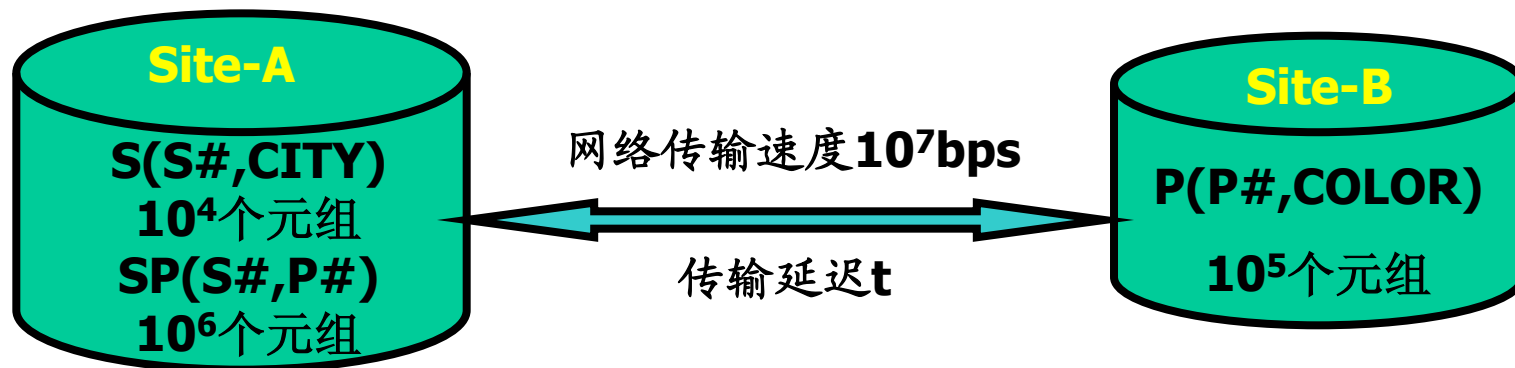
# 查询优化的目标

---

- 查询处理策略的选择是以执行查询的预期代价为依据的
  - 查询执行的开销：I/O代价+CPU代价+通信代价
- 分布式查询优化可分为查询策略的分布优化和局部优化，其中分布优化更重要

# 分布式查询示例

• 例

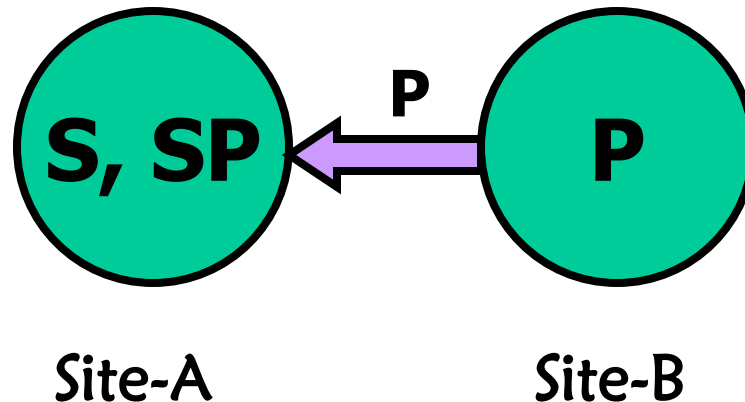


- 数据库：
  - Site-A: 供应商库S, 供应商装运单库SP; Site-B: 产品库P;
  - 每个关系的元组均为100byte长;
- 估算值: 红色零件数=10; 北京供应商的装运单数=10<sup>5</sup>;
- 传送时间T=总传输延迟+总数据量/传输速度
- 查询: 求供应红色零件的北京的供应商号。

*select S.S# from S, P, SP  
where S.S# = SP.S# and SP.P# = P.P#  
and COLOR = '红色' and CITY = '北京' ;*

# 6种可能的查询策略

1. 把关系P从B站传送到A站，在A站进行查询

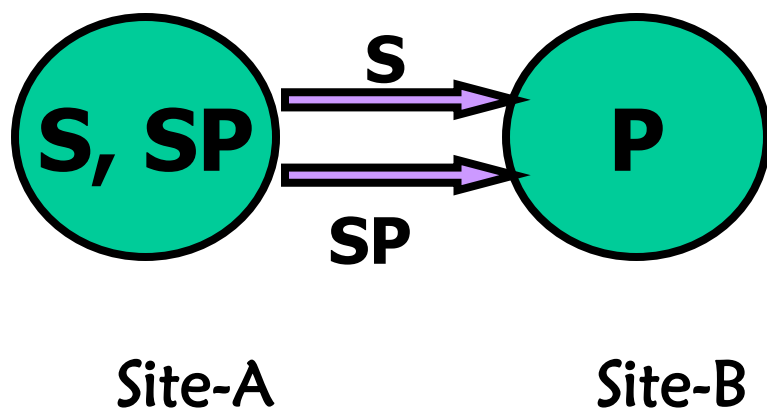


S(S#,CITY)	10 <sup>4</sup> 个元组
P(P#,COLOR)	10 <sup>5</sup> 个元组
SP(S#,P#)	10 <sup>6</sup> 个元组

红色零件数=10;  
北京供应商的装运单数=10<sup>5</sup>

$$\text{传送时间 } T = 1t + 10^5 * 100 / 10^7 = (t+1)s$$

## 2.把关系S, SP从A站传送到B站, 在B站进行查询

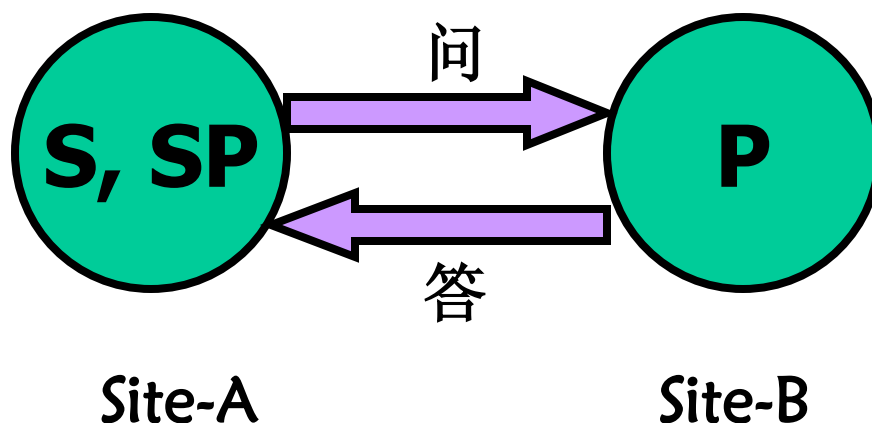


<b>S(S#,CITY)</b>	<b>10<sup>4</sup>个元组</b>
<b>P(P#,COLOR)</b>	<b>10<sup>5</sup>个元组</b>
<b>SP(S#,P#)</b>	<b>10<sup>6</sup>个元组</b>

红色零件数=10;  
北京供应商的装运单数=10<sup>5</sup>

$$\text{传送时间 } T = 2t + (10^4 + 10^6) * 100 / 10^7 \approx (2t + 10)s$$

3. 在A站连接S与SP，选出城市为北京的元组（有 $10^5$ 个），然后对其中每个元组的P#，询问B站，看其是否为红色。

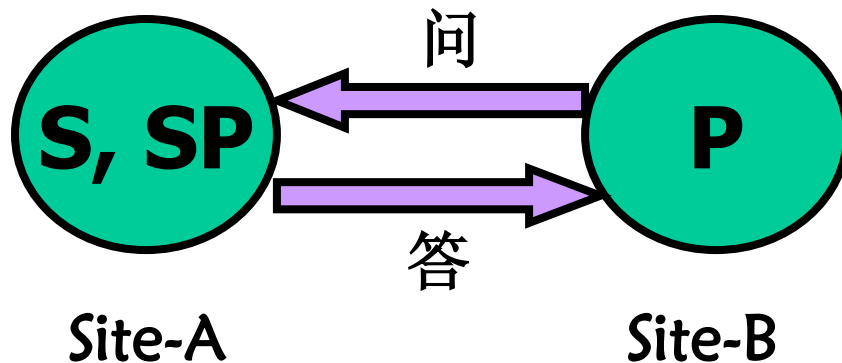


<b>S(S#,CITY)</b>	<b>10<sup>4</sup>个元组</b>
<b>P(P#,COLOR)</b>	<b>10<sup>5</sup>个元组</b>
<b>SP(S#,P#)</b>	<b>10<sup>6</sup>个元组</b>

红色零件数=10;  
北京供应商的装运单数=10<sup>5</sup>

传送时间  $T = 2t * 10^5s$

4. 在B站选出红色零件（有10个），然后对每个元组询问A站，看北京的供应商是否供应此零件。



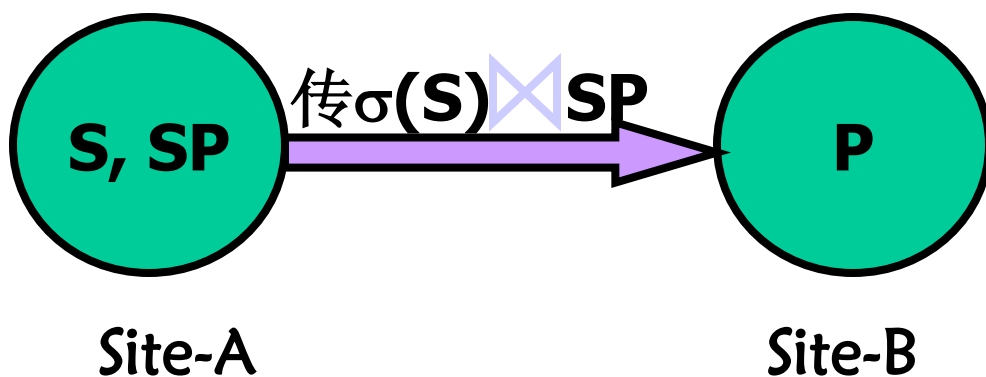
<b>S(S#,CITY)</b>	<b>10<sup>4</sup>个元组</b>
<b>P(P#,COLOR)</b>	<b>10<sup>5</sup>个元组</b>
<b>SP(S#,P#)</b>	<b>10<sup>6</sup>个元组</b>

红色零件数=10;  
北京供应商的装运单数=10<sup>5</sup>

$$\text{传送时间 } T = 2 t * 10 = 20t_s$$



5.在A站选出北京的供应商的装运单 ( $10^5$  个), 传送到B站, 在B站完成查询。

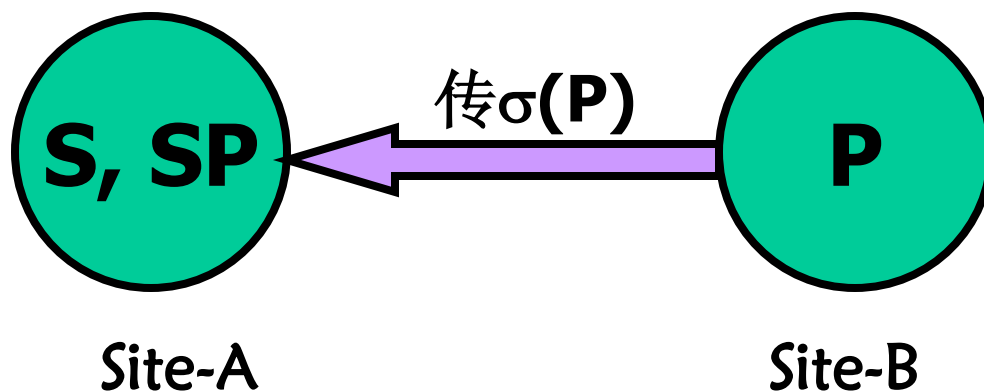


$S(S\#,CITY)$	$10^4$ 个元组
$P(P\#,COLOR)$	$10^5$ 个元组
$SP(S\#,P\#)$	$10^6$ 个元组

红色零件数=10;  
北京供应商的装运单数= $10^5$

$$\text{传送时间 } T = 1t + 10^5 * 100 / 10^7 = (t+1)s$$

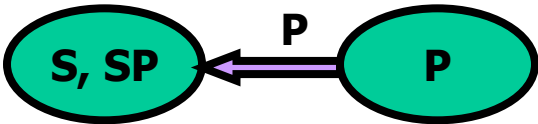
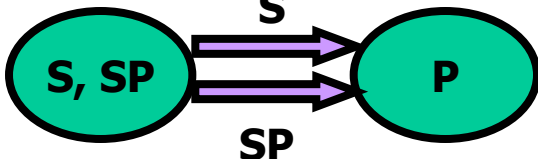
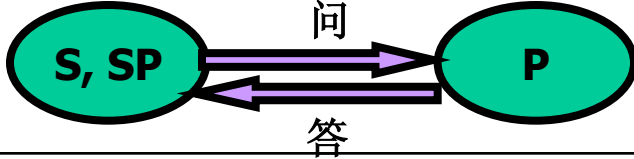
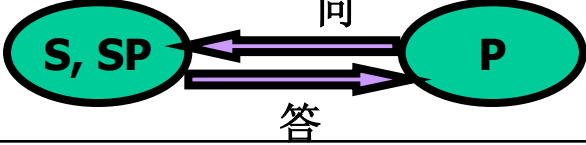

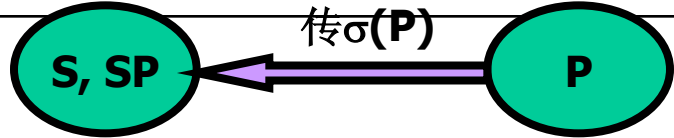
6. 在B站选出红色零件（10个），把结果传送到A站，在A站完成查询。



<b>S(S#,CITY)</b>	<b>10<sup>4</sup>个元组</b>
<b>P(P#,COLOR)</b>	<b>10<sup>5</sup>个元组</b>
<b>SP(S#,P#)</b>	<b>10<sup>6</sup>个元组</b>

红色零件数=10;  
北京供应商的装运单数=10<sup>5</sup>

$$\text{传送时间 } T = 1t + 10 * 100 / 10^7 \approx 1ts$$

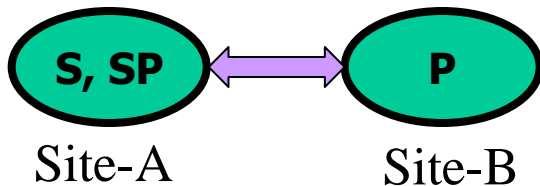
序号	示意图	$T(1s, 10kbps)$	$T(t, 10Mbps) (t=10^{-3}s)$
1		$10^3 s$ (16.7分)	$(t+1)$ (1s)
2		$10^4 s$ (28小时)	$(2t+10)$ (10s)
3		$2 * 10^5 s$ (2.3天)	$2t * 10^5$ (200s)
4		20s	20t (0.02s)
5		$10^3 s$ (16.7分)	$(t+1)$ (1s)
6		1.1s	1t ( $10^{-3}s$ )

- 
- 不同的查询策略通信时间相差很大，必须进行优化；
  - 有些策略中数据传输时间和传输延迟都要考虑，而另一些策略主要考虑传输延迟（如问答方式），还有一些策略数据传输量大，主要考虑传输时间。
- 
- 传送时间  $T = \text{总传输延迟} + \text{总数据量} / \text{传输速度}$

# 查询优化的目标

- 分布式数据库查询优化中，将“通信代价”作为首要因素进行研究。因此，查询优化的首要目标是：使查询执行时通信代价最省。
- 通信代价可计算，通常是数据传输量的函数：
$$TC(X) = C_0 + X * C_1$$

$C_0$ 为两结点初始化一次传输所花费的开销， $X$ 为数据传输量， $C_1$ 为传输率，即单位数据传输所花费的时间。
- 不同结点之间的连接操作和并操作是数据传输的主要原因，因此连接查询的优化是优化中研究的重要问题。



```
select S.S# from S, P, SP
where S.S# = SP.S# and SP.P# = P.P#
and COLOR = '红色'
and CITY = '北京' ;
```

# 连接查询的优化

- **半连接**：使用半连接来缩减关系（或片段）进而节省传输开销。
- 半连接法定义： $R \bowtie_{A=B} S = R \bowtie_{A=B} (\Pi_B(S))$

例：

R1

A	B
a1	b1
a2	b1
a2	b3
a2	b4
a3	b3

$\bowtie$

R2

B	C
b1	c1
b2	c2
b5	c1
b5	c2
b6	c4
b7	c2
b8	c3

=

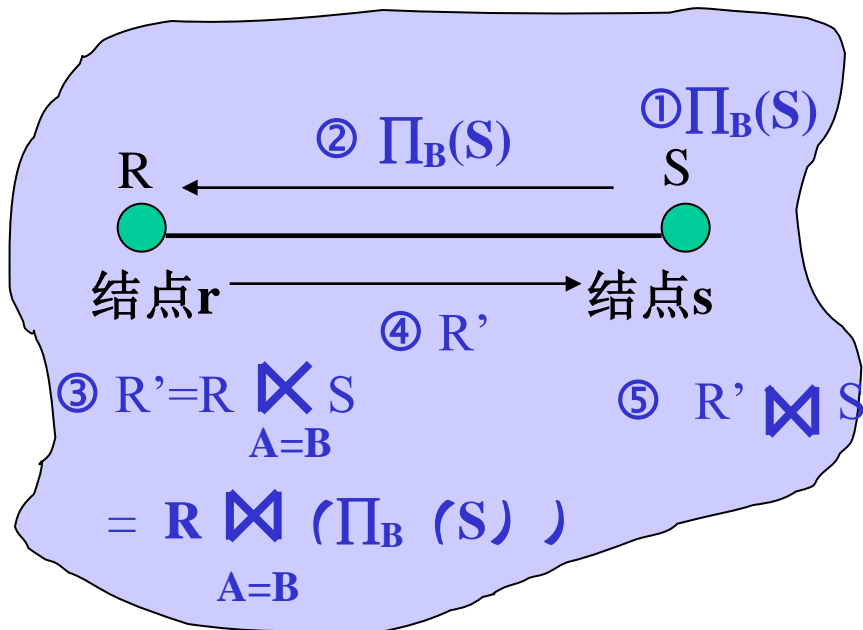
$R3 = R1 \bowtie_{A=B} R2$

A	B
a1	b1
a2	b1

- 根据半连接定义，有

$$R \bowtie_{A=B} S = (R \bowtie_{A=B} S) \bowtie_{A=B} S = (R \bowtie_{A=B} \Pi_B(S)) \bowtie_{A=B} S$$

- 半连接在查询优化中的应用



—采用半连接的通信代价：

$$C_{SJ} = 2C_0 + C_1 *$$

$$(\text{size}(B) * \text{card}(\Pi_B(S)) + \text{size}(R) * \text{card}(R'))$$

—不采用半连接的通讯代价：

$$C_{JN} = C_0 + C_1 * \text{size}(R) * \text{card}(R)$$

—当R中参与连接的元组足够少时  
采用半连接策略是有利的。

# 分布事务处理

---

- 分布事务的原子性与可串行性
  - 在分布式数据库系统中，一个全局事务被划分为在许多结点上的子事务。
  - 分布事务的原子性是：组成该事务的所有子事务要么一致地全部提交，要么一致地全部回滚。
  - 在多用户系统中，还必须保证分布式事务的可串行性。
- 分布事务管理主要包括：事务的恢复和并发控制。



# 分布事务的恢复

---

- 每个场地都有一个**局部事务管理器**，负责管理局部子事务的执行。同时，各**局部事务管理器之间必须相互协调**，保证分布事务的**原子性**：各子事务要么都提交，要么都回滚。
- 对局部事务管理器进行协调，保证分布事务原子性最常用的技术——**两段提交协议**  
( 2- Phase-Commitment Protocol)

# 两段提交协议 (1)

---

- 两段提交协议把一个分布事务的所有局部事务管理分为两类：**协调者（一个）**，**参与者**。
  - **协调者**：负责作出该事务是提交还是撤消的最后决定。
  - **参与者**：负责管理相应子事务的执行以及在各自局部数据库上执行写操作。

# 两段提交协议 (2)

---

- 两段提交协议内容

- 第一阶段：协调者征求意见作决定

- 协调者向所有参与者发出“准备提交”信息，并记入日志；参与者准备提交就回答“就绪”，否则回答“撤消”，并记入日志。
    - 如果在规定时间内，协调者收到所有参与者的“就绪”信息，则作出“提交”决定，否则将作出“撤消”决定。

- 第二阶段：参与者执行决定

- 协调者将有关决定写入日志，然后把这个决定发送给所有的参与者。
    - 所有参与者收到命令后，首先在日志中记入“收到提交/撤消决定”的信息，并向协调者发送应答消息，最后执行相应决定。
    - 协调者收到所有参与者的应答消息后，一个事务的执行到此结束。有关日志信息可以脱机保存。

- 采用两段提交协议后，当系统发生故障时，各场地利用各自有关的日志进行事务恢复。

# 并发控制 (1)

---

- 分布式数据库系统中的并发控制也可以采用封锁技术，但并发控制更加复杂：
  - 分布式数据库系统支持多副本；
  - 由于事务的分布执行，封锁的方法会引起全局死锁。

# 并发控制 (2)

---

- 分布式数据库系统并发控制进一步的策略：
  - 处理多副本封锁的几种可行方法：
    - 对写操作，要申请所有副本的X锁；对读操作，只要申请某个副本的S锁。
    - 无论是写操作还是读操作都要对大多数副本申请X锁或S锁。
    - 规定某个场地上的副本为主副本，所有的读、写操作均申请对主副本的封锁。
  - 解决全局死锁（两个以上场地上发生死锁）
    - 死锁检测及解除方式
    - 死锁预防，如对事务按某一标准进行排序，只允许事务按这一次序单向等待。

# 小 结

---

- 什么是分布式数据库系统，分布式数据库系统的特点
- 分布式数据库系统的模式结构
- 数据分片
- 分布透明性
- 分布式查询和优化
- 分布式事务处理技术

# 数据库系统的结构

